ACCEL: Accelerating the Bitcoin Blockchain for High-throughput, Low-latency Applications

Adiseshu Hari Murali Kodialam T.V. Lakshman
Nokia Bell Labs, USA
Email: firstname.lastname@nokia.com

***Abstract*—The Bitcoin blockchain is a secure, distributed ledger that enables trusted transactions across untrusted entities. However, many applications need much faster transaction confirmation than that of the current Bitcoin blockchain. In this paper, we present a high-throughput, low-latency, deterministic confirmation mechanism called ACCEL for accelerating Bitcoin's block confirmation mechanism. Our key idea for achieving faster confirmation is the quick identification of *singular* blocks that provably belong to the blockchain. While it is impossible to determine with certainty if a block belongs to a blockchain when network delays are unbounded, singular block detection exploits the fact that the end-to-end latency between Bitcoin miners is substantially lower than the inter-block spacing and can be assumed to be upper bounded. ACCEL is especially suitable for low-latency, permissioned blockchains, where the block spacing can be optimized to the blockchain's small latencies to greatly improve throughput. We evaluate ACCEL's performance with extensive simulations and with a real implementation built with minimal changes to and fully compatible with the Bitcoin blockchain. We show that with appropriate bounds on the end-to-end latency, it is possible to reduce transaction confirmation latencies to milliseconds with ACCEL, and so meet the performance needs of a wide range of applications.**

I. Introduction

Blockchains are secure, distributed ledgers that enable trusted transactions across untrusted entities. While it was popularized by the Bitcoin cryptocurrency, its use cases now span multiple applications for which a secure distributed ledger and transaction store is of fundamental importance. These include real estate, finance, smart property and networking applications.

A blockchain achieves distributed global consensus on the ordering of its constituent blocks. These blocks aggregate incoming transactions into the blockchain system using a *blockchain consensus algorithm* which lays down the rules for generating new blocks, accepting new blocks and removing existing blocks from the blockchain. A blockchain provides tamper resistance by including a cryptographic hash of the previous block in the current block. Since the previous block includes a hash of its predecessor and so on all the way to the original *genesis* block, each block therefore provides a cryptographic digest of the entire chain of previous blocks. As a result, it is not possible to modify a block within the blockchain without detection unless every subsequent block is also changed to reflect the new hash of the modified block. This large-scale subversion of the blockchain is inherently hard. Therefore, blockchains can be used as permanent, verifiable ledgers for recording transactions.

Unlike a centralized repository, the blockchain does not need any PKI-like root of trust and provides a distributed verifiable transaction log. The success of Bitcoin and its underlying blockchain technology has sparked a worldwide interest in using the blockchain for other applications that also require a distributed and secure transaction ledger. However, there are significant shortcomings in the Bitcoin blockchain that need to be overcome for it to be of use for many envisaged applications.

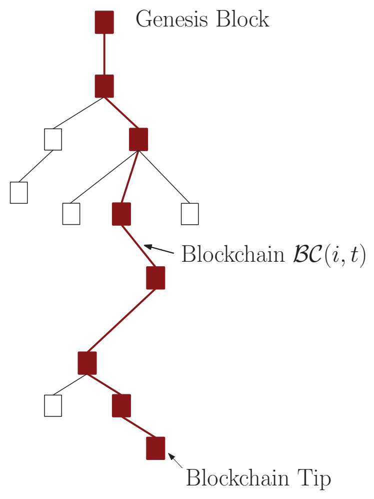


Fig. 1. Block Tree and the Corresponding Blockchain at user i at time t

Bitcoin uses a Proof-of-Work (PoW) consensus algorithm in which blocks can be generated (mined in Bitcoin terminology) by any user who is the first to complete a compute-intensive puzzle (repeated hashes until a value less than or equal to a threshold is generated, in Bitcoin). Each user maintains its view of the blockchain and adds to it by accepting the first block which advances the blockchain tip with the most cumulative work. It is possible for multiple blocks to be at the same height. This is because multiple miners may complete the puzzle at the same time leading to blockchain *forking*, in which different blockchain users advance their blockchain by accepting different blocks at the same level. This is illustrated in Figure 1 which highlights the blockchain $\mathcal{BC}(i,t)$ at a particular user i at a time t. The blockchain at each user is selected by choosing a unique block at each height of the *block tree* $\mathcal{BT}(i,t)$ resultant from blockchain forking. Non-zero block propagation delays means that different users might have different views of both the block tree and the blockchain at a given time t.

Forks are resolved when one chain advances faster than the others, leading to more cumulative work in that chain and causing the others to switch over to this chain. Therefore, fork resolution may cause the blockchain tip to switch at any time. If this happens, an accepted, confirmed block with all its transactions can be potentially replaced. Due to this rollback possibility, transaction confirmation in Bitcoin is probabilistic, with the confirmation probability increasing with a block's distance from the tip.

Probabilistic confirmation is one of the major drawbacks of the Bitcoin blockchain – only after waiting for multiple block confirmations, can we be highly confident that transactions in confirmed blocks will not be invalidated. This can cause excessive latencies in transaction confirmation.

The second major drawback of the Bitcoin blockchain is that it has a low write throughput that is limited by the block size and the fixed rate at which blocks are generated. The current Bitcoin blockchain has a write throughput of about 7 transactions per second [1].

These shortcomings of the Bitcoin blockchain are well known and alternate blockchain consensus mechanisms have been proposed that aim to solve some of the shortcoming, such as Proof-of-Stake [2], [3], Proof-of-Space [4], [5], Proof of Space-Time [6] and Proof-of-Elapsed-Time (PoET) [7]. Byzantine Agreement based consensus algorithms have also been proposed [8], [9], [10], as well as hybrid approaches such as [11], [12]. A comprehensive survey of recent blockchain developments is in [13].

This paper describes ACCEL, a new Bitcoin-compatible consensus algorithm that couples high transaction throughput with a low-latency and deterministic transaction and block confirmation mechanism. A key aspect of Bitcoin is that mining nodes tend to have low latency interconnections that bounds the end-to-end block propagation delay [14]. ACCEL exploits this bounded delay property to quickly locate *singular* blocks, which it leverages for improving the blockchain performance. We define singular blocks as blocks that are unique at a given height, i.e., they are not forked. Beyond Bitcoin, ACCEL is particularly suited for use with blockchains with known and bounded delays, characteristic of many blockchain use cases, in particular, permissioned blockchain applications. There are two key issues that we address in this paper. First is the problem of speeding up the blockchain. This targets permissioned blockchain implementations that can select a target block spacing. The second issue is the deterministic confirmation of transactions. This is applicable for both public blockchains like Bitcoin, and permissioned blockchains.

II. Blockchain Operation

The block chain starts off with a genesis block that is distributed to all users in the system. All subsequent blocks are generated using some previously generated block as predecessor. Since each block has a unique predecessor, the set of blocks form a tree. Each user maintains all the blocks that it has received so far in the form of a tree starting from the genesis block. The set of branches (children) from block B is the set of blocks that have declared block B as its predecessor. We refer to the tree of blocks that each user maintains as the **block tree**. The longest path in the block tree is the **block chain**. We use $\mathcal{BT}(i,t)$ to denote the block tree at user i and $\mathcal{BC}(i,t)$ to denote the block chain at user i at time t. Figure 1 shows the block tree and the corresponding blockchain at user i at time t_1. Note that the blockchain is the longest path in the block tree. More generally, the blockchain is the path in the block tree with the highest or most cumulative proof of work. We use the length of the path and the proof of work on a path interchangeably in the paper. User i attempts to add a block to the last block on the longest path of $\mathcal{BC}(i,t)$ which is called the tip of the blockchain. In general, transactions that are in blocks that are sufficiently deep in the blockchain are assumed to be confirmed transactions. However, there is no guarantee that the current blockchain may not be replaced by some other blockchain at a future time. For example, Figure 2 shows the block tree and the corresponding blockchains at user i at times t_1 and some subsequent time t_2. Note that the last five blocks in the blockchain at time t_1 have been replaced by new blocks at time t_2. While changes near the tip of the block chain can occur easily, it is quite unlikely to have large changes deep in the blockchain. This is the basis

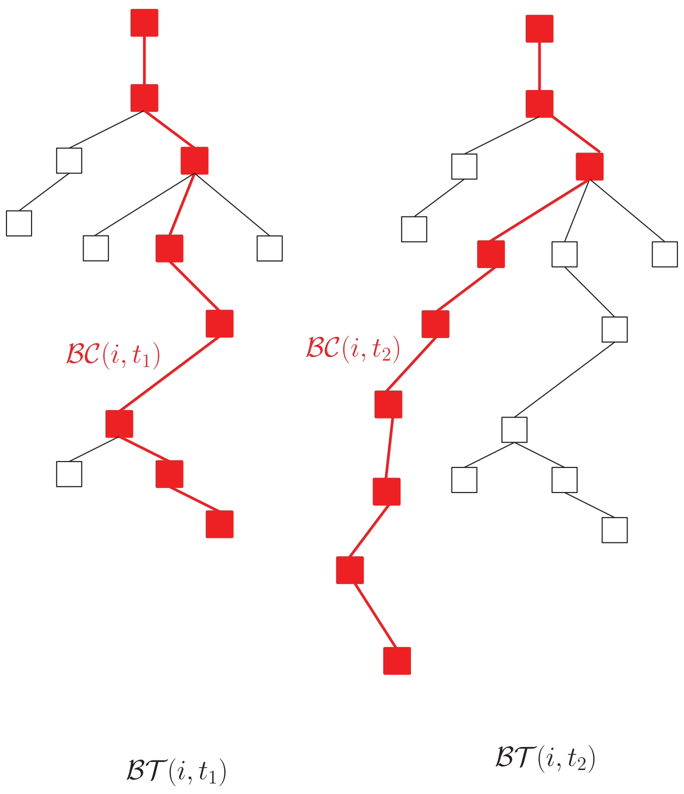


Fig. 2. Changes in the blockchain between times t_1 and t_2 at user i

for confirming transactions that belong in blocks that are in the blockchain and are sufficiently away from the tip of the blockchain. In the case of Bitcoin, any transaction that is in the blockchain that is six blocks away from the tip is assumed to be confirmed. In order to ensure that there are no significant changes in the blockchain far away from the tip, the process of adding new blocks to the blockchain is made computationally difficult.

A. *Rate of Block Addition*

The rate of block addition is controlled by changing the difficulty of adding a block to the blockchain. The process of adding a block to the blockchain is called mining. Once the user collects and verifies a subset of outstanding transactions, it includes the hash result of the block in the tip of its blockchain and a randomly generated nonce string and computes the hash of this combined string using a cryptographic hash function to a 256-bit hash value. The hash is considered successful if the resulting hash value is less than some pre-defined target. If the hash is not successful, then the hash is performed again with another nonce string.

Therefore, by controlling the target value, it is easy to control the rate at which blocks are added to the blockchain.

B. *Speeding Up the Blockchain*

The most obvious way to speed up the growth of the block tree is to make the hash target value large and this will result in blocks being added rapidly to the block tree. If the block addition rate is high, then it will result in significant branching of the block tree and it will be difficult to determine the stable blockchain. This is illustrated in Figure 3. Note that there are several competing paths that all have the same height. The tip of the current blockchain is block A. There is another chain with tip B that is the same height as block A. (Block A was picked since it was received earlier than B). There are two other blocks tips, C and D, that have height 6. Depending on the next few blocks that are received, the blockchain can either continue along block A or switch to blocks B, C or D. Therefore, it will be difficult to predict when a particular block can be considered as confirmed. Hence rapid growth of the block tree does not necessarily lead to faster confirmation of blocks. At the other extreme, if blocks are generated very

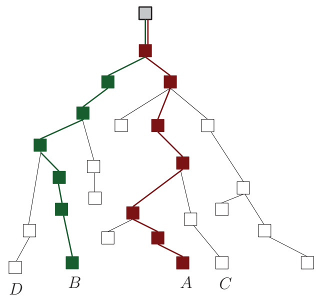


Fig. 3. High block generation rate results in a branched block tree

slowly, then the resulting block tree will be a single path. In this case, every block that is generated in the system will be part of the blockchain. However, the slow generation of blocks makes the blockchain growth very slow. Figure 4 shows the number of blocks at various heights from the genesis block when blocks are added at different rates to a blockchain. There are 10000 miners in the system and the λ represents the rate at which blocks are added to the block tree. The x-axis shows the height of the tree and the y-axis shows the number of blocks at a given height from the genesis block after 10000 blocks have been added to the block tree. The height of the block tree after 10000 blocks have been added are $1427, 7710$, and 9715 when $\lambda = 100, 5, 0.5$ respectively. From the plot note that the tree is very branched when $\lambda = 100$ as indicated by the large number of blocks at every height in the block tree. The number of blocks at most heights is one in the case where $\lambda = 0.5$. Therefore, blocks have to added in a controlled manner in

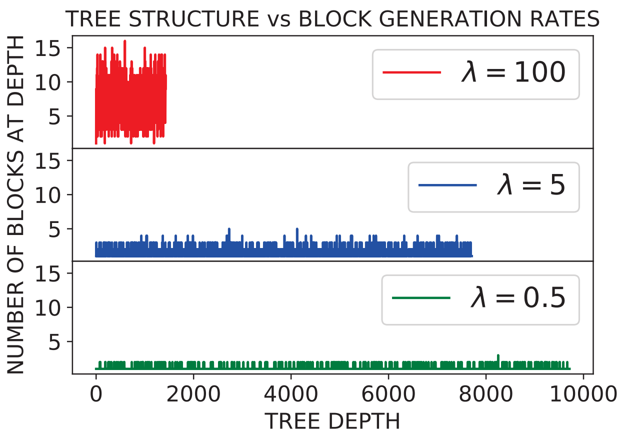


Fig. 4. Variation of Block tree structure with block generation rate λ after the addition of 10000 blocks

order to guarantee that the blockchain can be recognized in a timely manner.

C. *Deterministic Transaction Confirmation*

The other fundamental problem in standard blockchains is that if network delays are arbitrary, new blocks can be received at any point and time and can change the blockchain completely. For example, consider the network in Figure 5 in which nodes in the network are in two components that are separated by an unpredictable link with intermittent connectivity. The figure shows the block tree at node i in Component A at times t_1 and t_2. The open blocks are the blocks generated by nodes in component A and the solid blocks are the blocks generated in component B. Since the blocks from component B suffer long and variable delay a group of blocks can arrive after an arbitrary delay and change the blockchain at user i. Therefore, in general, in network with arbitrary delay, *it is not possible to guarantee that a block definitely belongs in the block chain.* In the rest of the paper, we consider blockchains in networks that have bounded delay. In these networks, we show that it is possible to definitively say when a block belongs in a blockchain.

III. Bounded Delay Blockchains

Given the impossibility of determining with certainty whether a block belongs to a blockchain when network delays are unbounded, we consider the case when all transmission delays in the network are upper bounded by Δ. In other words, any message will reach its intended recipient within a time of

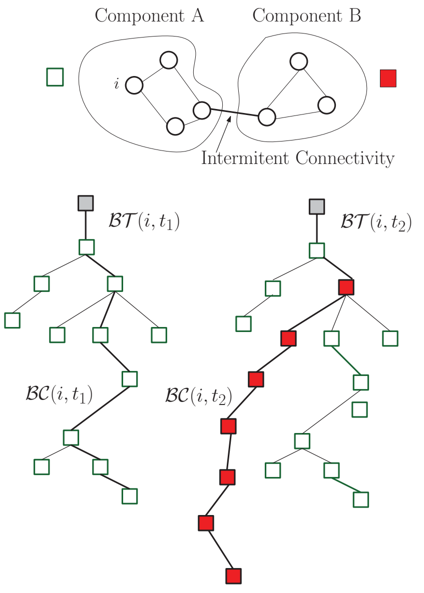


Fig. 5. Abrupt Change in blockchain due to large network delay

Δ of its transmit time. With this additional assumption, we show that we can definitively determine if a block belongs to the blockchain. Moreover, we show that this can be determined quickly and therefore, it is possible to speed up blockchains significantly compared to the case where network delays are unbounded. Before we outline the operation of a bounded delay blockchain, we first define some notation that we use in the rest of the paper. The system comprises of n users. User i can compute hashes at rate h_i. We use $h = \sum_i h_i$ to denote the total hash rate of the system. The hash is a 256-bit binary string. The hash is successful if the hash is less than some target value V. Therefore, the probability that a particular hash is successful is $\beta = V2^{-256}$. Let I denote the time between successive blocks introduced into the system. From the independence of the hashes,

$$\Pr[I > t] = (1-\beta)^{ht} \approx e^{-\beta h t}.$$

Lemma 1. *If the total hashing rate in the network is h hashes/second and the target value is V, then blocks are generated in a (approximately) Poisson process with rate $\lambda = \beta h$, where $\beta = V2^{-256}$ is the probability that a hash is successful.*

Therefore, the time between block introductions into the system is exponentially distributed and the mean time between blocks is $\frac{1}{\beta h}$. We use t_B to denote the generation time of block B and r_B^i the time at which block B reaches user i. Figure 6 shows the block generation and block arrival times at users i and j. From the maximum delay assumption, we know that $r_B^i - t_B \le \Delta$ for all blocks B for all users i. As stated earlier, we use $\mathcal{BT}(i,t)$ and $\mathcal{BC}(i,t)$ to denote the block tree and block chain at user i at time t. For any $B \in \mathcal{BT}(i,t)$, we use d_B to denote the height of block B. The height of block B is indicated in block B by its generating user. We use $N_d(i,t)$

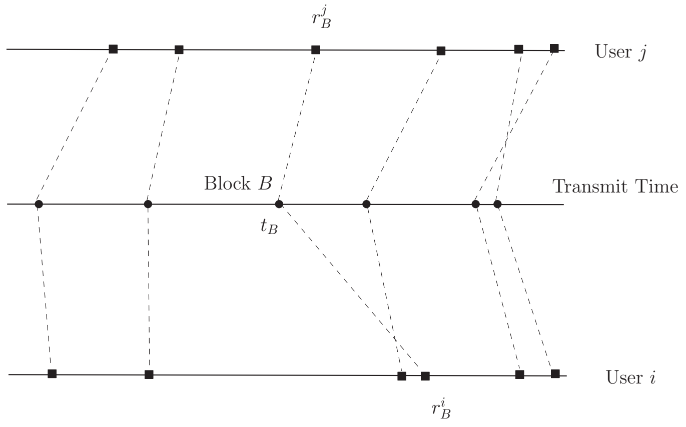


Fig. 6. Block Generation and Block Arrival Times

to denote the number of blocks at height d observed by user i at time t.

Definition 2. *A block B is called singular at user i at time t if*

$$N_{d_B}(i,t) = 1.$$

A block B is called permanently singular at user i if

$$N_{d_B}(i,\infty) = 1.$$

The reason permanently singular blocks are interesting is that every permanently singular block belongs to the blockchain.

Lemma 3. *If block B is permanently singular at user i, then it will be permanently singular at all other users. Moreover, block B belongs to the blockchain at all users.*

Proof Note that a block is permanently singular at all users or at none of the users. This is due to the fact that if a block B is permanently singular at user i and not at user j, then there is some block B' that is the same height as B at user j. Block B' will reach i (within a time of Δ of reaching j) and will be the same height as B. Therefore, B cannot be permanently singular at user i. We now show that every permanently singular block belongs in the blockchain. Every block in the blockchain has height one more than its predecessor. The blockchain starts at the genesis block with height zero. Since a permanently singular block is the only block at a given height, any path from the tip of the blockchain to the genesis block has to pass through a permanently singular block.

Since the permanently singular blocks belong to the blockchain, all blocks from the genesis block to the permanently singular blocks also belong to the blockchain. Therefore, if we can control the block generation rate such that there are sufficient number of permanently singular blocks and if we can identify these permanently singular blocks quickly, then we can confirm all the transactions belonging to all blocks from the genesis block to the permanently singular blocks. We now focus on the problems of

- Identifying permanently singular blocks quickly and efficiently.
- Controlling the block generation rate such that permanently singular blocks are generated at a sufficiently high rate.

In order to address these problems, we first give two different characterizations of permanently singular blocks.

A. *Permanently Singular Blocks*

We first outline a necessary and sufficient condition for a block to be permanently singular. This gives each user the means to identify permanently singular blocks quickly. Next we give a sufficient condition for a block to be permanently singular. This condition gives the protocol the means to derive a target block generation rate that ensures that permanently singular blocks are generated at a sufficiently high rate.

Theorem 4. *If the arrival time of block B at user i is r_B^i and it is singular at time $r_B^i + 2\Delta$ then block B is permanently singular,*

$$N_{d_B}(i, r_B^i + 2\Delta) = 1 \implies N_{d_B}(i, \infty) = 1.$$

Proof Since the arrival time of block B at user i is r_B^i, by time $r_B^i + \Delta$ block B has reached all users. Any block added after $r_B^i + \Delta$ has height greater than block B. Therefore, any block that is added between times t_B and $t_B + \Delta$ are the only blocks that can potentially be at the same height as block B. These blocks will reach all users by time $r_B^i + 2\Delta$. If a block is still singular at time $r_B^i + 2\Delta$, then it will be singular permanently and hence in the block chain. (Figure 7)

We now give a sufficient condition for a block to be singular. The condition basically states that block B is singular if no block is generated within a time of Δ before and after the generation time of the block B.

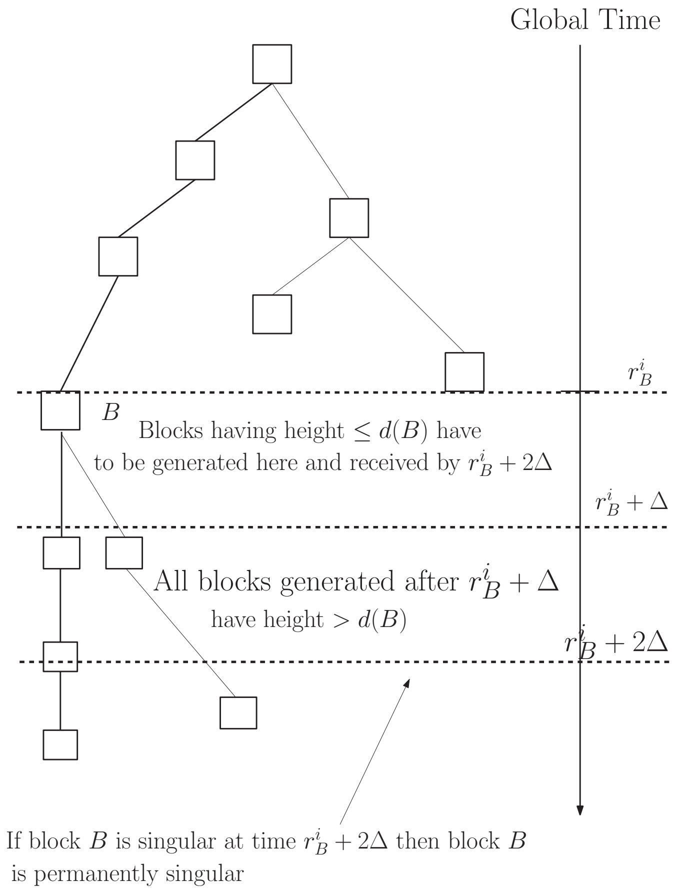
Lemma 5. *Block B is permanently singular if $t_B - t_{B-1} > \Delta$ and $t_{B+1} - t_B > \Delta$.*

Proof Since there is a gap of Δ before the generation time of block B it implies that all blocks up to and including block $B - 1$ have reached all users before block B is generated. Therefore, the height d_B of block B is one greater than the height of any other block in the system. Since no block is generated for time Δ after block B is generated, block B reaches all users. Any block generated after this will have a height one more than the height of block B. Therefore, block B is the only block at height d_B and hence it is singular. (See Figure 8 for an illustration of the proof.)

Lemma 5 can now be used to control the block generation rate in order to generate permanently singular blocks at a sufficiently rapid rate. The optimal block generation rate will depend on the performance metric that we would like to optimize. We now outline the performance metric and derive the optimal block generation rate to optimize the metric.

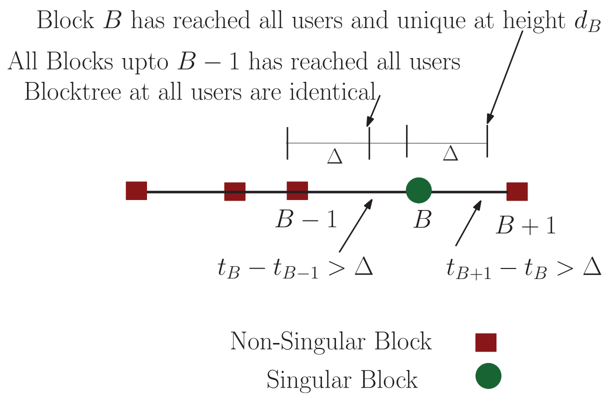
IV. Expected Confirmation Delay

The performance metric that we are primarily interested in is the time elapsed from the arrival of a transaction in the system until the transaction is confirmed. We call this the *confirmation delay*. Let T_j denote the time at which transaction j enters the system. Let C_j denote the time at which transaction j is included in a block that is confirmed to be part of the block chain. Then $C_j - T_j$ is the confirmation delay for transaction j. Note that a transaction may be confirmed at different time at different users. We take C_j to be the time at which the last user confirms transaction j. The metric that we are interested in is the average confirmation delay. In the following analysis, we assume that a block can contain all the transactions that



Global Time
r_B^i
B
Blocks having height $\leq d(B)$ have
to be generated here and received by $r_B^i + 2\Delta$
$r_B^i + \Delta$
All blocks generated after $r_B^i + \Delta$
have height $> d(B)$
$r_B^i + 2\Delta$
If block B is singular at time $r_B^i + 2\Delta$ then block B
is permanently singular

Fig. 7. Identifying Singular Blocks



Block B has reached all users and unique at height d_B
All Blocks upto $B-1$ has reached all users
Blocktree at all users are identical
Δ
Δ
$B-1$
B
$B+1$
$t_B - t_{B-1} > \Delta$
$t_{B+1} - t_B > \Delta$
Non-Singular Block 🟥
Singular Block ●

Fig. 8. Sufficient condition for a block to be permanently singular

are currently visible to the user that are not already included in the blockchain. We address the issue of setting the appropriate blocksize in Section V-B. The confirmation delay comprises of five components as shown in Figure 9:

1) Time taken for the transaction to be visible to all the users.
2) Time for the transaction to be included in a block.
3) Time to generate the first singular block after the transaction is included in a block.
4) Time for the singular block to reach all users.
5) Time to recognize the singular block.

The different times are illustrated in Figure 9. We now outline

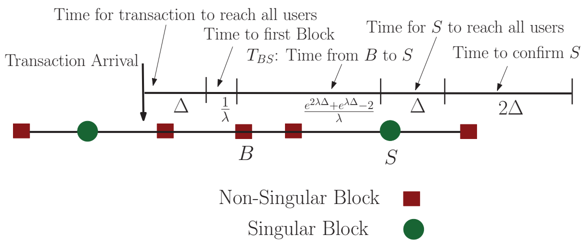

Time for transaction to reach all users
Time to first Block
Time for S to reach all users
Transaction Arrival
T_{BS}: Time from B to S
Time to confirm S
Δ
$\frac{1}{\lambda}$
$\frac{e^{2\lambda\Delta}+e^{\lambda\Delta}-2}{\lambda}$
Δ
2Δ
B
S
Non-Singular Block 🟥
Singular Block ●

Fig. 9. The different time components in confirmation process

the expected value of the time components in the *worst case.* Assume that transaction j is generated somewhere in the system at time T_j. Since the maximum network delay is bounded by Δ, by time $T_j + \Delta$, the transaction will be visible to all users in the system. Therefore, all blocks generated after time $T_j + \Delta$ will include transaction j. Once it is visible to all users, the time taken to generate the first block is the forward recurrence time in a Poisson process with rate λ which is $\frac{1}{\lambda}$. This is the second component of the delay. The third component of the delay is the time from this block to a singular block. We derive this in the next section. The fourth component is the time taken for this singular block to reach all users and this time is upper bounded by Δ. The fifth component is the time for the last user receiving the singular block to confirm the block as singular. This takes time 2Δ as in Theorem 4.

A. Expected Time to Singular Block

From Lemma 5, we know that a block is singular if no block is generated within a time of Δ before and after this block is generated. Blocks are generated at rate $\lambda = \beta h$ and the inter-block arrival times are exponentially distributed with mean $\frac{1}{\lambda}$. We define

$$p = e^{-\lambda\Delta}$$

to denote the probability that an inter-block time is greater than Δ. Let I denote the block inter-arrival time. Since I is exponentially distributed, it is easy to show that

$$\begin{aligned}\Delta^+ &= E[I|I > \Delta] = \Delta + \frac{1}{\lambda}\\ \Delta^- &= E[I|I \le \Delta] = \frac{1}{\lambda} - \frac{p\Delta}{1-p}\end{aligned}$$

Let T_{BS} denote the time between a random block B and the closest singular block S. We want to compute $E\big[T_{BS}\big]$. We define $T^-_{BS} = E[T_{BS}|t_B - t_{B-1} \le \Delta]$ and $T^+_{BS} = E[T_{BS}|t_B - t_{B-1} > \Delta]$. Figure 10 shows the computation of T^-_{BS} and T^+_{BS} by conditioning on the value of $t_{B+1} - t_B$.

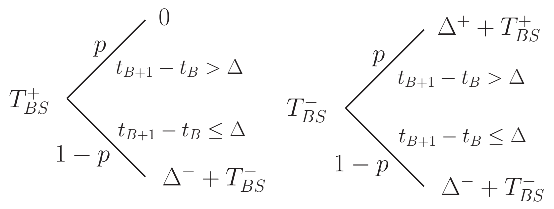


Fig. 10. Computation of T_{BS}

From the Figure 10, note that

$$\begin{aligned}T^+_{BS} &= (1-p)\Big(\Delta^- + T^-_{BS}\Big)\\ T^-_{BS} &= p\Big(\Delta^+ + T^+_{BS}\Big) + (1-p)\Big(\Delta^- + T^-_{BS}\Big)\end{aligned}$$

Solving these simultaneous equations, we obtain

$$\begin{aligned}T^-_{BS} &= \frac{1}{\lambda}\Big(\frac{1+p+p^2}{p^2}\Big) - \Delta\\ T^+_{BS} &= \frac{1}{\lambda}\Big(\frac{1-p^2}{p^2}\Big) - \Delta\end{aligned}$$

We can now compute

$$\begin{aligned}E\big[T_{BS}\big] &= pT^+_{BS} + (1-p)T^-_{BS}\\ &= \frac{1}{\lambda}\Big(\frac{1}{p^2} + \frac{1}{p} - 2\Big) - \Delta\end{aligned}$$

Therefore, for a given block generation rate λ, the expected confirmation delay $E[D]$ can be computed by summing the five components shown in Figure 10:

$$E[D] = \frac{1}{\lambda}\Big(\frac{1}{p^2} + \frac{1}{p} - 1\Big) + 3\Delta$$

We now compute the block generation rate λ^* that minimizes the expected confirmation time.

Theorem 6. *Given the maximum network delay of Δ, the expected confirmation delay for a randomly arriving transaction is minimized when the block generation rate*

$$\lambda^* = \frac{0.470}{\Delta}$$

and the corresponding value of $E[D] \le 9.73\Delta$.

Proof

$$\begin{aligned}E[D] &= \frac{1}{\lambda}\Bigg(\frac{1}{p^2} + \frac{1}{p} - 1\Bigg) + 3\Delta\\ &= 3\Delta + \frac{\Delta}{\lambda\Delta}\Bigg(e^{2\lambda\Delta} + e^{\lambda\Delta} - 1\Bigg)\end{aligned}$$

Setting $\lambda\Delta = y$ the minimum value of

$$\Delta\Bigg(3 + \frac{1}{y}(e^{2y} + e^y - 1)\Bigg)$$

occurs at $y = 0.47$ and is 9.73. The optimum value of λ denoted by λ^* corresponding to $y = 0.47$ is

$$\lambda^* = \frac{0.47}{\Delta}$$

and this results in

$$E[D] \leq 9.73\Delta.$$

Thus far we have not made any assumption about the distribution of network delay. If the delay distribution is known then it is possible to derive tighter bounds on the confirmation delay. In order to check the validity of the expressions derived above, we conducted a set of simulation experiments. The system comprises of a set of users processing a set of transactions. In the experiments, the number of users varied between 100 and 10000 over time. In the first set of experiments, the value of $\Delta = 6ms$ and the system processes 2000 transactions per second. The experiment was run for 500 seconds corresponding to roughly a million transactions. The block generation rate is varied from 10 blocks per second to 200 blocks per second. The value of $\lambda^* \approx 80$ blocks per second. The block size is set to 2000 transactions/block. (We comment on the block size in Section V-B). Figure 11 shows the variation of expected confirmation time as a function of the block generation rate. There is excellent agreement between the theoretical and experimental results. In the next set of experiments, the value of $\Delta = 60ms$. We process 200 transactions per second . The block generation rate is varied from 1 block per second to 20 blocks per second. The value of $\lambda^* \approx 8$ blocks per second. The block size is set to 2000 transactions/block. Again, note the close agreement between the theoretical and simulated delay values.

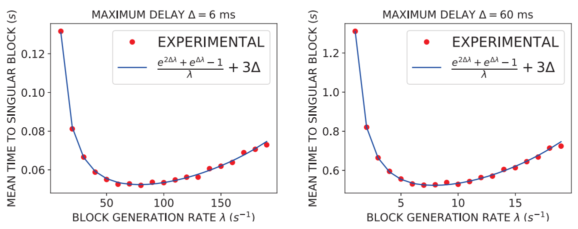


Fig. 11. Comparison of Theoretical and Experimental results for the variable portion of confirmation delay

In the second set of experiments, we measured the cumulative distribution function of the confirmation delay in the same two set up discussed above. We performed two sets of experiments. In the first set of experiments, the delay between any pair of users was set to Δ and in the second set of experiments the delay between any pair of users was set to a uniform distribution between 0 and Δ. Figure 12 shows the results. The mean confirmation delay value in both cases agree very well with the theoretical values when the delay is deterministic Δ. In the case where the delays are uniform random, the average confirmation delay is smaller since the mean delays are smaller. Note however, that it still takes 2Δ to confirm a singular block. The arrival rate is set to the value of λ^* for each case.

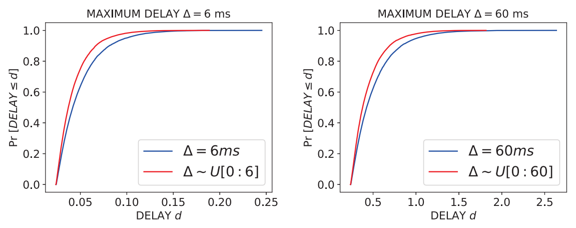


Fig. 12. Transaction delay CDF with Fixed and Uniform Delay when $\lambda = 50$

V. ACCEL Blockchain Operation

Each node receives and processes blocks the same way as in a regular PoW based blockchain such as Bitcoin, using the same type of block and transaction structure found in them. Note that ACCEL is opaque to transaction semantics and can be used for Ethereum style blockchains which have a richer transaction semantics than Bitcoin, but still follow PoW based consensus algorithm. One difference, discussed in Section V-A, is that the blockchain's Δ is used to derive the optimal block generation rate λ^* which determines the target block spacing. The major difference is the singular block detection illustrated in the flowchart shown in Figure 13. A block is a singular candidate if it is the first incoming block at a given height. If so, it causes a timer to be set for a duration of 2Δ. The incoming block is not a singular candidate if another block at the same height arrives, causing the timer to be reset. When the timer elapses, the block is known to be singular. Once a block's singularity is established, all transactions in this block as well as all preceding blocks in the blockchain till the previous singular block are deterministically confirmed as part of the singular block processing. Omitted from the figure is the non-singular block processing, which are added to the block tree. Just as in existing PoW algorithms, these are used to advance the blockchain based on following the branch with the most cumulative work.

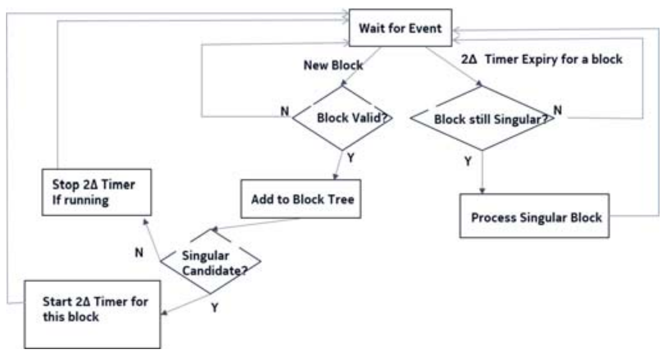


Fig. 13. ACCEL block processing

A. Dynamically Adjusting the Hash Target Value

ACCEL uses the same mechanism as the bitcoin blockchain to adjust the difficulty of mining. Once every R blocks, the target value V that determines the success of hashing is recomputed to adjust to the changing hashing power in the network. The main difference between the bitcoin rate adjustment mechanism and the rate adjustment mechanism in ACCEL is that ACCEL adjusts the hash target value to ensure that blocks are generated at rate λ^*, which is specific to the particular ACCEL use case's latency and throughput requirements. A low value for R will allow the blockchain's hash power to match the target rate λ^* more quickly if there is a change in the hash power due to users joining or leaving the blockchain, while a higher value for R will lead to lower variance in the instantaneous block rate when the hash power is stable.

B. Setting the Block Size

At the time of writing, the Bitcoin blockchain had a 7-day transaction confirmation average time of 2351 minutes [15]. To avoid excessive delays, the blockchain needs to provide the throughut needed for expected peak transaction volumes in order to meet transaction confirmation targets. Given a target block rate, a permissioned blockchain can tune its transactional throughput by setting a block size big enough to accommodate expected peak transaction bursts. This ensures that transactions are inserted in the first available block with a high probability. Assume that the transaction arrival rate is T. Given the maximum delay Δ and the block generation rate λ the blockchain grows at rate

$$G = \frac{1}{\frac{1}{\lambda} + \Delta}.$$

This can be seen as follows: Let t_B denote the generation time of block B at the maximum height in the blockchain. At time $t_B + \Delta$, block B has reached all users. Therefore, any block that is added after time $t_B + \Delta$ will have height greater than the height of B in the block tree. The inter-block generation time is exponential with mean $\frac{1}{\lambda}$. Therefore, the expected time from t_B until the block chain extends by one unit is at most $\Delta + \frac{1}{\lambda}$. Since each transaction belongs to the blockchain, it implies that roughly the average number of transactions in each block of the blockchain is given by T/G. This represents the average number of transactions per block. We size the blocksize to be roughly 5 to 10 times this value to ensure that very few transactions are delayed due to the unavailability of space in the first available block. With this block size, we find that there is no increase in the block confirmation delay

VI. Effect of Delay Violations

In this section we outline the performance of ACCEL when some blocks violate the delay bound Δ. In order to stress test the algorithm, we randomly drop blocks that are on the way to the user with some *loss probability* q. In this case, there is a chance that a block that is not singular can be misidentified as singular (false positive), since a block that is at the same height as the block may not have reached the user. Note that there will not be any false negatives. We track the number of such false positive identifications as a fraction of the total number of singular block identifications (number of singular blocks times the number of users). A block is permanently singular if it remains singular for a time 2Δ after the arrival of the block. In order to accommodate losses, we make the following modification to ACCEL. Instead of a factor of 2Δ we use $\gamma\Delta$ for some value of $\gamma > 2$. We call γ the *singular block identification factor*. This ensures that if a block does not reach a user, one of its descendants will reach the user and the user can use this block to infer the missing (or delayed) block. If a block did not have any descendants, then it does not cause any problems with inference of singular blocks. In Figure 14, we plot the ratio of the number of false positive singular blocks to the total number of singular block identifications as a function of the singular block identification factor. Note the exponential drop in the false positive fraction as the singular block identification factor increases. This suggests that using γ slightly greater than two can be used to guard against occasional delayed or lost packets at the expense of slightly longer confirmation times.

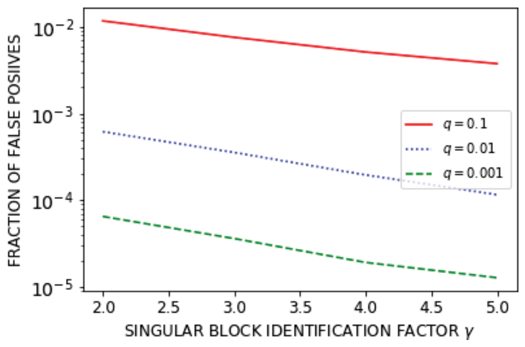


Fig. 14. Performance of ACCEL with delay violations

VII. Attacks on ACCEL

Apart from the standard attacks on a distributed ledger that is addressed in the Bitcoin blockchain, another potential attack on ACCEL is for malicious users to attempt to reduce the number of singular blocks. This can be done as follows: When a malicious user receives a block B, it immediately generates another block B' whose predecessor is the same as B and sends it out to the network. This implies that if block B originally was a potential singular block, it will no longer be one since B' will be the same height as B. Though this attack does not affect the correctness of the blockchain, it can delay transaction confirmation. In order to study the effect of this attack, we run simulations using the same parameters as in Section IV-A. We increase the fraction of malicious hashes from zero to 50% and observe the expected confirmation delay. We show the result in Figure 15. Note that there is only a gradual increase in the expected confirmation time. Therefore, there has to be a very large fraction of malicious hashes in the network in order to increase the confirmation time significantly. If the identity of the users is known, as in a permissioned blockchain, it will be relatively easy to detect gross violations by observing the block generation rate of users and not accepting blocks from users whose block generation rate is much higher than the expected rate set by ACCEL.

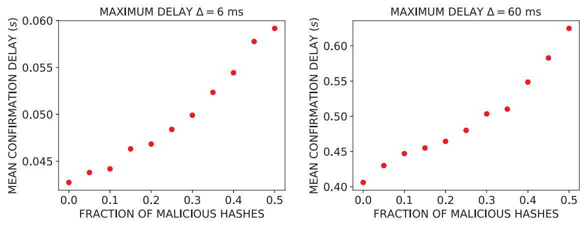


Fig. 15. Performance of ACCEL with malicious users

VIII. ACCEL TESTBED IMPLEMENTATION

We created a Docker based, containerized ACCEL testbed platform based on the official Bitcoin Core client. Each blockchain instance can be run in its own container with full control over its start time, stop time and connectivity to other nodes. Each container can host a miner or a transaction generator. A virtual network links all the containers together via another container based ACCEL node acting as a blockchain proxy. The virtual network can be programmed to have a specific delay and drop probability to mimic various network conditions and to stress test the testbed implementation.

We modified the Bitcoin client code to make it parameterizable. Configurable parameters for the testbed include the number of miners, as well as the virtual network latency and jitter.

Performance Results. Our results were obtained by running the ACCEL testbed on a 2.6GHz Xeon E5-2690 based system with 26GB RAM and 14 physical cores, each supporting 4 hyperthreads.

Similar to our simulations, we run our tests with 6ms and 60ms Δ message delay bounds, and measure the time taken for a transaction to be deterministically confirmed by the first singular block received after the transaction receives a block confirmation. This singular block confirmation time is averaged over all the blocks and the mean is displayed in the graph below.

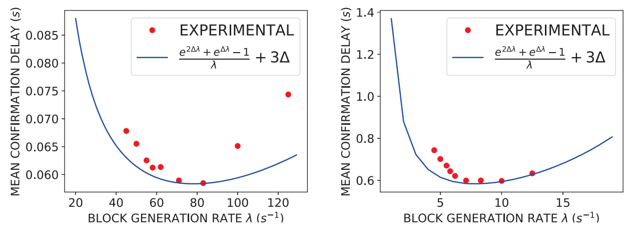


Fig. 16. Deterministic Transaction Confirmation Delay in the ACCEL testbed

Figure 16 shows the mean singular block confirmation for transactions for the cases when the network delay bound is set to 6ms and 60ms respectively in the ACCEL testbed. Both delay bounds are tested with a variety of block rates and compared to the expected simulation results as shown in the figure. As can be seen, both the experimental and the simulation results are similar, with marginal differences between the simulated and experimental block rates for minimizing transaction confirmation delays. The slight offset is because the simulation results do not account for the overhead of application processing at high block rates, and also due to the precision limitations in the testbed's generation of virtual network delays. These lead to a lower level of forking at lower block rates compared to the simulation results.

IX. FUTURE WORK

We intend to study the use of ACCEL for networking applications. Network applications of blockchains include user authentication [16], domain authentication [17], a decentralized Internet [18], PKI replacement [19] and RPKI replacement [20].

REFERENCES

[1] " Bitcoin Scalability ," https://en.bitcoin.it/wiki/Scalability.
[2] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without Proof of Work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 142–157.
[3] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.
[4] G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi, "Proofs of Space: When Space is of the Essence," in *International Conference on Security and Cryptography for Networks*. Springer, 2014, pp. 538–557.
[5] S. Park, K. Pietrzak, A. Kwon, J. Alwen, G. Fuchsbauer, and P. Gai, "SpaceMint: A Cryptocurrency Based on Proofs of Space," Cryptology ePrint Archive, Report 2015/528, 2015, https://eprint.iacr.org/2015/528.
[6] T. Moran and I. Orlov, "Rational Proofs of Space-Time," Cryptology ePrint Archive, Report 2016/035, 2016, https://eprint.iacr.org/2016/035.
[7] "Proof of Elapsed Time," https://sawtooth.hyperledger.org/.
[8] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance And Proactive Recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.
[9] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements For Cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.
[10] "Hyperledger Fabric," https://hyperledger.org/projects/fabric.
[11] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A Scalable Blockchain Protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 45–59.
[12] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing."
[13] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, June 2017, pp. 557–564.
[14] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10.
[15] " Bitcoin Average Transaction Confirmation Time, 2018/01/26 ," https://blockchain.info/charts/avg-confirmation-time?daysAverageString=7.
[16] " BitID ," https://github.com/bitid/bitid.
[17] "NameCoin," https://namecoin.info.
[18] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A Global Naming and Storage System Secured by Blockchains," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. Denver, CO: USENIX Association, Jun. 2016, pp. 181–194. [Online]. Available: https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali
[19] C. Fromknecht, D. Velicanu, and S. Yakoubov, "CertCoin: A NameCoin Based Decentralized Authentication System 6.857 Class Project," 2014.
[20] A. Hari and T. V. Lakshman, "The Internet Blockchain: A Distributed, Tamper-Resistant Transaction Framework for the Internet," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16. New York, NY, USA: ACM, 2016, pp. 204–210. [Online]. Available: http://doi.acm.org/10.1145/3005745.3005771