On the Performance of Pipelined HotStuff

***Abstract*—HotStuff is a state-of-the-art Byzantine fault-tolerant consensus protocol. It can be pipelined to build large-scale blockchains. One of its variants called LibraBFT is adopted in Facebook's Libra blockchain. Although it is well known that pipelined HotStuff is secure against up to $1/3$ of Byzantine nodes, its performance in terms of throughput and delay is still under-explored. In this paper, we develop a multi-metric evaluation framework to quantitatively analyze pipelined HotStuff's performance with respect to its chain growth rate, chain quality, and latency. We then propose several attack strategies and evaluate their effects on the performance of pipelined HotStuff. Our analysis shows that the chain growth rate (resp, chain quality) of pipelined HotStuff under our attacks can drop to as low as $4/9$ (resp, $12/17$) of that without attacks when $1/3$ nodes are Byzantine. As another application, we use our framework to evaluate certain engineering optimizations adopted by LibraBFT. We find that these optimizations make the system more vulnerable to our attacks than the original pipelined HotStuff. Finally, we provide two countermeasures to thwart these attacks. We hope that our studies can shed light on the rigorous understanding of the state-of-the-art pipelined HotStuff protocol as well as its variants.**

I. Introduction

In 2008, Nakamoto invented the concept of blockchain, a mechanism to maintain a distributed ledger for the cryptocurrency Bitcoin [1]. The core novelty behind blockchain is Nakamoto Consensus (NC), an unconventional (at that time) synchronous Byzantine fault-tolerant (BFT) consensus [2]–[5]. Despite the huge impact of Bitcoin, NC suffers from long confirmation latency and low transaction throughput, both of which hinder the original blockchain to support Internet-scale applications. For example, Bitcoin today can only process up to seven transactions per second with a confirmation latency of hours. On one hand, the long latency is a result of the probabilistic safety guarantee: a short latency cannot guarantee high confidence that a transaction has been confirmed. On the other hand, the low throughout is mainly due to the speed-security tradeoff: a higher transaction throughput leads to more severe forking, which greatly reduces the honest computation power against adversaries, making the system less secure.

One promising approach to addressing these dilemmas is leveraging the classical BFT consensus [6], [7], which is also referred to as BFT state machine replication (SMR) and has been extensively studied for the last few decades. Unlike NC, classical BFT protocols can provide a strong safety guarantee. That is, once a transaction is confirmed, it will stay there forever. Hence, clients do not need long waiting periods to confirm these transactions (which means a shorter transaction latency), and transaction processing does not need to be compromised with security (which implies a higher throughput). For example, experiments have demonstrated that PBFT [7], a pioneer BFT protocol, can proceed tens of thousands of transactions per second in a LAN [8] and have only hundreds of milliseconds latency in a WAN [9]. Despite all of these advantages, it is technically challenging to apply classical BFT protocols in a blockchain setup. First, the classical BFT protocols have a high message complexity (e.g., $O(n^2)$ message complexity for committing one block), so the number of participants is usually less than dozens [10], [11]. In other words, classical BFT protocols suffer from scalability issues and cannot support a large-scale blockchain. Second, classical BFT protocols are notoriously difficult to develop, test and prove [11]–[13]. Finally, classical BFT protocols rarely consider fairness among leaders[1]. In most leader-based BFT protocols [7], a node can serve as a leader as long as it behaves well. This is also called stability-favoring leader rotation [14], for this mechanism can avoid the $O(n^3)$ message complexity in the leader rotation.

HotStuff proposed by Yin et al. [15] is a state-of-the-art BFT consensus, aiming to address these issues. HotStuff creatively adopts a three-phase commit rule (rather than the two-phase commit rule used in classical BFT [7]) to enable the protocol to reach consensus at the pace of actual network delay[2] and leverages the threshold signature to realize linear message complexity. HotStuff can be further pipelined, which enables a frequent leader rotation and leads to a simple and practical approach to building large-scale blockchains. Due to these salient properties, Facebook adopts a variant of pipelined HotStuff called LibraBFT [16] for its global payment system, Libra blockchain, which aims to empower billions of people.[3] In addition, Dapper Lab describes how to deploy HotStuff in its Flow platform [17], and Cypherium Blockchain [18] combines HotStuff with NC together to build a permissionless blockchain. Although it is well known that pipelined HotStuff is secure against up to $1/3$ of Byzantine nodes, its performance in terms of throughput and delay is still under-explored.

In this paper, we first develop a multi-metric evaluation framework to quantitatively analyze pipelined HotStuff's performance with respect to its chain growth rate, chain quality, and latency. We then propose several attack strategies and evaluate their effects on the performance by using our framework. In addition, we use our framework to evaluate some engineering optimizations adopted by LibraBFT. We find that these optimizations make the system more vulnerable to certain attacks compared with the original pipelined

[1]Blockchain systems usually reward leaders with some self-issued tokens for incentivizing protocol participation [1]. Hence, leadership fairness is the foundation of such an incentive mechanism to fairly reward nodes.

[2]This property is called responsiveness in the literature.

[3]Pipelined HotStuff is also referred to as chained HotStuff [15].

HotStuff. Finally, we provide two countermeasures to thwart these attacks. We hope that our studies can shed light on the rigorous understanding of the state-of-the-art pipelined HotStuff protocol as well as its variants. Our contributions can be summarized as follows:

- We develop a multi-metric evaluation framework and leverage it to evaluate the impact of several new attacks. Our analysis shows that the chain growth rate (resp, chain quality) of pipelined HotStuff under these attacks can drop to $4/9$ (resp, $12/17$) of that without attacks when $1/3$ nodes are Byzantine.
- We use our framework to evaluate some engineering optimizations adopted by LibraBFT. We find that in pipelined HotStuff, an adversary controlling $1/3$ corrupted nodes can increase the latency to 8.33 rounds on expectation (2.7x times of the latency without attacks), however, with the same condition, the adversary in LibraBFT can increase the latency to 10.25 rounds.
- We propose two countermeasures against our attacks, which can reduce the latency by 3 rounds, improve the chain growth rate by 1.5x times and chain quality by 1.2x times.
- We develop a proof-of-concept implementation of pipelined HotStuff to validate our theoretical findings.

II. System Model and Preliminaries

A. System Model

We consider a system with n nodes denoted by the set $\mathcal{N}$. We assume a public-key infrastructure (PKI), and each node has a pair of keys for signing messages (e.g., blocks and votes). We assume that a subset of f nodes is *Byzantine*, denoted by the set $\mathcal{F}$, and can behave arbitrarily. The other nodes in $\mathcal{N} \setminus \mathcal{F}$ are honest and strictly follow the protocol. In order to ensure security, we have $n \geq 3f + 1$. We use α (resp. β) to denote the fraction of Byzantine (resp. honest) nodes. That is, $\alpha = f/n$. For simplicity, all the Byzantine nodes are assumed to be controlled by a single adversary, which is computationally bounded and cannot (except with negligible probability) forge honest nodes' messages.

We assume honest nodes are fully and reliably connected, i.e., every pair of honest nodes is connected with an authenticated and reliable communication link. We adopt the partial synchrony model of Dwork *et al*. [19]. In the model, there is a known bound Δ and an unknown Global Stabilization Time (GST), such that after GST, all message transmissions between two honest nodes arrive within a bound Δ. Hence, the system is running in *synchronous* mode after GST and *asynchronous* mode if GST never occurs.

B. Preliminaries

Quorum Certificate. A block's quorum certificate (QC) contains a set of signatures of the block's hash digest created by a quorum of nodes. More precisely, a quorum consists of more than $2n/3$ nodes (out of n). We say a block is certified when its QC is received, and certified blocks are ranked by their round numbers. Each node keeps track of all signatures for all blocks and keeps updating the highest certified block to its knowledge.

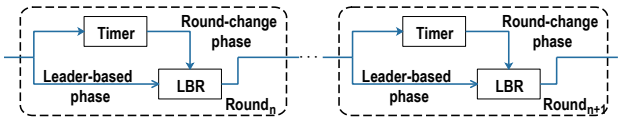


Fig. 1. **Overview of a sequence of the leader-based round (LBR) instances.** The leader-base phase is in charge of driving progress, while the round-change phase is to synchronize nodes to the same round.

Block and Block Tree. Clients send transactions to leaders, who then batch transactions into blocks. A block has four-tuple $\langle round, cmd, parent\_qc, \sigma \rangle$, where $round$ denotes the round number at which the block is proposed, cmd is a batch of transactions, $parent\_qc$ is the QC for the parent block, and σ is the block owner's signature of $\langle round, cmd, parent\_qc \rangle$. Every block except the genesis block must specify its parent block and include a QC of the parent block. In this way, blocks are chained. (Note that in Bitcoin [1], blocks are chained through hash references rather than QCs.) A block's height is its distance from the genesis block. As there may have forking blocks, each node maintains a block tree (referred to as $blockTree$) of received blocks.

C. Leader-based Round Abstraction

Pipelined HotStuff is executed into a sequence of rounds, and each round has a designated leader[4]. Each round can be further divided into two phases: 1) leader-based phase in which a designated leader proposes a new block and collects votes from other nodes to form a quorum certificate of this block (introduced shortly), and 2) round-change phase in which nodes safely *wedge* to next round if the current-round leader is faulty or no certified block is generated before the timeout, as shown in Fig. 1. By following recent work [22], we assume that each round can be encapsulated in a leader-based round (LBR) abstraction, which provides two important modules: pacemaker and leader-election modules. The pacemaker module can guarantee that honest validators are synchronized to execute the same rounds for sufficient overlap, and leaders propose a block that will be supported by honest nodes when the network is synchronous [15]. That is, an honest leader can send its block proposal to all the other honest nodes and receive their votes in one round. The leader-election module can guarantee that nodes are *fairly* elected as leaders [23]. That is, during synchronous periods, each node has the same chance to win the leadership for one round[5]. For convenience, a leader who is elected from honest nodes (resp. Byzantine nodes) is referred to as *honest* (resp. *adversarial*) leader. In the same way, a block proposed by an honest (resp. adversarial) leader is referred to as *honest* (resp. *adversarial*) block.

[4]Rounds are also referred to as views [7], [15], terms [20], instance values/epoch [21] or ballot numbers in the literature

[5]During asynchronous periods, some honest nodes may not be synchronized to the highest round and participate in the leader-election. Thus, the adversary can have a higher chance than α to be elected as leaders.

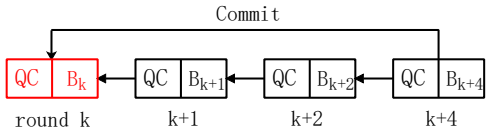


Fig. 2. **The chained blocks in pipelined HotStuff.** Blocks B_k, B_{k+1}, and B_{k+2} are three consecutive blocks, and nodes will commit block B_k when receiving block B_{k+4}.

III. Pipelined HotStuff and LibraBFT Algorithms

A. Pipelined HotStuff Algorithm

We describe the leader-based phase of pipelined HotStuff at round i. At the beginning, a unique leader, called $leader_i$, is randomly elected by the leader-election module. The leader's identity is known and can be verified by all nodes (see Sec. II-B). Then, the leader **proposes** a block to extend the *highest* certified block it has seen[6], and broadcasts this block to all other nodes. Every node **votes** for the first block it receives from the leader[7], as long as the block satisfies certain conditions introduced shortly. A vote for a block is a signature on the block's hash digest. When the leader receives at least $2n/3$ unique signatures (including its own signature), it aggregates these signatures into a QC, and sends it to the next-round leader, namely, $leader_{i+1}$.

We are now ready to describe the condition for voting. Every node maintains two local parameters: i) $last\_voted\_round$, the last round for which the node has voted, and ii) $locked\_round$, the highest known round number of the grandparent block that has been received. For example, in Fig. 2, when a node first receives the block B_{k+2} and votes for it, its $last\_voted\_round$ is updated to $k+2$, and the highest grandparent block is B_k, and so $locked\_round$ is k. After receiving block B_{k+4}, the node updates $last\_voted\_round$ to $k+4$ and $locked\_round$ to $k+1$. A node will vote for the first block proposed by the current-round leader if the block extends the block generated at $locked\_round$ (regardless of round numbers) or the round number of the received block's parent block is greater than $locked\_round$. Concretely, a block satisfies the above condition if and only if: i) its round number is greater than $last\_voted\_round$, and ii) the round number of its parent block is greater than or equal to $locked\_round$.[8]

After voting for a new block, a node will insert the block to its $blockTree$ and then update its state as follows: i)update $last\_voted\_round$ to the round number of this block, and ii) update the node's $locked\_round$ to the round number of this block' grandparent block if the latter is higher. Meanwhile, the node checks whether there are new committed blocks in its $blockTree$. Specifically, if there are three blocks B_k, B_{k+1} and B_{k+2} proposed in three consecutive rounds k, $k+1$, and $k+2$, and an additional block extends block B_{k+2}, the node will **commit** block B_k and all its predecessor blocks. The first three consecutive blocks are referred to as 3-direct chain[9]. A simple case, in which block B_k is committed, is shown in Fig. 2. Note that nodes maintain a chain containing committed blocks, and this chain is referred to as the *main* chain in our later analysis.

To sum up, in pipelined HotStuff, leaders propose new blocks, and every node votes for a new block according to certain voting condition and sometimes commits blocks if there is a 3-direct chain followed by another block. Also, every node starts a timer to track progress for each round. Whenever timeout happens or a block's QC is received, a node moves to the next round. Such a round synchronization procedure is provided by the pacemaker module. In fact, the pacemaker module can also guarantee a new leader to have the highest certified block and/or its parent block, which guarantees the leader can propose a block voted by honest nodes (see Sec. II-B).

B. LibraBFT Algorithm

LibraBFT is a variant of pipelined HotStuff with two subtle differences. *First*, in LibraBFT, a node sends its vote directly to the next-round leader (rather than the current-round leader) so that the next leader can form a QC and embed the QC in its own block. In this way, the current leader doesn't need to relay the QC to the next leader. That is, this optimization can cutoff the delay of the relay operation. *Second*, LibraBFT introduces a new block type called Nil block. This is, when the timer expires, and nodes have not received a proposal for the round, they can broadcast a vote on a Nil block (in a predetermined format). If more than $2n/3$ nodes have voted, the aggregated signatures can serve as a QC for the Nil block. The certified Nil block can guarantee that blocks are produced in consecutive rounds despite having faulty leaders, which can further accelerate the block commitment. For example, in Fig. 2, when the leader of round $k+3$ is faulty, and there is no block produced, nodes cannot commit block B_{k+1} even if receiving block B_{k+4} in pipelined HotStuff. By contrast, in LibraBFT, there will be a certified Nil block at round $k+3$, and nodes will commit block B_{k+1} after receiving block B_{k+4}.

IV. Performance Metrics and Attacks

In this section, we first introduce a multi-metric framework to evaluate the impact of various attacks, and then propose several attack strategies.

A. Performance Metrics

We focus on three performance metrics, namely, chain growth rate, chain quality, and latency. All of these metrics are meaningful only after the GST, which implies that the network is in synchronous mode[10].

[6]For simplicity, we follow the same way with LibraBFT, i.e., leaders extend the predecessor block with a direct child. However, in pipelined HotStuff, a leader has to include dummy blocks in its proposal if there are no certified blocks generated in previous rounds.

[7]Recall that an adversarial leader can propose multiple blocks.

[8]This condition is adopted by LibraBFT and is shown to be equivalent to the previous condition [16], [24].

[9]A 3-direct chain requires an additional block extending 3 consecutive blocks. If we only have 3 consecutive blocks, we don't call them a 3-direct chain.

[10]Before the GST, there may have no certified blocks at all, and so the three metrics become meaningless.

1) Chain Growth Rate: For a given adversarial strategy that controls a fraction α of total nodes, the chain growth rate $u_1(\alpha)$ is defined as the rate of honest blocks appended to the main chain over the long run. Let $B_h(m)$ denote the total number of honest blocks appended to the main chain in m rounds. (Note that $B_h(m)$ can be a random variable because of the randomness in the leader selection.) We have:

$$u_1(\alpha) = \lim_{m\to\infty} \frac{B_h(m)}{m}. \tag{1}$$

The chain growth rate corresponds to the liveness in the context of blockchains.

2) Chain Quality: For a given adversarial strategy that controls a fraction α of total nodes, the chain quality $u_2(\alpha)$ is defined as the fraction of honest blocks included in the main chain over the long run. Let $B_a(m)$ denote the total number of adversarial blocks appended to the main chain in m rounds. We have:

$$u_2(\alpha) = \lim_{m\to\infty} \frac{B_h(m)}{B_h(m) + B_a(m)}. \tag{2}$$

This metric affects the reward distribution. In blockchains, each block in the main chain brings its proposer a reward [1]. This reward incentivizes nodes to participate in the consensus and compete to win the leadership. Additionally, nodes are expected to get rewards, proportional to their devoted resources (e.g., hash power in Poof-of-Work [1], and stakes in Proof-of-Stake [25]). Intuitively, a chain quality less than $1-\alpha$ implies that the adversary can win a higher fraction of rewards than what it deserves, which ruins the incentive compatibility [26].

3) Latency: For a given adversarial strategy that controls a fraction α of total nodes, the latency $u_3(\alpha)$ is defined as the average rounds that honest blocks take from being included in the main chain until being committed over the long run. Let D_i denote the number of rounds that the i-th honest block takes to be committed during the m rounds. We have:

$$u_3(\alpha) = \lim_{m\to\infty} \frac{\sum_{i=1}^{B_h(m)} D_i}{B_h(m)}. \tag{3}$$

Remark 1. *Note that the chain growth rate and latency are measured in terms of* rounds *rather than in time. This round abstraction allows us to ignore the specific implementation of the pacemaker module and focus on the core of pipelined HotStuff.*

B. Attack Strategies

We introduce two attacks here. The forking attack launched by the adversary aims to minimize the chain growth rate and the chain quality by overriding honest blocks. The delay attack aims to maximize the latency by delaying the commitment of honest blocks. Both attacks, which are inspired by the selfish mining attack for Bitcoin, are new in the context of pipelined HotStuff. The optimality of these attacks will be discussed in a journal version of this work. Here, we emphasize that since the performance metrics are measured after the GST, we do not need to consider network-level attacks, such as eclipse attack [27] and Distributed Denial of Service (DDoS) attack, which may cause a network partition (i.e., asynchronous network) and blocking the progress.

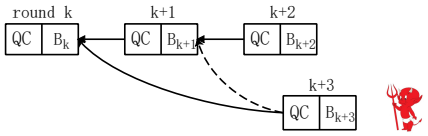


Fig. 3. **The forking attack on pipelined HotStuff.** The adversary is elected as a leader in round $k+3$. It proposes a block B_{k+3} after block B_k (or B_{k+1}) to override blocks B_{k+1} and B_{k+2} (or block B_{k+2}).

1) Forking Attack: In pipelined HotStuff, an adversarial leader can create forking blocks on purpose to override honest blocks without any loss. For example, in Fig. 3, if blocks B_k and B_{k+1} are both honest blocks, and the adversarial leader at round $k+3$ has no adversarial certified block with a round number larger than $locked\_round = k$, the adversarial leader will build a block on block B_k. As block B_{k+3} satisfies the voting condition (i.e., the round number of B_{k+3}'s parent block is no less than honest nodes' $locked\_round = k$), nodes will vote for B_{k+3} and all subsequent leaders will extend B_{k+3}. Similarly, if only block B_{k+2} is an honest block, the adversary can build on block B_{k+1} to override this block. In both cases, the adversarial leader overrides some honest blocks and suffers no loss of adversarial blocks. Also, note that the adversarial leader cannot override block B_k and its predecessor blocks, since block B_{k+3} cannot reference a certified parent block, which has a round number no less than k. In general, if there exist some adversarial certified blocks with round number no less than $locked\_round$, the adversarial leader extends the highest adversarial certified block. Otherwise, the adversarial leader extends the block produced at $locked\_round$.

2) Delay Attack: The main goal of the delay attack is to break the block commitment condition in order to increase the average delay of honest blocks in the main chain. More specifically, the delay attack is to prevent honest blocks to form the 3-direct chain. Recall that a block is committed if and only if three blocks extend it, and the first two blocks are produced in consecutive rounds after the block's round (i.e., the 3-direct chain structure). Note that once the block is committed, all its predecessor blocks are also committed.

Delay Attack in pipelined HotStuff. Recall that an honest leader always proposes a block on the highest certified block. By contrast, an adversarial leader can propose a block on a *non-highest* certified block or propose *no* block at all. An example is provided in Case A of Fig. 4 in which the adversarial leader of round $k+4$ observes three non-consecutive blocks B_k, B_{k+2}, and B_{k+3}. In this case, the leader proposes no block at all (leading to a timeout). As a result, subsequent honest leaders have to restart building a 3-direct chain. Another example is illustrated in Case B of Fig. 4 in which the adversarial leader of round $k+4$ observes three consecutive blocks B_{k+1}, B_{k+2}, and B_{k+3}. In this case, the leader proposes block B_{k+4} on top of B_{k+2}. This will override block B_{k+3} as explained before. In general, if there exist three

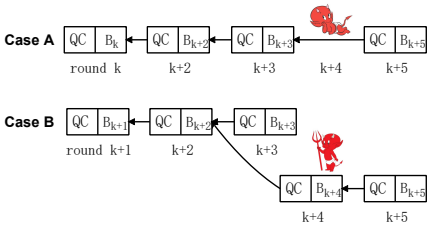


Fig. 4. **The delay attack on pipelined HotStuff.** The adversary proposes no block at all (in Case A) or overrides honest blocks at round $k+4$ (in Case B) according to whether there exists three consecutive blocks.

consecutive blocks that ended with the highest certified block, the subsequent adversarial leader overrides the highest certified block. Otherwise, the adversarial leader proposes no block.

Delay Attack in LibraBFT. Delay attack in LibraBFT is slightly different from that in pipelined HotStuff. First, due to the Nil block, the adversary cannot propose no block, because otherwise a certified Nil block will be produced that can be part of a 3-direct chain. Hence, the adversarial leader can propose multiple different blocks and send them to different honest nodes. In this way, honest nodes will vote for different proposed blocks so that neither a certified block nor a certified Nil block will be produced in this round. Second, in LibraBFT, as nodes send block votes to the next leader, an adversarial leader can hide the collected QC of a block proposed in the previous round. So, when there are three consecutive blocks, the adversarial leader can hide the QC for the last one. In this way, subsequent honest leaders cannot form a 3-direct chain based on the three consecutive blocks.

V. Performance Analysis under Forking and Delay Attacks

In this section, we first analyze the performance of pipelined HotStuff in terms of chain growth rate, chain quality, and latency under forking and delay attacks. Then, we evaluate some optimizations adopted by LibraBFT. Specifically, we consider a sequence of m rounds (when the network is in synchronous mode) and number these rounds as $1, 2, ..., m$. For each round, the possibility that the elected leader is honest (resp, adversarial) is β (resp, α). Now, Let X_j ($j \in [1, m]$) denote an indicator random variable which equals one if the leader of the jth round is honest and equals zero otherwise. As the network is synchronous, nodes can receive a block within Δ time after an honest leader sends the block. For simplicity, nodes are assumed to receive the block by the end of each round. In addition, honest leaders are assumed to be able to get the highest honest certified blocks from other nodes before proposing new blocks.

A. Performance Analysis of Pipelined HotStuff

1) Chain Growth Rate: Recall that an honest block proposed at round i will be overridden by an adversarial leader in round $(i+1)$ or $(i+2)$ under the forking attack (See Sec. IV-B for details.). In other words, an honest block can be kept in the main chain if and only if the subsequent two blocks are honest blocks. Let Z_i denote an indicator random variable, which equals to one if $\{X_i = 1, X_{i+1} = 1, X_{i+2} = 1\}$ and equals to zero otherwise. Next, let $Z = \sum_{i=1}^{m} Z_i$. The following lemma bounds the value of Z.

Lemma 1. *For m consecutive rounds, the number of block fragments $(X_i, X_{i+1}, X_{i+2}) = (1, 1, 1)$ has the following Chernoff-type bound: For $0 < \delta < 1$,*

$$\Pr(|Z - \beta^3 m| > \delta\beta^3 m) < e^{-\Omega\left(\delta^2\beta^3 m\right)}. \tag{4}$$

Proof. Without loss of generality, we assume that m is a multiple of 3. Let $Z^j = \sum_{i=0}^{m/3-1} Z_{j+3i}$ ($j \in [0, 1, 2]$). Then, $Z = Z^0 + Z^1 + Z^2$. It is easy to show that $E\left(Z^j\right) = \beta^3 m/3$, since $P\{Z_i = 1\} = P\{X_i = 1\}P\{X_{i+1} = 1\}P\{X_{i+2} = 1\} = \beta^3$. Note that $\{Z_0, Z_3, \ldots, Z_{m-1}\}$ are independent random variables, because Z_i is a function of (X_i, X_{i+1}, X_{i+2}). Hence, Z^j is a sum of i.i.d. random variables. By Lemma 4 in [28], we have

$$\Pr\left(Z < (1-\delta)\beta^3 m\right) < e^{-\Omega\left(\delta^2\beta^3 m\right)}.$$

Similarly, we have $\Pr\left(Z > (1+\delta)\beta^3 m\right) < e^{-\Omega\left(\delta^2\beta^3 m\right)}$. □

This lemma shows that as m increases, the number of block fragments $(X_i, X_{i+1}, X_{i+2}) = (1, 1, 1)$ is between $(1-\delta)\beta^3 m$ and $(1+\delta)\beta^3 m$ with high probability. Moreover, each block fragment corresponds to one honest block included in the main chain. This leads to the following theorem for the chain growth rate.

Theorem 1. *The chain growth rate of pipelined HotStuff under the forking attack converges to β^3 with high probability as $m \to \infty$.*

Proof. By Lemma 1 and the definition of $u_1(\alpha)$, we have $u_1(\alpha) = \lim_{m\to\infty} \sum_{i=0}^{m-1} Z_i/m \to \beta^3$. □

Note that when there does not exist the forking attack, the chain growth rate is β, for the probability that an honest node is elected as a leader is β. In other words, this theorem states that the forking attack reduces the chain growth rate from β to β^3. For instance, if $\beta = 2/3$, the chain growth rate is reduced from $2/3$ to $8/27$.

Remark 2. *The chain growth rate is measured in terms of rounds. If it is measured in time, the forking attack can reduce the chain growth rate even more. This is because an adversarial leader can push the duration of its round close to the timeout value, which is usually much longer than the actual network delay for producing a certified block.*

2) Chain Quality: Recall that the adversary suffers no loss of blocks when launching the forking attack. That is, every adversarial block can be kept in the main chain. Therefore, the adversary can produce αm adversarial blocks on expectation over m rounds. This observation, together with Theorem 1, allows us to derive the following chain quality theorem for pipelined HotStuff.

Theorem 2. *The chain quality of pipelined HotStuff under the forking attack converges to $\frac{\beta^3}{\beta^3-\beta+1}$ with high probability as $m \to \infty$.*

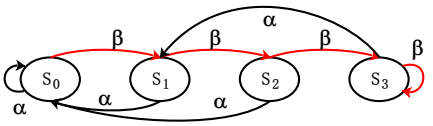


Fig. 5. **The state transition of the delay attack on pipelined HotStuff.**

Proof. As $m \to \infty$, $\frac{B_a(m)}{m}$ (the number of adversarial blocks divided by m) converges to α by Lemma 3 in [28], and $\frac{B_h(m)}{m}$ (the number of honest blocks in the main chain divided by m) converges to β^3 by Lemma 1. Hence, the chain quality converges to $\frac{\beta^3}{\beta^3-\beta+1}$. Note that $\alpha = 1-\beta$. □

For example, if $\alpha = 1/3$, the chain quality under the forking attack is $8/17$, whereas it should be $2/3$ without the forking attack. If each block can bring its owner a reward, the adversary can obtain a fraction of rewards $\frac{\alpha}{\beta^3-\beta+1}$ by launching forking attack, which is always higher than the deserved fraction α, for $\beta^3 - \beta + 1 < 1$. In other words, incentive compatibility of pipelined HotStuff cannot hold anymore under the forking attack.

3) Latency: Here, we compute the latency of honest blocks in the main chain. To achieve this goal, we need to track each honest block included in the main chain and obtain its associated delay to be committed. More precisely, the chance that an honest block is kept in the main chain and its delay is affected by the delay attack strategies, which further depend on the chain structure. For example, in Fig. 4, as shown in Case A, as there is no 3-chain structure of the latest blocks, the adversary proposes no block and honest blocks B_{k+2} and B_{k+3} are kept in the main chain; however, they will be overridden in Case B. Therefore, we need to track the previous block structure of every honest block. To this end, we define four states as follows:

- S_0: the state where the previous round is a timeout and no certified block is produced;
- S_i for $i \in \{1, 2, 3\}$: the state where there exists i consecutive blocks that are not committed yet.

Under the delay attack described in Sec IV-B, we can develop a Markov model of state transitions in Fig. 5. Recall that α (respectively, β) is the probability that an honest (respectively, adversarial) leader proposes a new block. Each transition denotes a new round and there exists a designated leader. An honest leader always proposes a new block that extends the highest certified block (denoted as the red line in Fig. 5). This Markov model allows us to track the previous chain structure for any new honest block as well as to obtain its chance to be included in the main chain and the associated delay. We have the following theorem on the delay of pipelined HotStuff.

Theorem 3. *The latency of pipelined HotStuff under the delay attack converges to* $\frac{\beta^7+3\beta^6-4\beta^5+2\beta^4+\beta^3-2\beta^2+\beta+1}{2\beta^7-2\beta^6+\beta^4}$ *with high probability as* $m \to \infty$.

Proof. First, by solving the above Markov model, we can obtain the steady-state distribution of each state as follows:

$$\pi_0 = \frac{(1+\beta)(1-\beta)^2}{\beta^3-\beta^2+1}, \quad \pi_1 = \frac{\beta(1-\beta)}{\beta^3-\beta^2+1},$$
$$\pi_2 = \frac{\beta^2(1-\beta)}{\beta^3-\beta^2+1}, \quad \pi_3 = \frac{\beta^3}{\beta^3-\beta^2+1}.$$

Next, we can analyze the latency of honest blocks in each state transition. In particular, we focus on the blocks that eventually end up in the main chain.

- *Case a:* $S_0 \xrightarrow{\beta} S_1$. All proposed honest blocks will be kept in the main chain, and their average delay is $\frac{\beta^3+\beta+1}{\beta^4}$.
- *Case b:* $S_1 \xrightarrow{\beta} S_2$. The honest blocks have α probability to be committed with an average delay $\frac{\beta^4+2\beta^3+\beta+1}{\beta^4}$, $\alpha\beta$ probability to be committed with an average delay $\frac{2\beta^4+\beta^3+\beta+1}{\beta^4}$, and β^2 probability to be committed with an average delay $\frac{2\beta^4+\beta^3-\beta^2+1}{\beta^4}$.
- *Case c:* $S_2 \xrightarrow{\beta} S_3$ *and* $S_3 \xrightarrow{\beta} S_4$. The honest blocks have β^2 probability to be committed with an average delay $\frac{2\beta^4+\beta^3-\beta^2+1}{\beta^4}$ and $\alpha\beta$ probability to be committed with an average delay $\frac{2\beta^4+\beta^3+\beta+1}{\beta^4}$.

Due to space constraint, the detailed proofs of these cases are provided in Appendix B1 of our technical report [28]. With these results, it is easy to obtain the latency as:

$$u_3(\alpha) = \frac{\beta^7+3\beta^6-4\beta^5+2\beta^4+\beta^3-2\beta^2+\beta+1}{2\beta^7-2\beta^6+\beta^4}. \quad (5)$$

This completes the proof. □

The theorem shows that when the adversary controls $1/3$ of Byzantine nodes (i.e., $\alpha = 1/3$ and $\beta = 2/3$), the average latency for committing one block under the delay attack is about 8.33 rounds. By contrast, without the delay attack, a block is committed if the next three consecutive blocks extend it with a latency of 3 rounds.

B. Performance Analysis of LibraBFT

We analyze the performance of LibraBFT under the attack strategies. In particular, we will evaluate the differences between pipelined HotStuff and LibraBFT. These differences made by LibraBFT aim to cut off the delay of relaying blocks' QCs or fasten the block commitment (see Sec. III).

On the one hand, as the forking attack strategies are the same, LibraBFT has the same chain growth rate and chain quality as pipelined HotStuff. In other words, these changes do not affect these two metrics. On the other hand, we can develop a similar Markov model to analyze the delay attack in LibraBFT as shown in Fig. 6. This allows us to obtain the following theorem on the latency for LibraBFT.

Theorem 4. *The latency of LibraBFT under the delay attack converges to* $\frac{\beta^7+\beta+1}{\beta^7-\beta^6+\beta^4}$ *with high probability as* $m \to \infty$.

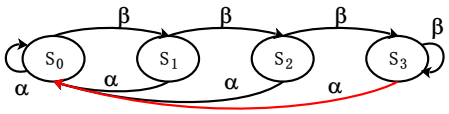


Fig. 6. **The state transition of the delay attack on LibraBFT.** The red line denotes a different action adopted by the adversarial leader in LibraBFT.

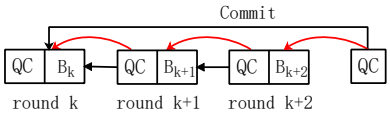


Fig. 7. **The committing rule with broadcasting QCs in Pipelined HotStuff.**

Proof. First, by solving the above Markov model, we can obtain the steady-state distribution of each state as follows:

$$\pi_0 = 1-\beta, \quad \pi_1 = \beta(1-\beta), \quad \pi_2 = \beta^2(1-\beta), \quad \pi_3 = \beta^3.$$

Next, we can analyze the latency of honest blocks in each state transition. We detail honest blocks on each event below.

- *Case a:* $S_0 \xrightarrow{\beta} S_1$ *and* $S_1 \xrightarrow{\beta} S_2$. All honest blocks produced after a previous timeout round or just one consecutive block will be kept in the main chain, and their average delay is $\frac{\beta^2+\beta+1}{\beta^4}$.
- *Case b:* $S_2 \xrightarrow{\beta} S_3$ *and* $S_3 \xrightarrow{\beta} S_4$. These honest blocks have β probability to be committed with an average delay $\frac{\beta^4+\beta+1}{\beta^4}$.

Due to space constraint, the detailed proofs of these cases are provided in Appendix B2 of our technical report [28]. With these results, it is easy to obtain the latency as:

$$u_3(\alpha) = \frac{\beta^7+\beta+1}{\beta^7-\beta^6+\beta^4}. \tag{6}$$

This completes the proof. □

The theorem shows that when $\beta = 2/3$, the average latency for committing one block under the delay attack is about 10.25 rounds. Compared with the delay in pipelined HotStuff, it suggests that the mechanism of sending votes to the next-round leader makes the system more vulnerable against the delay attack.

VI. Countermeasures

A. Broadcasting QCs

The first countermeasure is that current-round leaders broadcast QCs to all nodes (rather than just relaying QCs to next-round leaders)[11]. Broadcasting QCs can provide two benefits. First, when nodes receive QCs, they can update their $locked\_round$, which can effectively thwart the forking attack. For example, in Fig. 7, when honest nodes receive QC for block B_{k+2}, they can update $locked\_round$ from k to $k+1$. As a result, the adversarial leader of round $k+3$ can only override block B_{k+2}; however, without this mechanism, the adversarial leader can override both blocks B_{k+1} and B_{k+2}. Second, broadcasting QCs can fasten block commitments. Specifically, when nodes observe a 3-direct chain of B_k, B_{k+1}, and B_{k+2}, as well as block B_{k+2}'s QC, they can commit block B_k, as shown in Fig. 7. By contrast, in pipelined HotStuff, without broadcasting QCs, nodes need to wait until the subsequent block carrying block B_{k+2}'s QC is received and then commit block B_k. As said in Sec. V-A3, this enables an adversarial leader at round $k+3$ to hide B_{k+2}'s QC and ruin the 3-direct chain to increase block delay. Additionally, even in the ideal case, broadcasting QC can accelerate the block commitment; the latency for committing one block is reduced to two rounds. In the following, we provide a formal analysis of the improvement in chain growth rate, chain quality, and delay.

1) Chain Growth Rate and Chain Quality: With broadcasting QCs, an honest block proposed at round i can only be overridden by an adversarial leader at round $(i+1)$. In other words, an honest block can be kept in the main chain if and only if the subsequent leader is an honest leader. Let Y_i denote an indicator random variable, which equals to one if $\{X_i = 1, X_{i+1} = 1\}$ and equals to zero otherwise. Next, let $Y = \sum_{i=1}^{m} Y_i$. By following our previous analysis, we can bound the value of Y:

Lemma 2. *For m consecutive rounds, the number of block fragments $(X_i, X_{i+1}) = (1,1)$ has the following Chernoff-type bound: For $0 < \delta < 1$,*

$$\Pr(|Y - \beta^2 m| > \delta\beta^2 m) < e^{-\Omega(\delta^2\beta^2 m)}. \tag{7}$$

Proof. The analysis is very similar to that for Lemma 1. □

With this lemma, we can easily derive the following theorems on chain growth rate and chain quality, respectively.

Theorem 5. *The chain growth rate of pipelined HotStuff with broadcasting QCs under the forking attack converges to β^2 with high probability as $m \to \infty$.*

Theorem 6. *The chain quality of pipelined HotStuff with broadcasting QCs under the forking attack converges to $\frac{\beta^2}{\beta^2-\beta+1}$ with high probability as $m \to \infty$.*

These two theorems can be easily proved by following our previous analysis for pipelined HotStuff. Due to space constraints, we do not provide the proofs here. When $\beta = 2/3$, the chain growth rate improves by 1.5x, while the chain quality improves by 1.2x.

2) Latency: Following our previous analysis, we can develop a Markov model of the delay attack in Fig. 8. Note that as the adversarial leader always chooses to propose no blocks, all honest blocks can be kept in the main chain. Moreover, by solving the above Markov model, we have the following theorem.

Theorem 7. *The latency of pipelined HotStuff with broadcasting QCs under the delay attack converges to $\frac{\beta+1}{\beta^3}$ with high probability as $m \to \infty$.*

[11] An earlier version of LibraBFT has adopted this mechanism.

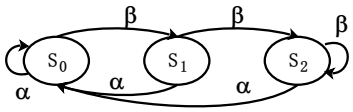


Fig. 8. **The state transition of the delay attack on pipelined HotStuff with broadcasting QCs.**

Proof. All produced honest blocks will be kept in the main chain, and their average delay is $\frac{\beta+1}{\beta^3}$. Due to space constraint, the detailed proofs of these cases are provided in Appendix B3 of our technical report [28]. □

This theorem shows that when $\beta = 2/3$, the average latency for committing one block under the delay attack is 5.63 rounds. It suggests that broadcasting QCs can reduce the average block latency by almost 3 rounds. Note that broadcasting QCs also brings additional delay. Therefore, it is a design tradeoff, which should be evaluated in real settings in order to decide whether to adopt it.

B. Longest Chain Rule

Our second countermeasure is changing the block proposing rule. In pipelined HotStuff, an honest node always extends the highest certified block. This deterministic block proposing rule enables the adversary to override at most two previous honest blocks by a higher certified block without any loss (i.e., decreasing chain growth rate and chain quality) and to break the 3-direct chain (i.e., increasing latency). Therefore, we suggest that nodes can choose to extend the longest certified block chain[12]. In particular, when there are two forking branches with the same length, they randomly choose one to extend. This randomized block proposing rule is inspired by the longest chain rule in NC. The detailed analysis of this countermeasure will be provided in a journal version of this work.

VII. Evaluation

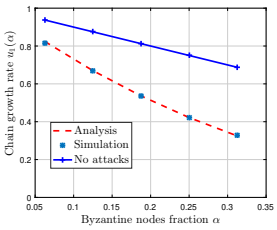
We implement a proof-of-concept of pipelined HotStuff to evaluate its performance in terms of chain growth rate, chain quality, and latency under the forking and delay attacks.

A. Testnet Setup

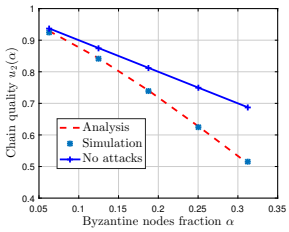
We consider a system of 16 nodes, and the number of Byzantine nodes is up to 5. For simplicity, nodes are set to have synchronized clocks, and so the protocol proceeds in synchronized rounds[13]. We build a full-fledged implementation of pipelined HotStuff using Golang (around 3,600 LoC). We run the simulation on a late 2013 Apple MacBook Pro, 2.7GHz Intel Core i7. In our experiments, the adversary runs the attack strategies in Sec. IV-B. Our simulation results are based on an average of 10 runs, where each run generates 100,000 blocks.

[12]In LibraBFT, it sometimes says that nodes extend the longest chain. This is because in the ideal case, the highest certified block is the head of the longest chain.

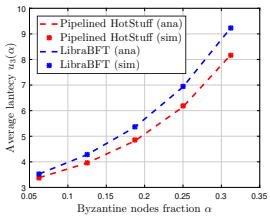
[13]In a partially synchronous network, nodes can establish a synchronized clock as long as they have clocks with bounded drift [19].



(a) The chain growth rate under the forking attack.



(b) The chain quality under the forking attack.



(c) The average latency of honest blocks under the delay attack.

Fig. 9. The performance of pipelined HotStuff as well as LibraBFT under the forking and delay attacks.

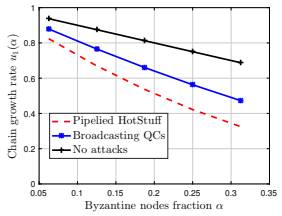
B. Pipelined HotStuff and LibraBFT

We evaluate the performance of pipelined HotStuff as well as LibraBFT through extensive experiments. Since LibraBFT has the same chain growth rate and chain quality as pipelined HotStuff, these two performance metrics are only given for pipelined HotStuff.

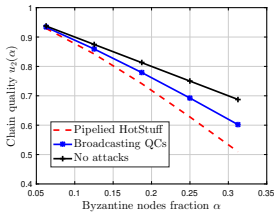
1) Chain Growth Rate: Fig. 9(a) shows the chain growth rate of pipelined HotStuff with different fractions of Byzantine nodes. First, we observe that the simulation results well match the analysis results. Second, the results show that as the fraction α of Byzantine nodes increases, the gap between the chain growth rates with and without the forking attack also increases. When α is close to 0.3, the chain growth rate under the attack can drops to almost half of that without attacks.

2) Chain Quality: Fig. 9(b) shows the chain quality of pipelined HotStuff with different fractions of Byzantine nodes. First, the evaluation results match our previous analysis. Second, The results show that by the forking attack, the adversary can lower the chain quality and obtain a higher fraction of blocks in the main chain than what it deserves. If each block in the main chain brings to its owner a reward, this implies that the adversary can always gain a higher fraction of rewards. In other words, the incentive compatibility cannot be held anymore under the attack. For example, when α is close to 0.3, the chain quality drops to 0.51. This implies that 0.3 of Byzantine nodes can produce almost half of the blocks in the main chain (and obtain half of the rewards).

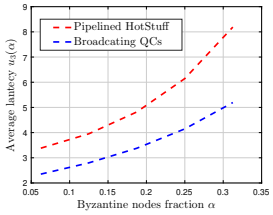
3) Latency: Fig. 9(c) shows the average latency of honest blocks in the main chain in pipelined Hotstuff and LibraBFT. First, the evaluation results, once again, validate our analysis. Second, the results show that as the fraction α of Byzantine nodes increases, both the latency in pipelined Hotstuff and



(a) The chain growth rate under the forking attack.



(b) The chain quality under the forking attack.



(c) The average latency under the delay attack.

Fig. 10. The performance of pipelined HotStuff under the forking and delay attacks with and without broadcasting QCs.

LibraBFT increase. In addition, the latency in LibraBFT is larger than that in pipelined HotStuff, which implies that the engineering optimizations adopted by LibraBFT may make it more vulnerable to the delay attack. Note that in both pipelined HotStuff and LibraBFT, the latency for committing one block is three rounds without attacks. Therefore, when α is close to 0.3, the average latency of honest blocks under the delay attacks is almost 3x of that without attacks.

C. Countermeasures

We evaluate the performance of pipelined HotStuff with broadcasting QCs. Fig. 10(a) and 10(b) show that as the fraction α of Byzantine nodes increases, the gap between the chain growth rates (and chain qualities) of pipelined HotStuff with and without broadcasting QCs also increases. This implies that the higher α is, the higher performance improvement that broadcasting QCs brings. Fig. 10(c) shows that the latency of pipelined HotStuff with broadcasting QCs is at least one round shorter than the original HotStuff. In addition, when α is close to 0.3, the average latency of honest blocks can drop by almost 3 rounds. As previously explained, broadcasting QCs also brings additional delay. Therefore, it is a design tradeoff that should be evaluated in *real settings*. Finally, we would like to point out that the longest chain rule can also significantly enhance the performance of pipelined HotStuff. Moreover, the longest chain rule brings no additional overhead, and it can be combined with broadcasting QCs. We will present these results in a journal version of this work.

VIII. Related Work

Reaching consensus in face of Byzantine failures was formulated as the Byzantine agreement problem by Lamport *et al*. [29], and has been studied for several decades. Various BFT consensus protocols such as PBFT [7], Zyzzyva [30], and Q/U [10] have been proposed. However, these classical BFT protocols suffer from poor scalability, notorious complexity and leader fairness issues (see Sec. I) and are hard to be used in large-scale blockchains. To address these issues, several state-of-the-art BFT protocols [14], [15], [31], [32] are proposed for building large-scale blockchains.

Tendermint. Tendermint [31] features a continuous leader rotation (also called the democracy-favoring leader rotation [14]) based on PBFT protocol. Specifically, Tendermint embeds round-change mechanism into the common-case pattern, and the leader is re-elected from all the nodes by some desired policy after every block, resulting in better leadership fairness.

Casper FFG. Buterin and Griffith [32] proposed a protocol called Casper FFG, which works as an overlay atop NC to provide "finality gadget". Casper FFG applies an elegant pipelining idea to the classical BFT protocol, i.e., if each block required two rounds of voting, one can piggyback the second round on the next block's voting. This pipelining idea enables the system to have one identical round (rather than multiple rounds with different functionalities and names[14]), and so significantly simplifies the protocol design.

Pala and Streamlet. Pala [14] is a simple BFT consensus protocol that also adopts the pipelining idea. However, for high throughput, it uses a stability-favoring leader rotation policy. Based on this work, Chan *et al*. [33] proposed Streamlet, which further simplifies the voting rule. Streamlet aims to provide a unified, simple protocol for both teaching and implementation.

HotStuff. HotStuff proposed by Yin *et al*. [15] creatively adopts a three-phase commit rule (rather than two-phase commit rule used in Casper FFG, Pala, and Streamlet) to enable the protocol to reach consensus at the pace of actual network delay. In addition, HotStuff adopts the threshold signature to realize linear message complexity, and can also be pipelined into a practical protocol for building large-scale blockchains.

IX. Conclusion

The state-of-the-art pipelined HotStuff not only provides linear message complexity and responsiveness, but also is efficient for building large-scale blockchains. Thus, pipelined HotStuff has been adopted in many blockchain projects such as Libra, Flow, and Cypherium. In this paper, we propose a multi-metric evaluation framework including chain growth rate, chain quality, and latency. We also propose two attacks, namely the forking attack and delay attack, and systematically study the impacts of these two attacks on the performance of pipelined HotStuff. Also, we leverage the framework to evaluate some engineering designs in LibraBFT. Finally, we propose some countermeasures to enhance the performance of pipelined HotStuff against these attacks. We hope that our framework can contribute to proposing new variants of HotStuff as well as making HotStuff more understandable for developers and practitioners in terms of performance.

[14]In PBFT, for committing one proposal, there are two phases: prepare and commit phases, and each phase has different functionalities.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Working Paper*, 2008.

[2] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015*. Berlin Heidelberg: Springer, 2015, pp. 281–310.

[3] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Advances in Cryptology – EUROCRYPT 2017*. Cham: Springer, 2017, pp. 643–673.

[4] L. Ren, "Analysis of Nakamoto consensus," Cryptology ePrint Archive, Report 2019/943, 2019.

[5] J. Niu, C. Feng, H. Dau, Y.-C. Huang, and J. Zhu, "Analysis of Nakamoto consensus, revisited," *arXiv preprint arXiv:1910.08510*, 2019.

[6] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, p. 228–234, Apr. 1980.

[7] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186.

[8] A. Bessani, J. Sousa, and E. E. P. Alchieri, "State machine replication for the masses with BFT-SMART," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 355–362.

[9] J. Sousa and A. Bessani, "Separating the wheat from the chaff: An empirical design for geo-replicated state machines," in *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*, 2015, pp. 146–155.

[10] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-scalable Byzantine fault-tolerant services," *SIGOPS Oper. Syst. Rev.*, vol. 39, no. 5, p. 59–74, Oct. 2005.

[11] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 BFT protocols," in *Proceedings of the 5th European Conference on Computer Systems*, ser. EuroSys '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 363–376.

[12] J. Mickens, "The saddest moment," *Login Usenix Mag*, vol. 39, no. 3, pp. 52–54, 2014.

[13] M. Vukolić, "The quest for scalable blockchain Fabric: Proof-of-Work vs. BFT replication," in *Open Problems in Network Security*. Cham: Springer International Publishing, 2016, pp. 112–125.

[14] T.-H. H. Chan, R. Pass, and E. Shi, "Pala: A simple partially synchronous blockchain." *IACR Cryptology ePrint Archive*, vol. 2018, p. 981, 2018.

[15] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT consensus with linearity and responsiveness," ser. PODC '19. New York, NY, USA: ACM, 2019, pp. 347–356.

[16] S. Bano, M. Baudet, A. Ching, A. Chursin, G. Danezis, F. Garillot, Z. Li, D. Malkhi, O. Naor, D. Perelman *et al.*, "State machine replication in the Libra blockchain," May 2020. [Online]. Available: https://developers.libra.org/docs/state-machine-replication-paper.

[17] A. Hentschel, Y. Hassanzadeh-Nazarabadi, R. M. Seraj, D. Shirley, and L. Lafrance, "Flow: Separating consensus and compute - block formation and execution," *ArXiv*, vol. abs/2002.07403, 2020.

[18] Y. Guo, Q. Yang, H. Zhou, W. Lu, and S. Zeng, "Syetem and methods for selection and utilizing a committee of validator nodes in a distributed system," Cypherium Blockchain, Feb 2020, patent. [Online]. Available: https://github.com/cypherium/patent

[19] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 1988.

[20] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, Jun. 2014, pp. 305–319.

[21] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems," in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIXATC'10. USA: USENIX Association, 2010, p. 11.

[22] A. Spiegelman and A. Rinberg, "ACE: Abstract consensus encapsulation for liveness boosting of state machine replication," *arXiv preprint arXiv:1911.10486*, 2019.

[23] I. Abraham, D. Malkhi, and A. Spiegelman, "Asymptotically optimal validated asynchronous Byzantine agreement," ser. PODC '19, New York, NY, USA, 2019, p. 337–346.

[24] S. Bano, A. Sonnino, A. Chursin, D. Perelman, and D. Malkhi, "Twins: White-glove approach for BFT testing," *arXiv preprint arXiv:2004.10617*, 2020.

[25] 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*, ser. SOSP '17. New York, NY, USA: ACM, 2017, pp. 51–68.

[26] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, ser. PODC '17. New York, NY, USA: ACM, 2017, pp. 315–324.

[27] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on Bitcoin's peer-to-peer network," in *Proceedings of the 24th USENIX Conference on Security Symposium*. USA: USENIX Association, 2015, p. 129–144.

[28] Anonymous Authors, "On the performance of pipelined hotstuff." [Online]. Available: https://github.com/infocom2021HotStuffReport/Report

[29] L. Lamport, R. E. Shostak, and M. C. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[30] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative Byzantine fault tolerance," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 45–58, 2007.

[31] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains." M. Eng. thesis, The University of Guelph, Ontario, Canada, Jun. 2016.

[32] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *CoRR*, vol. abs/1710.09437, 2017.

[33] B. Y. Chan and E. Shi, "Streamlet: Textbook streamlined blockchains," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 88, 2020.

A. Concentration Bounds

We denote the probability of an event E by $\Pr[E]$ and the expected value of a random variable X by $\mathbb{E}[X]$. We will use the following bounds in our proofs.

Lemma 3 (Chernoff bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables, and let $\mu := \mathbb{E}[\sum_{i=1}^{n} X_i]$. For $0 < \delta < 1$, we have $\Pr[\sum_{i=1}^{n} X_i > (1+\delta)\mu] < e^{-\Omega(\delta^2\mu)}$ and $\Pr[\sum_{i=1}^{n} X_i < (1-\delta)\mu] < e^{-\Omega(\delta^2\mu)}$*

Lemma 4 (Chernoff bound for dependent random variables [5])**.** *Let T be a positive integer. Let $X^{(j)} = \sum_{i=0}^{n-1} X_{j+iT}$ be the sum of n independent indicator random variables and $\mu_j = \mathbb{E}\left[X^{(j)}\right]$ for $j \in \{1, \ldots, T\}$. Let $X = X^{(1)} + \cdots + X^{(T)}$. Let $\mu = \min_j\{\mu_j\}$. Then, for $0 < \delta < 1$, $\Pr[X \leq (1-\delta)\mu T] \leq e^{-\Omega(-\delta^2\mu/2)}$.*

B. Latency Analysis

1) Pipelined HotStuff: Based on the defined states in Sec.V-A3, we introduce an additional state S_x, where the highest certified block together with its predecessor blocks form a 3-chain structure. By the aforementioned commit rule, when nodes observer a 3-chain structure, the first block in the 3-chain structure together with its predecessor blocks are all committed (see Sec. III-A). This further means when an honest block is kept in the main chain, and its descendant blocks first form a 3-chain structure, it will be committed.

By slightly modified the Markov model in Fig. 5, we can show the state transitions for a node to observe some new committed blocks (i.e., entering state S_x) in Fig. 11. Furthermore, we can compute the expected rounds for a state S_i ($i \in \{0, 1, 2, 3\}$) to first enter state S_x. To realize this, we introduce a new variable X_i, which denote the number of rounds that starting from state S_i, the state first hits state

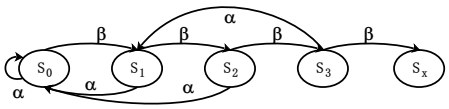


Fig. 11. **The state transition of the delay attack on pipelined HotStuff.**

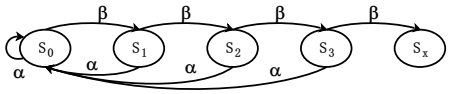


Fig. 12. **The state transition of the delay attack on pipelined HotStuff.**

S_x. In addition, we use $\mathbb{E}[X_i]$ to denote the expectation of X_i. With the state transitions, we can have the following equation.

$$\begin{aligned}
\mathbb{E}[X_0] &= \alpha\mathbb{E}[X_0] + \beta\mathbb{E}[X_1] + 1 \\
\mathbb{E}[X_1] &= \alpha\mathbb{E}[X_0] + \beta\mathbb{E}[X_2] + 1 \\
\mathbb{E}[X_2] &= \alpha\mathbb{E}[X_0] + \beta\mathbb{E}[X_3] + 1 \\
\mathbb{E}[X_3] &= \alpha\mathbb{E}[X_1] + 1
\end{aligned} \tag{8}$$

By solving the equation (8), we can get:

$$\mathbb{E}[X_0] = \frac{2\beta^3+\beta+1}{\beta^4}, \quad \mathbb{E}[X_1] = \frac{\beta^3+\beta+1}{\beta^4},$$
$$\mathbb{E}[X_2] = \frac{(1+\beta)(\beta^2-\beta+1)}{\beta^4}, \quad \mathbb{E}[X_3] = \frac{\beta^3-\beta^2+1}{\beta^4}.$$

Next, we can track honest blocks kept in the main chain and their associated delay. In addition, we refer to these tracked honest block as target blocks. As we said previously, the chance that an honest block is kept in the main chain and its delay is affected by the delay attack strategies, which further depend on the chain structure. Hence, according to its predecessor blocks structure, we can divide honest blocks into the following three cases:

- *Case a:* $S_0 \xrightarrow{\beta} S_1$. This state transition denotes that a target block is proposed after a previous timeout round. By the delay attack strategies in Sec. IV-B2, the honest block will always be kept in the main chain. Further, as the current system state is S_1, the delay for the target block to be committed is $\mathbb{E}[X_1]$.
- *Case b:* $S_1 \xrightarrow{\beta} S_2$. This state transition denotes that a target block together with its parent block can only form two consecutive blocks. According to the subsequent leaders (and blocks), the delay for this target block can be divided into three subcases:
  Subcase 1: The next leader is adversarial and proposes no block. This subcase happens with probability α. By then, the system state is S_0, and the average delay for this target block to be committed is $\mathbb{E}[X_0] + 1$.
  Subcase 2: Some honest leader propose the next block, and then this block is overridden by a block of the subsequent adversarial leader. This subcase happens with probability $\beta\alpha$. By then, as the state is S_1, the average delay for this target block to be committed is $\mathbb{E}[X_1] + 2$.
  Subcase 3: Some honest leader propose the next two blocks. This subcase happens with probability β^2. As the state is S_3, the average delay for this target block to be committed is $\mathbb{E}[X_3] + 2$.
- *Case c:* $S_2 \xrightarrow{\beta} S_3$ *and* $S_3 \xrightarrow{\beta} S_4$. This state transition denotes that a target block together with its predecessor blocks can form three consecutive blocks. According to the subsequent leaders (and blocks), the delay for this target block can be divided into two subcases:
  Subcase 1: Some honest leader propose a block, and then this new block is overridden by the next adversarial block. This subcase happens with probability $\beta\alpha$. As the system state is S_1, the average delay for this target block to be committed is $\mathbb{E}[X_1] + 2$.
  Subcase 2: Some honest leader propose two consecutive block. This subcase happens with probability β^2. As the system state is S_3, the average delay for this target block to be committed is $\mathbb{E}[X_3] + 2$.

2) LibraBFT: By following a similar analysis of pipelined HotStuff, we first can show the state transitions for the system to enter the next state S_x in Fig. 12. Furthermore, we can compute the expected rounds for a state S_i ($i \in \{0,1,2,3\}$) to first enter state S_x and have the following equation.

$$\begin{aligned}
\mathbb{E}[X_0] &= \alpha\mathbb{E}[X_0] + \beta\mathbb{E}[X_1] + 1 \\
\mathbb{E}[X_1] &= \alpha\mathbb{E}[X_0] + \beta\mathbb{E}[X_2] + 1 \\
\mathbb{E}[X_2] &= \alpha\mathbb{E}[X_0] + \beta\mathbb{E}[X_3] + 1 \\
\mathbb{E}[X_3] &= \alpha\mathbb{E}[X_0] + 1
\end{aligned} \tag{9}$$

By solving the equation (9), we can get:

$$\mathbb{E}[X_0] = \frac{\beta^3+\beta^2+\beta+1}{\beta^4}, \quad \mathbb{E}[X_1] = \frac{\beta^2+\beta+1}{\beta^4},$$
$$\mathbb{E}[X_2] = \frac{\beta+1}{\beta^4}, \quad \mathbb{E}[X_3] = \frac{1}{\beta^4}$$

Next, we can track honest blocks kept in the main chain and their associated delay. According to its predecessor blocks structure, we can divide honest blocks into two cases as followings.

- *Case a:* $S_0 \xrightarrow{\beta} S_1$ *and* $S_1 \xrightarrow{\beta} S_2$. By the delay attack strategies in Sec. IV-B2, the honest block will always be kept in the main chain. Further, as the current system state is S_1, the delay for the honest block to be committed is $\mathbb{E}[X_1]$.
- *Case b:* $S_2 \xrightarrow{\beta} S_3$ *and* $S_3 \xrightarrow{\beta} S_4$. This state transition denotes that an honest block together with its predecessor blocks can only form three consecutive blocks. According to the subsequent leaders (and blocks), the delay for this target block can be divided into two subcases:
  Subcase 1: Some honest leader propose a block, and then the subsequent adversarial leader hides the QC and proposes no block. This subcase happens with probability $\beta\alpha$. As the system state is S_0, the average delay for this target block to be committed is $\mathbb{E}[X_0] + 2$.

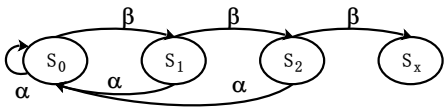


Fig. 13. **The state transition of the delay attack on pipelined HotStuff.**

Subcase 2: Some honest leader propose two consecutive block. This subcase happens with probability β^2. As the system state is S_3, the average delay for this target block to be committed is $\mathbb{E}[X_3] + 2$.

3) Broadcasting QC: By following the previous analysis, we first can show the state transitions for the system to enter the next state S_x in Fig. 13. Furthermore, we can compute the expected rounds for a state S_i ($i \in \{0, 1, 2, 3\}$) to first enter state S_x, and have the following equation.

$$\begin{aligned} \mathbb{E}[X_0] &= \alpha\mathbb{E}[X_0] + \beta\mathbb{E}[X_1] + 1 \\ \mathbb{E}[X_1] &= \alpha\mathbb{E}[X_0] + \beta\mathbb{E}[X_2] + 1 \\ \mathbb{E}[X_2] &= \alpha\mathbb{E}[X_0] + 1 \end{aligned} \tag{10}$$

By solving the equation (10), we can get:

$$\mathbb{E}[X_0] = \frac{\beta^2 + \beta + 1}{\beta^3}, \quad \mathbb{E}[X_1] = \frac{\beta + 1}{\beta^3}, \quad \mathbb{E}[X_2] = \frac{1}{\beta^3}.$$

Next, we can track honest blocks kept in the main chain and their associated delay. Note that all honest blocks will be kept in the main chain by the delay attack strategies in pipelined HotStuff with broadcasting QCs. Furthermore, as the system state is S_1, the delay for the honest block to be committed is $\mathbb{E}[X_1]$.