Unraveling Responsiveness of Chained BFT Consensus with Network Delay

Anonymous Author(s)

***Abstract*—With the advancement of blockchain technology, chained Byzantine Fault Tolerant (BFT) protocols have been increasingly adopted in practical systems, making their performance a crucial aspect of the study. In this paper, we introduce a unified framework utilizing Markov Decision Processes (MDP) to model and assess the performance of three prominent chained BFT protocols. Our framework effectively captures complex adversarial behaviors, focusing on two key performance metrics: chain growth and commitment rate. We implement the optimal adversarial strategies obtained from MDP analysis on an existing evaluation platform for chained BFT protocols and conduct extensive experiments under various settings to validate our theoretical results. Through rigorous theoretical analysis and thorough practical experiments, we provide an in-depth evaluation of chained BFT protocols under diverse attack scenarios, uncovering optimal attack strategies. Contrary to conventional belief, our findings reveal that while responsiveness can enhance performance, it is not universally beneficial across all scenarios. This work not only deepens our understanding of chained BFT protocols, but also offers valuable insights and analytical tools that can inform the design of more robust and efficient protocols.**

***Index Terms*—Chained BFT, Responsiveness, MDP, Attack strategy, Performance metrics.**

I. Introduction

The growing popularity of decentralized applications, including global payments [1], [2], DeFi [3], and online gaming [4], has revitalized interest in Byzantine Fault Tolerant (BFT) consensus. Recently, a family of chained BFT protocols utilizing the chaining structure of blockchains has attracted extensive attention for their ability to support large-scale decentralized applications. Notable examples include Tendermint [5], Casper FFG [6], HotStuff [7], Streamlet [8], Fast-HotStuff [9], and HotStuff-2 [10]. These protocols have been used in tens of blockchains, both permissioned (*e.g.*, Xuper-Chain [11] and Hyperchain [12]) and permissionless (*e.g.*, Ethereum 2.0 [13], Aptos [14], Cypherium [15], Flow [16], Zilliqa 2.0 [17], and DeSo [18]).

Tendermint [5] and Casper FFG [6] are the first generation of chained BFT protocols that utilize chain structure and pipelining techniques to realize linear message complexity and improve system efficiency. However, they cannot guarantee responsiveness, by which a designated leader can drive nodes to reach a consensus in time depending only on the actual message delays (denoted as δ), independent of any known upper bound delays (denoted as Δ). Responsiveness property is considered as a hallmark of practical BFT protocols, as claimed in [7]. After realizing its importance, subsequent chained BFT protocols such as chained HotStuff (CHS) and Fast-HotStuff (FHS) adopt different designs to achieve it. Specifically, CHS extends two-phase message exchanges to three-phase exchanges, whereas (FHS) includes the latest QC proof collected from nodes in blocks. Despite the introduced protocol overhead, one common belief is that responsiveness can significantly enhance performance, guiding all following designs [10], [19], [20].

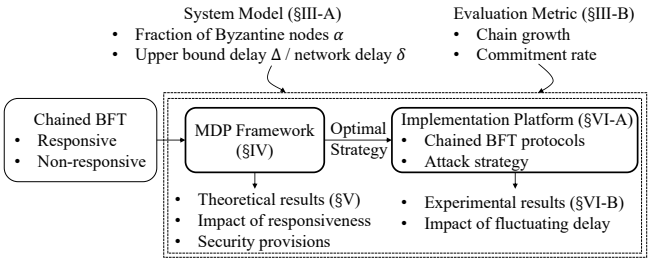


Figure 1: An overview of the proposed framework.

Unfortunately, the responsiveness belief behind chained BFT protocols has not been carefully examined. The prior benchmarks [21]–[24] try to evaluate the performance (*i.e.*, throughput and latency) of chained BFT protocols under various experimental settings. However, they mainly evaluate performance in the ideal cases, *i.e.*, all nodes follow the protocol. Studies [25]–[27] have shown that in chained HotStuff, an adversary can strategically fork uncommitted blocks from honest leaders to weaken the system performance. Since chained BFT protocols are usually deployed among distrusting nodes, it is important to analyze or evaluate the impact of responsiveness on performance under attacks.

In this paper, we propose a comprehensive framework to analyze the impact of responsiveness on system performance. With the framework, we can analyze the performance of existing chained BFT protocols, *i.e.*, chained HotStuff (CHS), Two-Chain HotStuff (2CHS), and Fast-HotStuff (FHS) as well as other chained BFT protocols, under attacks. Rather than using traditional throughput and latency metrics [21]–[24], we propose two new performance metrics: *chain growth* and *commitment rate*. These two metrics can directly reflect performance under attack scenarios and are easier to trace in Markov Decision Processes (MDP) modeling.

Our framework consists of two key components: (i) an MDP-based analytical model and (ii) an evaluation platform, as shown in Fig. 1. The first component enables us to explore optimal adversarial strategies in chained BFT protocols by

considering factors such as the fraction of Byzantine nodes, the impact of network delay, and various consensus rules. With this MDP framework, we can fairly compare the performance of responsive and non-responsive chained BFT protocols under diverse attack scenarios. Although MDP has been used to analyze the performance and security of the Nakamoto-style consensus [28]–[31], these modeling techniques cannot be directly extended to chained BFT consensus due to fundamental design differences. As one of our contributions, we properly simplify the system state and constrain adversary actions to facilitate MDP modeling of various chained BFT protocols.

The second component enables us to implement the optimal attack strategies identified by the MDP modeling on some implementation platforms of chained BFT protocols. In this work, we adopt the open-sourced Bamboo platform [32], which supports CHS, FHS, FHS, and Streamlet [8]. Although Bamboo also supports performance evaluation under attacks, these strategies are straightforward and so cannot provide a fair comparison between chained BFT protocols. Our extension to Bamboo not only can validate the theoretical results from MDP modeling, but also can evaluate performance with consideration of more factors of practical systems such as fluctuating network delay.

Our work provides a way to holistically compare the impact of responsiveness on the system performance. For instance, when the fraction of Byzantine nodes is small, responsive protocols can achieve better chain growth and commitment rates. Our key findings are summarized as follows:

- Finding 1: The chain growth and commitment rate of all three chosen protocols significantly degrade under attacks, especially with a higher fraction of Byzantine nodes. Responsiveness improves the performance of chained BFT protocols at lower fractions of Byzantine nodes but provides diminishing returns as the fraction increases.
- Finding 2: While responsiveness is crucial, its implementation does not always improve protocol performance. The design needs to consider specific protocol needs and the presence of attacks.
- Finding 3: The experiments affirm the framework's validity, demonstrating that chain growth and commitment rate maintain robustness even under realistic network conditions.

Contributions. The contributions of this paper are listed below.

- We propose an evaluation framework with new metrics, chain growth and commitment rate, to systematically analyze the performance of chained BFT protocols.
- We model and evaluate three representative chained BFT protocols under different fractions of Byzantine nodes based on the proposed framework. We obtain the optimal theoretical results and attack strategies. Through an analysis of the results under different fractions, we find the impact of responsiveness on protocol performance
- We conduct extensive empirical experiments to validate the theoretical framework in real-world settings, examining the impact of network delay fluctuations on protocol performance. The integration of theoretical analysis and empirical validation enhances the reliability of our findings.

Code availibility. The source code of MDP and evaluation framework are provided in [33] and [34], respectively.

Roadmap. Sec. II introduces background and related work on chained BFT protocols. Sec. III provides the system model and metrics. Sec. IV present MDP model of chained BFT protocols. The theoretical results of MDP model and experimental results are provided in Sec. V and Sec. VI, respectively. We introduce the limitations and extensions of the framework in Sec. VII and conclude the paper in Sec. VIII.

II. Background and Related Work

We first provide background of chained BFT protocols and the associated performance issues. Then, we review prior studies on the performance analysis of chained BFT protocols.

A. Chained BFT Consensus

Chained BFT consensus represents a family of BFT protocols that leverage the chain structure for better performance and scalability [7]–[9]. Chained BFT protocols run in views, in which a designated replica (called the leader) coordinates with others to vote for its block, as shown in Fig. 2. Specifically, each view can be further divided into two stages: the leader-based stage and the view-change stage. In the former, the leader proposes a new block to extend previous blocks, *i.e.*, forming a chain, according to the *proposing* rule. Then, other nodes append the first valid block from the leader to their chain and update the local state of the locked block. Later, they vote for the block by the voting rule. Once a block has enough votes (from more than $2/3$ of the nodes), it forms a Quorum Certificate (QC) and is certified. Node follows the committing rule to check whether an uncommitted block in the chain can be committed. Note that once a block is committed, its uncommitted ancestor blocks are also committed.

In the view-change stage, nodes safely wedge to the next round if the leader is faulty or the proposed block's quorum certificate is not formed before the timeout. This stage is also referred to as Pacemaker in [7]. It is crucial for the *liveness* property, by which honest clients' transactions are eventually included in committed blocks. Meanwhile, chained BFT protocols also need to ensure the *safety property*, by which honest nodes accept the same committed chain of blocks, referred to as the *main chain*—a key concept that will be used in our analysis. In other words, a block is included in the main chain once being committed.

Responsiveness. Responsiveness requires that a designated leader can drive nodes to reach a consensus in time depending only on the actual message delays (denoted as δ), independent of any known upper bound delays (denoted as Δ). Specifically, the upper bound delay Δ guarantees message delivery between any two honest nodes after Global Stabilization Time (GST), concerned with system liveness, whereas the network delay δ directly affects the system performance. Thus, responsiveness

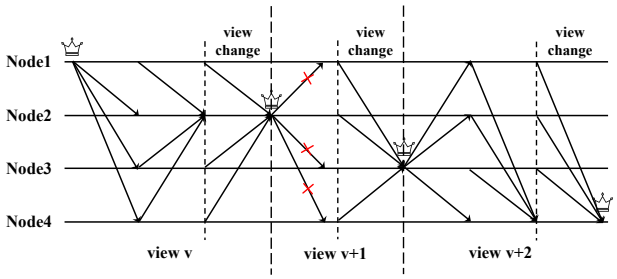


Figure 2: The consensus process of chained BFT protocols.

foregoes a hallmark of practical BFT protocols, as claimed in [7].

Due to its importance, many chained BFT protocols adopt different designs to achieve responsiveness. Specifically, Two-Chain HotStuff (2CHS) formulated from Tendermint [5] and Casper FFG [6] is the original chained BFT protocol without responsiveness. Later, chained HotStuff (CHS) extends two-phase message exchanges to three-phase exchanges. Fast-HotStuff (FHS) introduces the latest QC proof and achieves responsiveness within two-phase exchanges. FHS also optimizes the protocol to skip view change in the happy path. As said previously, achieving responsiveness or not directly affects system performance, further determining their practicality. Thus, exploring the impact of responsiveness or how to achieve better responsiveness is an important question.

Performance under Attacks. The performance of BFT protocols is usually measured in terms of throughput and latency [35], [36]. Throughput refers to the number of transactions that can be processed per unit of time, while latency measures the time taken to reach consensus on a transaction. Applying BFT protocols to blockchain introduces the concept of block, which leads to a greater focus on honest blocks.

The performance of chained BFT protocols can be affected by malicious nodes. A malicious leader may intentionally cause delays in the protocol. For example, it can put off the decision-making or vote-collecting process, leading to slow consensus progress before triggering a timeout. It can also remain silent [27] or propose invalid blocks, causing the entire consensus process to stall until a timeout occurs. The malicious leader can also exclude honest blocks from the chain or disrupt the committing rule of blocks [25], causing the transactions in the block to be delayed. Therefore, it is essential to identify potential vulnerabilities and then measure their impact on performance.

B. Related Work

We review existing studies to analyze the performance of chained BFT protocols, as well as MDP modeling of blockchain consensus protocols. Table I lists existing benchmarks and evaluation frameworks of chained BFT protocols.

Performance Analysis of Chained BFT Protocols. Prior benchmarks [21]–[24], [27] focus on evaluating the throughput and latency metrics of various BFT protocols under different

Table I: Summary of prior frameworks to evaluate the performance of chained BFT protocols under attacks.

Framework	Methodology		Optimal Strategy
	Experimental	Theoretical	
[21]–[24], [27]	✓	✗	✗
[25]	✗	✓	✗
This work	✓	✓	✓

experimental settings. However, most of them do not consider the impact of attacks. Besides, the accuracy of evaluation results is affected by many factors such as implementation and deployment environment. Few studies analyze the performance of CHS protocols under attacks. Niu et al. [25] propose delay attack and forking attack, and provide theoretical analysis. Cohen et al. [26] design an attack and quantify its impact on throughput and latency through experiments. However, these studies only focus on a single protocol and cannot be directly adapted to analyze other protocols.

MDP Modeling of Blockchain Consensus. There have been extensive studies [28]–[31], [37] that apply MDP to evaluate the security and/or performance of blockchain consensus protocols. However, they focus on Nakamoto-style consensus, e.g., Bitcoin [2], Bitcoin-NG [38], or Ethereum [3]. Their models cannot be directly applied to analyze chained BFT protocols that follow significantly different design principles from Nakamoto-style consensus.

III. System Model and Metrics

A. System Model

We consider a system with f Byzantine nodes among n ($n \geq 3f+1$) nodes by following existing chained BFT protocols [6], [7], [9], [10], [39]. The Byzantine nodes can behave arbitrarily, while honest nodes always follow the protocol. We define $\alpha = f/n$ as the fraction of Byzantine nodes, with the remaining $1-\alpha$ representing the fraction of honest nodes. We assume a leader election mechanism in which the probability of electing a Byzantine leader is α, with honest and Byzantine leaders proposing honest and adversarial blocks, respectively. We consider the worst-case situation where a single adversary controls all Byzantine nodes. The adversary tries to worsen the system performance (*i.e.*, minimizing the chain growth and commitment rate introduced shortly) with optimal strategies.

Network Model. We assume a partially synchronous model by following previous work [23]–[25], [27], ensuring all honest nodes are in sync with the protocol's state after Global Stabilization Time (GST). In particular, after GST, a message is guaranteed to be delivered within a known and bounded delay Δ. This paper focuses on analyzing chained BFT protocols during network synchrony (*i.e.*, after GST) as prior analysis or evaluation works [23]–[25], [27].

To show the impact of responsiveness, we assume the actual network delay between two nodes is δ by following prior

work [7], [40]–[42]. This delay varies depending on the real-time network conditions. We assume a fixed δ in the analysis, while considering varied δ in our experiments. We use k to denote the ratio between the maximum bound delay Δ and the actual network delay δ, *i.e.*, $k = \Delta/\delta$.

B. Evaluation Metrics

We introduce two new metrics, *i.e.*, chain growth and commitment rate, to evaluate the performance of chained BFT protocols. Compared to the commonly used throughput and latency metrics [21]–[24], these new metrics reflect performance under attacks more directly and are easier to trace in MDP modeling (introduced shortly). Detailed discussions on these metrics are provided below.

1) Chain Growth: The chain growth $G(\alpha)$ is defined as the rate of honest blocks appended to the main chain, given that the adversary corrupts a fraction α of the total nodes. It indicates the impact of the adversary on system efficiency; the adversary can strategically minimize chain growth to reduce the system capacity for processing clients' transactions, leading to significant service delays or system congestion. Thus, we capture this by minimizing chain growth calculated in m views. We use B_{hi} and T_i to denote the number of honest blocks added to the main chain in the i-th view and time units consumed in the i-th view, respectively. We have:

$$G(\alpha) = \lim_{m\to\infty} \frac{\sum_{i=1}^{m} B_{hi}}{\sum_{i=1}^{m} T_i} \tag{1}$$

This metric considers only honest blocks, since adversarial leaders can selectively exclude honest clients' transactions. Thus, it can reflect the *effective* system throughput. To lower chain growth, the adversary can either prevent honest blocks from being included in the main chain or delay block generation. This metric is first proposed in [25], [27] to evaluate the performance of CHS.

2) Commitment Rate: The commitment rate $R(\alpha)$ is defined as the rate of block commitment event at the main chain, given the fraction of adversarial nodes α. This metric captures system stability to continuously commit blocks and then confirm clients' transactions. The adversary may target to minimize this metric to delay transaction confirmation, increasing transaction delay to affect time-sensitive applications. We use C_i to denote whether blocks are committed in the i-th view. Here, in chained BFT protocols, when a block is committed, all its uncommitted ancestor blocks are also committed. Thus, in a commitment event, several blocks may be committed. We have:

$$R(\alpha) = \lim_{m\to\infty} \frac{\sum_{i=1}^{m} C_i}{\sum_{i=1}^{m} T_i} \tag{2}$$

In chained BFT protocols, a block has to satisfy the commitment conditions to be committed (see Sec.II-A). Thus, the adversary deviates from the protocol to ruin the conditions. Note that this metric can also serve as an indicator of *liveness*. For example, if $R(\alpha)$ is 0 for a chained BFT protocol, it means no transactions can be committed, *i.e.*, no liveness.

IV. MDP Modeling

In this section, we use MDP to model three chained BFT protocols, *i.e.*, 2CHS, CHS, FHS, to show the impact of responsiveness on chain growth and commitment rate. We choose to model these three protocols as i) they achieve distinct responsiveness guarantees, and evaluating them will reveal insights on responsiveness, and ii) they share similar designs such that the model can exclude design variables such as voting message patterns. A detailed description of these three protocols is provided in §II-A.

A. Overview

The Markov Decision Process (MDP) is a sophisticated mathematical framework for formalizing the decision-making process of an agent within a given environment. MDP framework captures the randomness in the agent's decision-making and correlates the rewards it receives with the current state of the environment [43], [44].

MDP provides a way of representing actions taken in a sequence of states, where each state transition is associated with reward allocations. MDP enables us to find optimal strategies that maximize the cumulative rewards. Thus, we can use it to analyze the decision-making processes of the adversary in chained BFT protocols. Specifically, MDP modeling includes four fundamental components: $\langle S, A, P, R\rangle$. In the context of chained BFT protocols, S is a set of all possible states that denote the current status of the chaining blocks and affect the decision-making process. A is the set of all possible actions that can be taken by the adversary. The actions alter the state of the chaining blocks and determine the corresponding rewards. P corresponds to the transition probability function, which represents the likelihood of moving from one state to another based on the action. R reflects the benefits or penalties incurred by the system as a result of the transition, which is essential in guiding the policy search toward actions that yield the most favorable outcomes. Note that reward is not the reward obtained by nodes participating in consensus, but rather the reward of terms in calculating the metrics.

B. Modeling Chained BFT Protocols

In this section, we introduce the MDP modeling of three chained BFT protocols. Due to space constraints, we take CHS as an example to illustrate the modeling process and only describe the differences in the modeling of other protocols.

1) CHS: CHS is the first chained BFT protocol that introduces responsiveness to accelerate the consensus process. Specifically, it introduces one extra phase to the consensus process. The introduced delay of the addition phase is considered insignificant compared to the time saved in view change due to the responsiveness property, as claimed by authors in [7].

State space. The state space is a four-tuple form ($\texttt{cS}$, l_a, l_h, L), as introduced below.

- **Commitment state ($\texttt{cS}$).** The $\texttt{cS}$ represents the consecutive block structure that has already been formed. It has five possible values: $\{0, 1, 2, 3, 3'\}$. This is because CHS specifies

that when three consecutive blocks are formed, the next block triggers a commitment. Therefore, $\texttt{cS}$ is at most 3. The special value $3'$ is used when three consecutive blocks are not continuous with the next block. In this case, the generation of the next block still triggers a commitment, but the value of $\texttt{cS}$ will reset and become 1.

- **Uncommitted honest block (l_h).** The notation l_h represents the number of honest blocks that have not been committed. It has three possible values: $\{0, 1, 2\}$. CHS locks the grandparent of the latest block, in the sense that the grandparent block and its prefix can be committed. Therefore, at most two blocks can be manipulated.
- **Uncommitted adversarial block (l_a).** The notation l_a represents the number of adversarial blocks that have not been committed. It has two possible values: $\{0, 1\}$. The adversary can decide when honest nodes will see a block by releasing or temporarily hiding its block. Since honest nodes follow the consensus rule, once the adversary releases a certified adversarial block, it will not be overridden and can be considered as ultimately committed.
- **Current Leader (L).** The notation L represents the leader type (honest or adversarial) of the current view. It can be either honest (denoted as H) or adversarial (denoted as A).

Actions. The adversary can deviate from the prescribed consensus rules to maximize their benefits. The following actions are available to the adversary within the MDP framework.

- **`Adopt`**. The adversary adopts all available honest blocks that are not locked and discard the hidden adversarial blocks. If the adversary is the leader, it will propose a new block that extends the last adopted block. If the adversary is not the leader, it waits for an honest leader to propose a new block.
- **`Wait`**. If the adversary is elected as the leader, it will create a forking block or extend its prior forking block, to override honest blocks. Otherwise, it will wait for the honest leader to propose a new block.
- **`Release`**. This action is only available when there is a hidden adversarial block. The adversary releases the block before the leader proposes a new block, so the new block will be proposed after it. If the hidden block is a forking block, the non-locked honest blocks will be overridden.
- **`Silent`**. The adversary remains inactive in this view. If the adversary is the leader, it either proposes no block or an invalid one. If the leader is honest, the silence of the adversary does not affect the protocol, and honest nodes continue to agree on the block proposed in this view.

State transition. Table II presents a comprehensive view of the possible state transitions, including the conditions under which each transition occurs and its resulting state. We enumerate all possible state transitions under each possible state, and derive each state transition, including its probability, reward and the resulting state.

- **Commitment state (`cS`).** When the leader is honest, the transition of $\texttt{cS}$ mainly depends on l_a. If $l_a{=}0$, it means that there is no hidden adversarial block, and `Adopt`, `Wait` and `Silent` actions increment $\texttt{cS}$ by 1. If $l_a{>}0$, these actions mean the adversary abandons its block. This breaks the consecutive structure and $\texttt{cS}$ becomes 1. When the adversary chooses `Release` with $l_h{>}0$, it means the released block is a forking block. This again breaks the consecutive structure, and due to the new honest block, $\texttt{cS}$ becomes 2. If $l_h{=}0$, $\texttt{cS}$ increments by 2.

When the leader is adversarial, the transition mainly depends on the action. If an `Adopt` action is taken, when $l_a{=}0$, the adversary will extend the previous honest block, and the consecutive structure will be preserved. Also, as the proposed adversarial block is temporarily hidden, $\texttt{cS}$ remains unchanged. When $l_a{>}0$, the adversary gives up the hidden block, the consecutive structure is destroyed, and $\texttt{cS}$ is reset. If the previous $\texttt{cS}$ equals 3, then $\texttt{cS}$ becomes $3'$, otherwise $\texttt{cS}$ becomes 0. For `Wait` and `Release` actions, if a fork is formed, $\texttt{cS}$ will be reset; If extending the previous adversarial block, $\texttt{cS}$ increases by 1. If the `Silent` action is taken, $\texttt{cS}$ becomes 0.

Table II: State transition and reward matrices for CHS. The reward of B_h and C is analyzed in §IV-B.

State $\times$ Action	Resulting State	Pr.	T
$(\texttt{cS}, 0, l_h, H)$	$(\min(\texttt{cS}{+}1, 3), 0, 1, A)$	α	$\delta{+}2\Delta$
`Adopt`	$(\min(\texttt{cS}{+}1, 3), 0, 1, H)$	$1{-}\alpha$	3δ
$(\texttt{cS}, 1, l_h, H)$	$(1, 0, 1, A)$	α	$\delta{+}2\Delta$
`Adopt`	$(1, 0, 1, H)$	$1{-}\alpha$	3δ
$(\texttt{cS}, 0, l_h, A)$	$(\texttt{cS}, 1, 0, A)$	α	3Δ
`Adopt`	$(\texttt{cS}, 1, 0, H)$	$1{-}\alpha$	$\delta{+}2\Delta$
$(\texttt{cS}, 1, l_h, A)$	$(0/3', 1, 0, A)$	α	3Δ
`Adopt`	$(0/3', 1, 0, H)$	$1{-}\alpha$	$\delta{+}2\Delta$
$(\texttt{cS}, 0, l_h, H)$	$(\min(\texttt{cS}{+}1, 3), 0, \min(l_h{+}1, 2), A)$	α	$\delta{+}2\Delta$
`Wait, Silent`	$(\min(\texttt{cS}{+}1, 3), 0, \min(l_h{+}1, 2), H)$	$1{-}\alpha$	3δ
$(\texttt{cS}, 1, l_h, H)$	$(1, 0, \min(l_h{+}1, 2), A)$	α	$\delta{+}2\Delta$
`Wait, Silent`	$(1, 0, \min(l_h{+}1, 2), H)$	$1{-}\alpha$	3δ
$(\texttt{cS}, 0, l_h, A)$	$(0/3', 1, l_h, A)$	α	3Δ
`Wait`	$(0/3', 1, l_h, H)$	$1{-}\alpha$	$\delta{+}2\Delta$
$(\texttt{cS}, 1, 0, A)$	$(\min(\texttt{cS}{+}1, 3), 1, 0, A)$	α	3Δ
`Wait`	$(\min(\texttt{cS}{+}1, 3), 1, 0, H)$	$1{-}\alpha$	$\delta{+}2\Delta$
$(\texttt{cS}, 1, l_h, A)$	$(1, 1, 0, A)$	α	3Δ
$l_h{>}0$, `Wait`	$(1, 1, 0, H)$	$1{-}\alpha$	$\delta{+}2\Delta$
$(\texttt{cS}, 1, 0, H)$	$(\min(\texttt{cS}{+}2, 3), 0, 1, A)$	α	$\delta{+}2\Delta$
`Release`	$(\min(\texttt{cS}{+}2, 3), 0, 1, H)$	$1{-}\alpha$	3δ
$(\texttt{cS}, 1, l_h, H)$	$(2, 0, 1, A)$	α	$\delta{+}2\Delta$
$l_h{>}0$, `Release`	$(2, 0, 1, H)$	$1{-}\alpha$	3δ
$(\texttt{cS}, 1, 0, A)$	$(\min(\texttt{cS}{+}1, 3), 1, 0, A)$	α	3Δ
`Release`	$(\min(\texttt{cS}{+}1, 3), 1, 0, H)$	$1{-}\alpha$	$\delta{+}2\Delta$
$(\texttt{cS}, 1, l_h, A)$	$(1, 1, 0, A)$	α	3Δ
$l_h{>}0$, `Release`	$(1, 1, 0, H)$	$1{-}\alpha$	$\delta{+}2\Delta$
$(\texttt{cS}, 0, l_h, A)$[a]	$(0, 0, l_h{-}1, A)$	α	2Δ
`Silent`	$(0, 0, l_h{-}1, H)$	$1{-}\alpha$	$\delta{+}\Delta$
$(\texttt{cS}, l_a, l_h, A)$[b]	$(0, 0, l_h, A)$	α	2Δ
`Silent`	$(0, 0, l_h, H)$	$1{-}\alpha$	$\delta{+}\Delta$

[a] $l_h{>}0 \wedge \texttt{cS} \neq 0/3'$
[b] $l_a{=}1 \vee (l_a{=}0 \wedge (l_h{=}0 \vee (l_h{>}0 \wedge \texttt{cS} = 0/3')))$

- **Uncommitted adversarial block (l_a).** If the current leader is adversarial and takes `Adopt`, `Wait`, or `Release` actions, it will propose a temporarily hidden adversarial block, and l_a becomes 1. In all other cases, l_a becomes 0.
- **Uncommitted honest block (l_h).** `Adopt` and `Release` actions will cause l_h to become 0, and l_h will increase by 1 if the current leader is honest, otherwise it will remain unchanged. In the `Wait` action, if the adversary extends its block, the hidden block will be passively released, and l_h will increase by 1 if the current leader is honest, otherwise it will remain unchanged. In `Silent` action, if the leader is honest, l_h will increase by 1. If the leader is adversarial and an honest block is generated in the previous view, the block will be excluded and l_h will be reduced by 1.
- **Current Leader (L).** There are two possible transitions of L upon each action: being adversarial with a probability of α, and being honest with $1-\alpha$.

Reward allocation. Rewards are allocated to certain variables in each state transition and used to calculate the chain growth and commitment rate. The two metrics are calculated from the number of honest blocks in the chain (B_h), the commitment times (C), and the time consumed in the process (T). Due to space constraints, we only show the common variable T used in the two metrics in Table II.

- **Committed honest block (B_h).** B_h will increase by 1 if a block is eventually committed on the chain. After the adversary takes `Adopt` action, the existing honest blocks will not be overridden, B_h will increase by l_h. In addition, B_h will increase by 1 when l_h transits to $l_h{+}1$ if $l_h{=}2$. Due to the locking rule, if $l_h{=}2$, the subsequent block will cause its grandparent block to be locked, and B_h will also increase by 1.
- **Commitment times (C).** C increases by 1 for each commitment. In CHS, when a new block extends three consecutive blocks, it triggers a commitment. This requires `cS` to be 3 or $3'$, and meets one of the following three conditions: 1) the current leader is honest and the adversary takes `Adopt`, `Wait`, or `Silent` actions; 2) The current leader is honest, and the adversary takes `Release` action without any forks ($l_h{=}\,0$); or 3) The current leader is adversarial and takes `Wait` or `Release` actions without a fork ($l_h{=}\,0$).
- **Elapsed time (T).** Recall that we use δ and Δ to denote the actual network delay and upper bound network delay after GST, respectively. The time consumed within each view consists of three parts: 1) the leader broadcasts a block, 2) the next leader collects votes from other nodes, and 3) the view-change process. For the first two parts, they take δ if the leader is honest, and Δ otherwise.

  For the last one, it is determined by the next leader and the responsiveness/non-responsiveness properties of the protocol. Due to the responsiveness of CHS, the view-change process takes δ if the leader is honest and Δ otherwise. The leader of two consecutive views can be divided into four groups, and their consumption time differs. Note that if an adversarial leader takes the `Silent` action, honest nodes will trigger timeout upon the leader proposing the block (part 1) and directly entering the view change (part 3) without collecting votes (part 2).

2) 2CHS: 2CHS shares a similar design as CHS and Casper [6], and can also be viewed as the pipelined version of Tendermint [5]. As mentioned previously, 2CHS has one less phase than CHS, but does not have responsiveness. Besides, the commitment of 2CHS slightly differs from CHS. In 2CHS, the first block of two consecutive blocks and its previous blocks will be committed if a new block extends the second block. In other words, in 2CHS, the parent block of a certified block, instead of the grandparent block, is locked.

The state space of 2CHS has the same four-tuple as CHS. Due to the two consecutive blocks structure, l_h is at most 1, and `cS` can be $\{0, 1, 2, 2'\}$. This also leads to corresponding changes in the state transition and reward allocation of B_h and C. When l_h increases from 1 to $l_h{+}1$, B_h will increase by 1. When `cS` becomes 2 or $2'$, honest nodes trigger a commitment, and C increases by 1. For time consumption, the rewards for T in the leader-based stage remain unchanged, but since 2CHS is not responsive, the view change part always takes Δ time regardless of the identity of the next leader.

3) FHS: FHS is a responsive two-phase chained BFT protocol, which allows a leader to included $f + 1$ latest QCs in its block. Compared with CHS, the QC inclusion in FHS slightly increases the message exchange overhead, but does not require an additional phase to achieve responsiveness. Despite FHS's resilience against forking attacks, honest nodes may still lose blocks. This is because votes sent to a subsequent adversarial leader could be hidden, preventing the formation of a QC. As a result, the actions available to the adversary in FHS are the same as 2CHS.

The state transition and reward allocation of FHS are the same as 2CHS, except for T. Besides responsiveness, FHS optimizes the view change process by incorporating the "happy path" mechanism, where the next leader skips the view change if they successfully form the QC of the previous block. This QC is ensured to be the highest QC, so honest nodes can directly propose a new block based on this QC. This mechanism further reduces T.

V. Theoretical Results

In this section, we evaluate the performance of three chained BFT protocols based on our MDP model. The evaluation aims to answer a key question: *How do the responsiveness property and different responsiveness designs affect the chain growth and commitment rate of chained BFT protocols?*

To answer this question, we apply the algorithm proposed by Sapirshtein et al. [28] to find the optimal strategy for minimizing chain growth and commitment rate in our MDP model. In particular, we calculate $G(\alpha)$ and $R(\alpha)$ for values of α ranging from 0 to $1/3$ (the system's maximum tolerable Byzantine nodes), incrementing α by 0.03 each step, with a precision of 10^{-4}. We assume that the maximum network delay Δ and the actual delay δ satisfy the relationship $\Delta{=}5\delta$.

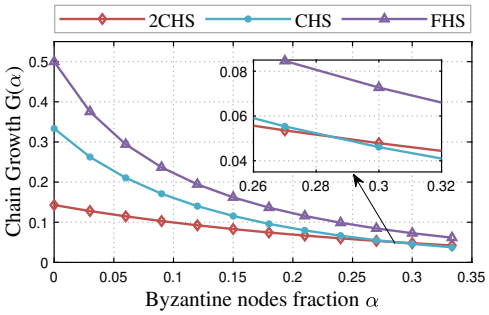


Figure 3: The chain growth of 2CHS, CHS, and FHS.

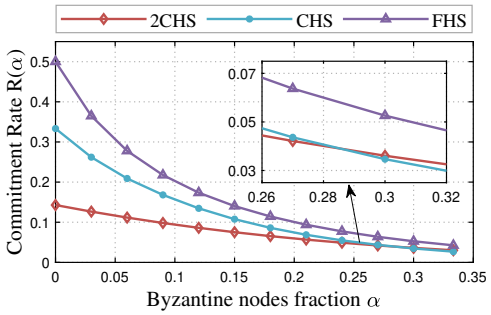


Figure 4: The commitment rate of 2CHS, CHS, and FHS.

We obtain the theoretical optimal values and attack strategies through the framework, and then compare our results with existing attack strategies. The MDP source code is provided in [33].

A. Theoretical Results with Optimal Strategy

We consider the chain growth and commitment rate of the protocols under optimal attack strategies found by the MDP model. Fig. 3 and Fig. 4 provide the evaluation results of chain growth and commitment rate, respectively.

The performance under attacks. First, there is a downward trend in both of chain growth and commitment rate of the three protocols. Fig. 3 shows that, when the fraction α of Byzantine nodes is 0.3, the chain growth of CHS decreases from 0.333 to 0.046, and chain growth of FHS decreases from 0.5 to 0.073. For 2CHS, the chain growth drops to nearly one-third when α is close to 0.3. Fig. 4 shows that as α increases from 0 to 1/3, the commitment rate of CHS drops from 0.333 to 0.027, FHS drops from 0.5 to 0.042, and 2CHS decreases from 0.143 to 0.03. This is because when α increases, more adversarial nodes are elected as leaders, and adversarial leaders can attack the two performance metrics.

The impact of responsiveness. For CHS and FHS with responsiveness, chain growth and commitment rate are decreased significantly with smaller α. For example, in FHS, when α increases from 0 to 0.2, the chain growth decreases by 31%; and when α increases from 0.2 to $1/3$, chain growth decreases by 13%. In addition, with larger α, responsiveness brings less benefits to performance: responsive CHS and FHS achieve comparable chain growth and commitment rate with non-responsive 2CHS when α reaches 1/3. When α is small, the impact of the adversary's attacks is limited, and the responsiveness property can effectively accelerate the view change. So the impact of responsiveness on performance is relatively large when α is small, but as α increases, its advantage gradually weakens.

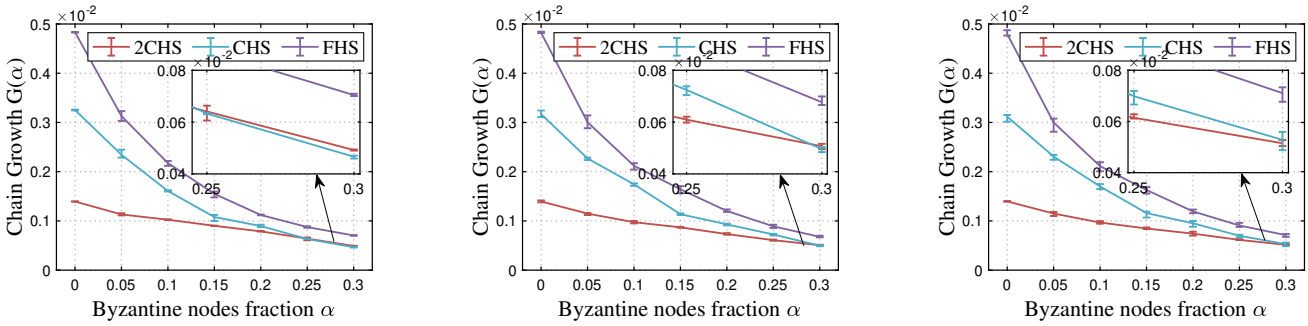
We observe that FHS outperforms the other two protocols, and 2CHS is less performant than the other two protocols in most cases. When $\alpha = 0$, the chain growth of FHS is 3.5 and 1.5 times compared to 2CHS and CHS, respectively. When $\alpha = 1/3$, the chain growth of FHS becomes 1.5 and 1.6 times compared to 2CHS and CHS. When $\alpha = 0.3$, the commitment rate of FHS becomes 1.5 times compared to 2CHS and CHS. The evaluation lines of 2CHS and CHS intersect between $\alpha = 0.27$ and $\alpha = 0.3$. After the intersection, 2CHS performs better than CHS. This is because of the commit rule of three consecutive blocks in CHS. For chain growth, the additional phase allows the adversary to override two honest blocks in a forking attack. For commitment rate, triggering a block commitment is less likely compared to the rule of two consecutive blocks, especially with larger α. This indicates that in some cases, implementing responsiveness by adding a new phase may not necessarily improve the performance of the protocol. This is also evidenced in FHS that achieves better chain growth and commitment rate due to its optimization for the happy paths.

Summary. The evaluation shows that attacks can significantly degrade chain growth and commitment rate, especially when the fraction α is large. Furthermore, achieving responsiveness can improve chain growth and commitment rate when α is small. However, when α is relatively large, responsiveness brings less benefit to both metrics. It indicates that although responsiveness is an important property, certain designs for achieving responsiveness may not necessarily improve protocol performance in all cases. Appropriate implementation needs to be considered based on protocol needs, especially in the presence of attacks.

B. Comparison to Existing Attack Strategies

Modeling chained BFT protocols into MDPs enables us to identify the optimal strategies and associated worst performances. To show the effectiveness, we compare our strategies with the strategy in [20], [27], where adversarial leaders keep silent in their views. This will trigger a timeout in the proposing stage, and all nodes enter the view change. Meanwhile, it will reset the count for consecutive blocks, preventing block commitment.

Fig. 6 shows the commitment rate under different strategies. Both attack strategies reduce the commitment rate compared with the baseline. With larger α, the gap between the evaluation results with and without attack strategies increases. When $\alpha = 0.3$, commitment rate under the simple attack strategy and our attack strategy decreases to 12% and 10% of the baseline, respectively. The simple attack strategy is less effective than



(a) Network delay $\delta = 100ms$. (b) Network delay $\delta = 100 \pm 25ms$. (c) Network delay $\delta = 100 \pm 50ms$.

Figure 5: The experimental results chain growth of 2CHS, CHS, and FHS with varing δ.

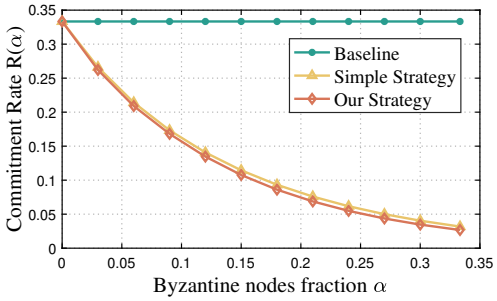


Figure 6: The commitment rate of CHS under our strategy, the simple strategy, and the baseline.

our attack strategy. This is because our framework can guide the adversary to adopt the optimal action in certain scenarios, making it difficult to determine manually. Furthermore, by comparing the adversarial strategies, we find that the exception arises when an adversarial node is elected as the leader and there is no existing three-consecutive block structure. At this moment, the adversary can propose a block, but selectively send the proposal to only a subset of honest nodes. This strategy prevents honest nodes from collecting sufficient votes to form a QC, and meanwhile avoids the timeout during the proposal phase. By doing so, the adversary resets the number of consecutive blocks and simultaneously prolongs the consumed time. This more sophisticated attack strategy leverages the protocol's characteristics more effectively, thereby exerting a greater impact on commitment rate.

VI. Experiments

The theoretical models provide a foundation for understanding the impact of adversarial behaviors on system performance under various conditions. To study the practical implications of the theoretical results, we implement the chained BFT protocols and evaluate their chain growth and commitment rate in real-world scenarios. Besides, the evaluation performance with consideration of more factors of practical systems such as fluctuating network delay. The adversary in our experiments is equipped with the attack strategies found by MDP in §V. The experiment will check whether the experimental results are consistent with the theoretical results and explore the impact of fluctuating network delay δ.

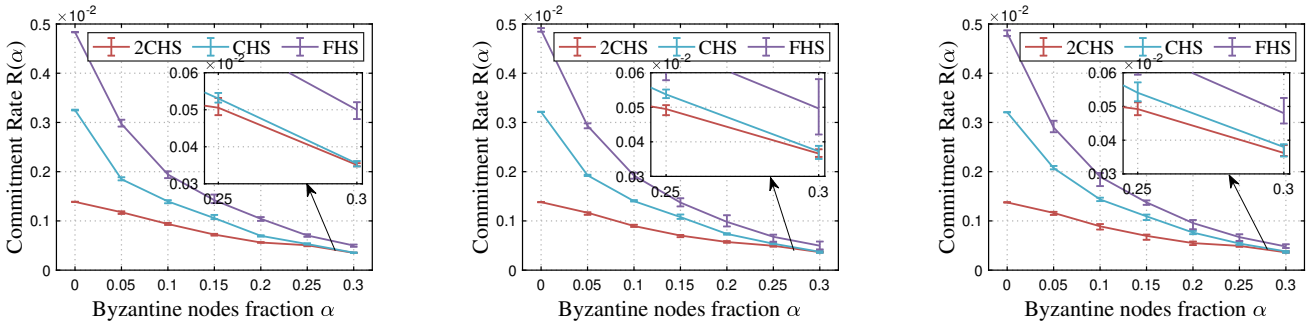
A. Implementation and Setup

Implementaion. We have extended an open-source evaluation platform Bamboo [27] in the Go language to implement the chained BFT protocols. The source code can be found at [34]. Our modifications add approximately 1000 lines of Golang code to Bamboo for implementing CHS, 2CHS, and FHS. We implement the strategies of forking attacks and silience attacks by adjusting the logic of view change, proposing and voting stages. These adversarial behaviors slow down the block generation and block commitment.

System setup. The experimental environment includes a cluster consisting of 4 servers, each equipped with a 16-core CPU, 32GB RAM with the operating system of Ubuntu Server 22.04. The network latency between servers is approximately 1ms. The environment includes a controlled network environment with a specified number of nodes. The key parameters for the experiments include the fraction of Byzantine nodes in the network and the network latency settings. To simulate a realistic network scenario, we set up a network topology that could accommodate up to 60 nodes (each server supports 15 nodes), with the capability to introduce up to 18 Byzantine nodes to test the system's performance under attacks. To be consistent with the theoretical setting, we first set the minimum delay between nodes at 100 milliseconds and the maximum latency for communication at 500 milliseconds. Meanwhile, in order to demonstrate the impact of a random network environment, we set the minimum delay between nodes to a fluctuating value between 25 milliseconds and 50 milliseconds, and conduct corresponding experiments based on the previous ones. The fluctuation of delay conforms to a uniform distribution. We sampled the execution of 10 minutes for each group of experiments and averaged the results over 6 distinct executions.

B. Chain Growth

We first use a fixed network delay $\delta = 100ms$ in experiments. Fig. 5a shows that the experimental results closely



(a) Network delay $\delta = 100ms$. (b) Network delay $\delta = 100 \pm 25ms$. (c) Network delay $\delta = 100 \pm 50ms$.

Figure 7: The experimental results commitment rate of 2CHS, CHS, and FHS with varing δ.

match the results in §V. The chain growth of CHS, 2CHS and FHS gets worse as α increases. When $\alpha = 0.3$, chain growth of 2CHS drops to 35% of that when $\alpha = 0$. FHS outperforms 2CHS and CHS in chain growth, and 2CHS and CHS have an intersection point.

In the presence of network delay fluctuations, as shown in Fig. 5b and Fig. 5c, we observe that the performance of chain growth remains largely consistent with the stable delay scenario. However, the measurement errors indicated by the error bars increase with the delay fluctuations. In the case of CHS, with a fluctuation range of 25ms, the measurement error is 1.3 times compared to the stable delay, and with a fluctuation range of 50ms, the error doubles compared to without fluctuation. Overall, the experimental results of chain growth with stable delay are consistent with the theoretical results, thus confirming the findings in §V.

C. Commitment Rate

Fig. 7a shows the commitment rate results under stable network delay. The commitment rate also shows a decline as α grows, with FHS outperforming 2CHS and CHS. As α increases from 0 to 0.3, commitment rate of FHS decreases by 11%. Fig. 7b and Fig. 7c show the evaluation results with a delay fluctuation amplitude of 25 and 50 milliseconds. The commitment rate is similar to that with stable delay. The measurement errors increase with increasing fluctuations. Regarding 2CHS, at fluctuation ranges of 25ms and 50ms, the measurement errors are 1.2 times and 2.1 times compared to those without fluctuations, respectively. For FHS, the measurement errors become 2 and 2.3 times of the original values, respectively. CHS and FHS show more significant deviations compared to 2CHS. While the measurement error increases with more fluctuating network latency, the average evaluation results remain largely unchanged and the observed trend is consistent with previous experiments.

In a nutshell, the experiments provide direct evidence supporting the validity of the framework, and the results of chain growth and commitment rate from the framework remain robust even in realistic network settings.

VII. Discussion

Limitations of the framework. There are some limitations that can be addressed in future works. First, the adversary's behavior is simplified in the MDP modeling to analyze the worst-case scenario. For example, we assume the adversary coordinates all Byzantine nodes without message delay, which may not hold in dynamic and diverse network conditions. As a result, the system performance under attack scenarios may be underestimated. Second, our analysis does not consider the recent BFT protocols with accountability and forensics support [45], [46], in which honest nodes can detect certain attacks and identify Byzantine nodes. Thus, some attack strategies of the adversary may be prohibited. Third, our analysis focuses on synchronous periods after GST. Modeling and evaluating chained BFT protocol in asynchronous periods (*i.e.*, before GST) may reveal more insights about the impact of responsiveness.

Extending the framework. Apart from the protocols analyzed in this paper, there are several chained BFT protocols, such as Streamlet [8], Marlin [19], and BeeGees [20]. Since they have not been deployed in real distributed systems, we select three classic protocols that have already been practically applied. Nevertheless, our modeling framework can be applied to these protocols with slight modifications. For example, our framework is still applicable to Streamlet. Due to space constraints, we show our further analysis of Streamlet in Appendix A [47].

VIII. Conclusion

In this paper, we analyze the performance of chained BFT protocols through a unified framework based on MDP, including chain growth and commitment rate. The evaluation of three representative protocols provides the worst-case performance, and reveals the effect of responsiveness, which inspires the protocol design. The framework also provides the optimal attack strategies, discovering better attack strategies than previous work. We also conduct an experimental deployment and verify the correctness of our framework. In addition, the framework is scalable and provides a convenient tool for comparing the performance of different protocols.

REFERENCES

[1] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proc. of SOSP*, 2017.

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

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger Byzantium version," *Ethereum project yellow paper*, pp. 1–32, 2018. [Online]. Available: https://github.com/ethereum/yellowpaper

[4] D. Labs, "Crypto kitties," https://www.cryptokitties.co/.

[5] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," *M.Sc. Thesis, University of Guelph, Canada*, Jun 2016.

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

[7] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT consensus with linearity and responsiveness," in *PODC*, 2019.

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

[9] M. M. Jalalzai, J. Niu, C. Feng, and F. Gai, "Fast-HotStuff: A fast and robust BFT protocol for blockchains," *TDSC*, pp. 1–17, 2023.

[10] D. Malkhi and K. Nayak, "HotStuff-2: Optimal two-phase responsive BFT," *Cryptology ePrint Archive*, 2023.

[11] X. Lab, "XuperChain Platform," June 2021. [Online]. Available: https://github.com/xuperchain/xuperchain

[12] "Introduction to Hyperchains," https://blog.matter-labs.io/introduction-to-hyperchains-fdb33414ead7, retrieved Nov, 2023.

[13] C. Schwarz-Schilling, J. Neu, B. Monnot, A. Asgaonkar, E. N. Tas, and D. Tse, "Three attacks on Proof-of-Stake Ethereum," in *FC*, 2022.

[14] "Aptos homepage," https://aptoslabs.com, retrieved Nov, 2023.

[15] 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

[16] A. Hentschel, Y. Hassanzadeh-Nazarabadi, R. Seraj, D. Shirley, and L. Lafrance, "Flow: Separating consensus and compute–block formation and execution," *arXiv preprint arxiv:2002.07403*, 2002.

[17] J. McKane, "EVM and the road to Zilliqa 2.0 - Upgrading network efficiency," https://blog.zilliqa.com/evm-and-the-road-to-zilliqa-2-0-upgrading-network-efficiency/, retrieved Nov, 2023.

[18] "Revolution Proof of Stake," https://revolution.deso.com/, retrieved Nov, 2023.

[19] X. Sui, S. Duan, and H. Zhang, "Marlin: Two-phase BFT with linearity," in *DSN*, 2022.

[20] N. Giridharan, F. Suri-Payer, M. Ding, H. Howard, I. Abraham, and N. Crooks, "BeeGees: stayin' alive in chained BFT," in *PODC*, 2023.

[21] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *SIGMOD*, 2017.

[22] G. Shapiro, C. Natoli, and V. Gramoli, "The performance of Byzantine fault tolerant blockchains," in *NCA*, 2020.

[23] M. J. Amiri, C. Wu, D. Agrawal, A. E. Abbadi, B. T. Loo, and M. Sadoghi, "The BEDROCK of BFT: A unified platform for BFT protocol design and implementation," *arXiv preprint arXiv:2205.04534*, 2022.

[24] V. Gramoli, R. Guerraoui, A. Lebedev, C. Natoli, and G. Voron, "Diablo: A benchmark suite for blockchains," in *Eurosys*, 2023.

[25] J. Niu, F. Gai, M. M. Jalalzai, and C. Feng, "On the performance of pipelined HotStuff," in *INFOCOM*, 2021.

[26] S. Cohen, R. Gelashvili, L. K. Kogias, Z. Li, D. Malkhi, A. Sonnino, and A. Spiegelman, "Be aware of your leaders," in *FC*, 2022.

[27] F. Gai, A. Farahbakhsh, J. Niu, C. Feng, I. Beschastnikh, and H. Duan, "Dissecting the performance of Chained-BFT," in *ICDCS*, 2021.

[28] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," in *FC*, 2017.

[29] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of Proof of Work blockchains," in *CCS*, 2016.

[30] R. Zhang and B. Preneel, "On the necessity of a prescribed block validity consensus: Analyzing Bitcoin unlimited mining protocol," in *CoNEXT*, 2017.

[31] J. Niu, Z. Wang, F. Gai, and C. Feng, "Incentive analysis of bitcoin-ng, revisited," *SIGMETRICS Perform. Eval. Rev.*, vol. 48, no. 3, p. 59–60, mar 2021.

[32] "Bamboo: Evaluation framework of chained BFT," https://github.com/gitferry/bamboo, retrieved Nov, 2023.

[33] A. Authors, "Mdp source code," https://anonymous.4open.science/r/info997-67C3/MDPModel/.

[34] ——, "Attack implementation based on bamboo," https://anonymous.4open.science/r/info997-67C3/experiment/.

[35] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 bft protocols," *TOCS*, vol. 32, no. 4, pp. 1–45, 2015.

[36] D. Gupta, L. Perronne, and S. Bouchenak, "Bft-bench: A framework to evaluate bft protocols," in *ICPE*, 2016, pp. 109–112.

[37] C. Feng and J. Niu, "Selfish mining in ethereum," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1306–1316.

[38] 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.

[39] E. Shi, "Streamlined blockchains: A simple and elegant approach (a tutorial and survey)," in *Asiacrypt*, 2019.

[40] I. Abraham, K. Nayak, and N. Shrestha, "Optimal good-case latency for rotating leader synchronous BFT," *Cryptology ePrint Archive*, 2021.

[41] A. Momose, J. P. Cruz, and Y. Kaji, "Hybrid-BFT: Optimistically responsive synchronous consensus with optimal latency or resilience," *Cryptology ePrint Archive*, 2020.

[42] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and M. Yin, "Sync HotStuff: Simple and practical synchronous state machine replication," in *SP*. IEEE, 2020, pp. 106–118.

[43] O. Sigaud and O. Buffet, *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

[44] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[45] P. Sheng, G. Wang, K. Nayak, S. Kannan, and P. Viswanath, "BFT protocol forensics," in *CCS*, 2021.

[46] A. Shamis, P. Pietzuch, B. Canakci, M. Castro, C. Fournet, E. Ashton, A. Chamayou, S. Clebsch, A. Delignat-Lavaud, M. Kerner, J. Maffre, O. Vrousgou, C. M. Wintersteiger, M. Costa, and M. Russinovich, "IA-CCF: Individual accountability for permissioned ledgers," in *NSDI*. USENIX Association, Apr. 2022. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/shamis

[47] A. Authors, "Unraveling responsiveness of chained BFT consensus with network delay," https://anonymous.4open.science/r/info997-67C3/report.pdf.

APPENDIX

A. Streamlet

Streamlet is a streamlined chained BFT protocol that is different from 2CHS, CHS and FHS. Firstly, it follows the longest certified chain rule, where an honest node only votes to the block that extends the longest certified chain. Due to this rule, the forking attack described in the main text will not appear in Streamlet. If the adversary tries to override an honest block, the honest nodes will not vote for it. However, the longest certified chain rule also introduces another way of attack. If the adversarial leader sends its proposal of block B to only part of $(1/3 < \#honestnodes/\#nodes \leq 2/3)$ the honest nodes, excluding the next leader. The next leader does not see B and thus proposes a block B' at the same height of B. After this new proposal, the adversarial nodes release their votes on B. At this point, honest nodes that previously received B will consider it the longest chain and will not vote for B'. We can consider this type of attack as preemptive forking attack. This difference will add a new action to the action space A.

Table III: State transition and reward matrices for Streamlet.

State $\times$ **Action**	**Resulting State**	**Pr.**	B_h	C
$(\texttt{cS}, 0, l_h, H)$ Adopt	$(\min(\texttt{cS}+1, 3), 0, 1, A)$ $(\min(\texttt{cS}+1, 3), 0, 1, H)$	α $1-\alpha$	l_h	if $\texttt{cS}= 2/3$ C =1
$(\texttt{cS}, 1, l_h, H)$ Adopt	$(1, 0, 1, A)$ $(1, 0, 1, H)$	α $1-\alpha$	l_h	
$(\texttt{cS}, 0, l_h, A)$ Adopt	$(\texttt{cS}, 1, 0, A)$ $(\texttt{cS}, 1, 0, H)$	α $1-\alpha$	l_h	
$(\texttt{cS}, 1, l_h, A)$ Adopt	$(0, 1, 0, A)$ $(0, 1, 0, H)$	α $1-\alpha$	l_h	
$(\texttt{cS}, 0, l_h, H)$ Wait, Silent	$(\min(\texttt{cS}+1, 3), 0, l_h+1, A)$ $(\min(\texttt{cS}+1, 3), 0, l_h+1, H)$	α $1-\alpha$	0	if $\texttt{cS}= 2/3$ C =1
$(\texttt{cS}, 1, l_h, H)$ Wait, Silent	$(1, 0, l_h+1, A)$ $(1, 0, l_h+1, H)$	α $1-\alpha$	0	
$(\texttt{cS}, 0, l_h, A)$ Wait	$(0, 0, l_h, A)$ $(0, 0, l_h, H)$	α $1-\alpha$	0	
$(\texttt{cS}, 1, l_h, A)$ Wait	$(\min(\texttt{cS}+1, 3), 1, 0, A)$ $(\min(\texttt{cS}+1, 3), 1, 0, H)$	α $1-\alpha$	l_h	if $\texttt{cS}= 2/3$ C =1
$(\texttt{cS}, 1, l_h, H)$ Release	$(\min(\texttt{cS}+2, 3), 0, 1, A)$ $(\min(\texttt{cS}+2, 3), 0, 1, H)$	α $1-\alpha$	l_h	(1)
$(\texttt{cS}, 1, l_h, A)$ Release	$(\min(\texttt{cS}+1, 3), 1, 0, A)$ $(\min(\texttt{cS}+1, 3), 1, 0, H)$	α $1-\alpha$	l_h	if $\texttt{cS}= 2/3$ C =1
$(\texttt{cS}, 1, l_h, H)$ Withhold	$(0, 0, 0, A)$ $(0, 0, 0, H)$	α $1-\alpha$	l_h	
$(\texttt{cS}, 1, l_h, A)$ Withhold	$(\min(\texttt{cS}+1, 3), 1, 0, A)$ $(\min(\texttt{cS}+1, 3), 1, 0, H)$	α $1-\alpha$	l_h	if $\texttt{cS}= 2/3$ C =1
$(\texttt{cS}, l_a, l_h, A)$ Silent	$(0, 0, 0, A)$ $(0, 0, 0, H)$	α $1-\alpha$	0	

(1) If $\texttt{cS}$ =1, C =1; if $\texttt{cS}$ =2/3, C =2

Secondly, the lock and commit rules of Streamlet (referred to as notarize and final respectively in Streamlet) are different from the previous three protocols. Due to the broadcasting vote pattern, nodes can collect votes and generate QC locally at the end of each view. The block will be locked once certified. If a structure of three consecutive blocks is formed, the first two blocks will be committed. This difference will affect the reward allocation for B_h and C.

Moreover, Streamlet does not have view change step and is not responsive, so the time consuming in each view is the same, i.e. 2Δ. This mainly affects the reward distribution of T. The MDP modeling of Streamlet is shown in Table III in detail.