Streamlet: Textbook Streamlined Blockchains

Benjamin Y Chan[1] and Elaine Shi[1]

[1]Cornell University - {byc, elaine}@cs.cornell.edu

January 24, 2020

Abstract

In the past five years or so, numerous blockchain projects have made tremendous progress towards improving permissioned consensus protocols (partly due to their promised applications in Proof-of-Stake cryptocurrencies). Although a significant leap has silently taken place in our understanding of consensus protocols, it is rather difficult to navigate this body of work, and knowledge of the new techniques appears scattered.

In this paper, we describe an extremely simple and natural paradigm for constructing consensus protocols called Streamlet. Our protocols are inspired by the core techniques that have been uncovered in the past five years of work; but to the best of our knowledge our embodiment is simpler than ever before and we accomplish this by taking a "streamlining" idea to its full potential. We hope that our textbook constructions will help to decipher the past five year's of work on consensus partly driven by the cryptocurrency community — in particular, how remarkably simple the new generation of consensus protocols have become in comparison with classical schemes such as PBFT and Paxos.

1 Introduction

Distributed consensus allows a set of players to agree on an ever-growing, linearly ordered log of transactions, and is the core abstraction behind modern cryptocurrency systems such as Bitcoin and Ethereum. Since transactions are typically batched into "blocks" during consensus, we also refer to such consensus protocols as *blockchain protocols*.

As we all know, by leveraging Proof-of-Work (PoW), Bitcoin's Nakamoto consensus [9,17,19,20] was the first to achieve consensus in an open environment where everyone can join as a participant. However, partly due to the enormous energy waste of PoW, the community has been pushing for a new paradigm called Proof-of-Stake (PoS), where, roughly speaking, players have voting power proportional to their stake (i.e., in terms of cryptocurrencies owned) in the system. Interesting, as many works have shown [6, 7, 12], PoS systems actually return to the roots of the classical distributed systems literature, and rely instead on "permissioned" consensus as a core building block. In essence, these PoS systems typically employ a committee reconfiguration mechanism to rotate the consensus participants over time based on the latest stake distribution; and at any snapshot of time, the elected participants run a permissioned consensus protocol — motivating a renewed focus on the permissioned setting.

Consensus protocols are notoriously difficult to understand [18]. In this paper, building on recent work motivated by new applications in blockchain protocols, we present an absurdly simple protocol that we hope simplifies the consensus problem once and for all.

1.1 The Streamlined Blockchain Paradigm

In this paper, we focus on how to construct such a permissioned consensus protocol. Specifically, we consider a classic setting where the set of players is known a-priori and their public keys are common knowledge. A subset of the players may be corrupt, and corrupt players can deviate from the protocol arbitrarily (also called the Byzantine fault model). Partly driven by the PoS paradigm, in the past five years, cryptocurrency developers and academic researchers have made tremendous efforts at designing better permissioned consensus protocols, and a plethora of protocols [1, 2, 4–6, 11, 14, 22] have been proposed, making it difficult to navigate and compare.

Recently, a tutorial by Shi [21] tried to decipher the recent body of work on distributed consensus. Shi points out that in fact, a tremendous leap has silently taken place: thanks to the community's joint efforts, we now have a new family of protocols, referred to by Shi as "streamlined blockchains", that are *remarkably simple* and depart from the framework adopted by classical protocols such as PBFT [3] and (Byzantine) Paxos [15].

More concretely, almost all known classical protocols such as PBFT [3], Paxos [15], and their variants [10,13], adopt a bi-modal approach: the protocol typically consists of a simple normal path where a leader makes proposals and everyone votes; and when the normal path fails, the protocol switches to a (typically much more complicated) fall-back mode typically called a "view change" [3]. From a technical perspective, the normal path promises only consistency and offers no liveness if the leader is corrupt; whereas the view change protocol must essentially embed a "full-fledged" consensus that offers both consistency and liveness. As such, the complexity of classical protocols such as PBFT and Paxos arises due to the view change.

By contrast, Shi [21] described streamlined blockchains as a family of protocols that are (almost) as simple and natural as the normal path of classical protocols, but provide full consensus. In a streamlined blockchain protocol, there is only one mode of operation: a leader proposes a block, players vote, and this propose-vote process repeats over time for confirming more and more blocks. Essentially, by making a couple minor and natural tweaks to the normal path of classical consensus (e.g., PBFT and Paxos), we can get full consensus!

We point out that like in most schemes that have been adopted in practice, we directly construct a blockchain; whereas an alternative approach is through sequential and/or parallel composition of single-shot consensus instances (often called Byzantine Agreement [16]). The former approach (i.e., direct construction) typically results in a blockchain protocol where there is no clear-cut boundary between consensus instances — this is favorable in practice partly because it facilitates system-level pipelining and optimization (some of which are "cross-instance" in nature). In our protocol, every epoch implicitly offers a view-change opportunity, and this is also why we can have a different leader in every epoch.

1.2 Goal: A Simplest-Possible, Unified Protocol

In the same tutorial, Shi [21] also advocated for *having a simplest-possible, unified protocol under the streamlined blockchain paradigm — such that not only is this protocol an ideal candidate for teaching, but also an ideal candidate for practical implementation and deployment.*

Like in [21], we take simplicity to be a first-order principle. Especially with consensus protocols, complexity often results in safety or liveness bugs in practice, a priori defeating the point of performance optimization in applications where safety is important (like cryptocurrency). A protocol that is hard to understand makes for code that is hard to understand. Therefore, we strongly believe that designing simpler protocols is important work; if it is performant, even better.

One area for improvement immediately comes to mind: the protocols described in Shi's tutorial [21] (and the closely related Pili [5] and Pala [4]) use a freshness/staleness trick where voters must check that the proposed block is extending from a parent chain that is "not too stale" by some technical definition of "staleness" before voting on the block. The definition of staleness is subtle and requires significant care. Unfortunately, this complexity has proven to be a central point of confusion while teaching the protocol and throughout attempts to implement the protocol.

An ideal protocol would avoid complexity such as the staleness check, while preserving consistency and liveness. This is non-trivial. We emphasize that even though the output of this process may be a simple 5-line protocol, it is the culmination of a long line of work by the distributed systems community, marred by failure, to make consensus understandable to all.

1.3 Our Contributions

In this work, we make a further attempt at designing a simplest-possible, textbook streamlined blockchain protocol. We present a new family of protocols called Streamlet, which we believe to be simpler and more natural than any of the existing efforts towards this direction [1,4,5,21,22] (we will elaborate on the comparison later on). We stress that our goal is *not* to propose a new consensus protocol with theoretically novel properties, but rather, to simplify consensus to an extreme such that the entire protocol is the simplest possible, takes little effort to understand and memorize, and thwarts mistakes during implementation in the sense that there is no subtlety in the protocol description that a non-expert would find unintuitive.

Specifically, we present two instantiations of our Streamlet paradigm: one that is secure against $< 1/3$ corruptions and tolerates arbitrary network delays; and one that secures against minority corruptions and whose safety guarantees rely on network synchrony assumptions.

Since our protocols are extremely simple, we informally describe the Streamlet paradigm right off the bat. The protocol below is for the $1/3$ setting and tolerates arbitrary network delays — such protocols are said to be partially synchronous [8] — but we will later explain how to make a couple minor tweaks to turn it into a synchronous, honest-majority protocol.

Consider a network where delays can be arbitrary. We would like to have a consensus protocol that always preserves *consistency* regardless of the network delay; however, when the network conditions are good (often called a period of synchrony), i.e., when honest processes can send messages to each other within Δ rounds of delay, we would like the protocol to achieve *liveness*.

Imagine that the protocol continues in incrementing epochs, where each epoch consists of 2Δ rounds, i.e. each epoch is long enough for honest processes to perform at least one network round-trip during a period of synchrony. In each epoch e, a well-defined and publicly known process is elected as the *leader* who will be in charge of proposing a block. A valid *blockchain* is an ordered sequence of blocks each containing a *parent hash* that encodes the hash of the parent chain, an *epoch number* denoted e, and an arbitrary *payload* string (e.g., a set of transactions). In our protocol, processes will vote on blocks proposed by the leader of each epoch. Henceforth, a collection of signatures on some block from at least $2n/3$ distinct processes is called a *notarization* for the block. A notarized block is a block accompanied by a notarization. If every block is notarized in some chain, we say that the chain is notarized.

Notably, blocks are tagged with the epoch number e, so that at most one block can be notarized (in honest view) per epoch. This fact is crucial to the design and safety of the protocol.

The Streamlet protocol in a nutshell. In each epoch $e = 1, 2, \ldots,$, the processes perform the following where all messages are signed by the sender:

- *Propose.* The leader of epoch e identifies the longest notarized chain observed so far, and proposes a new block extending from this parent chain.
- *Vote.* A process votes on the first proposal received from the current epoch's leader iff 1) it has observed a notarized parent chain consistent with the parent hash declared in the proposed block; 2) it has not yet observed any block notarized at the same *height* (henceforth a block's position in a blockchain is said to be its height).

Finality rule. At any time, if a notarized chain has been observed in which the last three blocks have consecutive epochs, then the prefix of the chain, chopping off the last block, is considered *final*.

In Section 3 we present our protocol more formally, and we show a simple consistency and liveness proof for a partially synchronous network with $< 1/3$ corruptions. In Section 4, we make a couple minor tweaks to the above protocol and prove it secure under honest majority in a synchronous network[1]. The tweaks include 1) requiring $\geq n/2$ signatures for notarization; 2) looking for 6 consecutive epochs in a notarized chain and chopping off 5 for finalization; further, at finalization time, checking to make sure that the 6 blocks at consecutive epochs do not have conflicting notarizations at the same heights.

[1]Note that due to a well-known lower bound by Dwork et al. [8], for $1/3$ or more corruptions, network synchrony assumptions are necessary for reaching consensus.

For both our partially synchronous and synchronous protocols, we will show that transactions get confirmed in expected $O(1)$ rounds under random leader election. In the appendix, we show tweaks to our protocol that yield better constants.

1.4 Comparison with Existing Works

As mentioned, the streamlined blockchain idea was born out of the joint efforts of the community, but knowledge of this approach seems very much scattered, which strongly calls for a unified textbook scheme. In comparison with earlier works, we believe that our protocols are the simplest and most natural of them all — we therefore recommend it for pedagogical purposes and for practical implementation. A major reason we can achieve simplicity is that we exploit the full power of "streamlining":

- *Single type of vote.* Whereas earlier works including classical approaches (e.g., PBFT [3], Paxos [15]) and even very recent, more streamlined approaches (e.g., Hotstuff [1,2]) required multiple rounds of voting or multiple types of votes (e.g., in PBFT, the multiple rounds of voting are called `prepare` and `commit` respectively), we get away with a single type of vote. In some (informal) sense, we are piggybacking the later rounds of voting onto the subsequent blocks' notarizations; and notarizations on subsequent blocks amplify the common belief of ancestor blocks.

- *A unified "propose-vote" paradigm.* Classical approaches (e.g., PBFT [3], Paxos [15]) work by combining a normal path and a (often complicated) view change sub-protocol. Although more recent works such as Hotstuff [1,2] have made the view change subprotocol very simple, it did not appear to be a primary goal of these works to streamline the blockchain protocol to the full potential and completely remove the need for a separate view change path.

  In a sense, STREAMLET builds upon the core techniques these earlier works (e.g., Hotstuff [1, 2]) leveraged to simplify the view change, but we streamline even the view change away and fit the entire protocol under a unified "propose-vote" paradigm — this is accomplished with the help of a new and natural finalization rule that works very much like the way Bitcoin's Nakamoto consensus chops off trailing blocks for finalization (however our finality guarantees are deterministic assuming ideal signatures and hash functions whereas Bitcoin's finality guarantees are probabilistic).

Closest related works. Our protocol is closest in nature to the ones described in Shi's tutorial [21], which in turn are inspired by the recent works Pili [5] and Pala [4]. To the best of our knowledge, these [4,5,21] are the only known works (offering full consistency and liveness proofs) where the authors aim to fit the entire consensus protocol under a unified "propose-vote" paradigm (we may call this paradigm "full streamlining"). The STREAMLET family of protocols are immediately inspired by their full-streamlining paradigm, but our approach is even simpler. Specifically, the prior works [4,5,21] use a freshness/staleness trick where voters must check that the proposed block is extending from a parent chain that is "not too stale" by some technical definition of "staleness" before voting on the block. The definition of staleness is the most subtle part of the prior protocols [4,5,21] and also requires some care in implementation.

In our STREAMLET protocol family, we completely remove the notion of freshness/staleness. Instead, the staleness check during voting is replaced with a natural rule that is easy to implement: a voter now checks that there is no conflicting block notarized at the same height before voting.

Finally, we point out that the first streamlined blockchain is in fact implicitly embodied in the Casper protocol by Buterin and Griffith [22], which Ethereum plans to use as a finality gadget on top of their PoW chain. Casper in fact also leverages a full-streamlining idea although their work did not provide a full liveness proof and their (unproven) liveness mechanism is somewhat coupled with the underlying PoW's block proposal process.

Non-goals. We stress that micro-optimizing the concrete constants related to confirmation time is not a goal of our paper. In this paper, we instead take simplicity and understandability to be first-class principles.

In our protocols, transactions get confirmed in 4Δ rounds during a period of synchrony in the partially synchronous protocol in the optimistic case, and in 12Δ rounds for the synchronous protocol. It is in fact easy to reduce the constants further by introducing additional artifacts into the protocol description. Indeed, some earlier works (e.g., Sync-Hotstuff [2]) focused on the somewhat opposite goal of getting the constants to be the smallest possible at the price of complicating the protocol description — for example, Sync-Hotstuff claims 2Δ confirmation latency but they need to retain a bi-modal approach with a normal path and a separate view change. To illustrate this point, in our appendices, we show some ways to improve the confirmation time but at the price of introducing more types of votes or more synchronization points during the protocol — we stress that even the protocols in our appendices are notably simple in comparison with related works and they only have one mode of operation rather than two.

2 Preliminaries

2.1 Execution Model

We consider the following execution environment. There are in total n processes, and all processes have a public key that is common knowledge. Each process retains its own secret key for signing messages.

The adversary may control a subset of the processes which are said to be *corrupt*, and the remaining processes not under adversarial control are *honest*. Honest processes always faithfully follow the prescribed protocol but corrupt processes can deviate from the prescribed protocol arbitrarily (often called Byzantine faults). W assume that the adversary chooses which processes to corrupt prior to the execution, i.e., we consider the *static* corruption model.

A protocol's execution proceeds in *rounds*, which we use to denote a basic unit of time. The adversary can arbitrarily delay messages sent by honest processes. Obviously one cannot hope to achieve liveness when message delays are arbitrary, we therefore model "periods of synchrony" using the standard Global Stabilization Time (GST) approach [8]. Roughly speaking, before the GST, message delays can be arbitrary; however, after the GST, messages sent by honest processes are guaranteed to be received by honest recipients within Δ number of rounds. More precisely, we assume the following:

> Δ-*bounded assumption during periods of synchrony:* When an honest process sends a message in round r, an honest recipient is guaranteed to receive it at the beginning of round $\max(GST, r + \Delta)$.

Partial synchrony vs synchrony. We assume that a protocol is always informed of (i.e., parametrized with) the maximum network delay Δ during periods of synchrony (i.e., after GST)[2]. The main difference between partial synchrony and synchrony is the following:

- A *partially synchronous* protocol does not know when GST will take place, and is required to ensure consistency regardless; however, liveness is only required after the GST.
- A *synchronous* protocol is essentially promised that GST will start from the beginning of the execution, i.e., the network must satisfy the aforementioned Δ-bounded assumption.

By a famous lower bound proven by Dwork et al. [8], one cannot tolerate $1/3$ or more corruptions in a partially synchronous environment and therefore the $1/3$ protocol we present achieves optimal resilience.

Message echo assumption. Throughout our paper, we assume that when an honest process receives a fresh, previously unseen message (or input from the environment), it immediately multicasts the message to all other processes. To keep the presentation concise, we do not repeat this implicit echoing in our protocol descriptions. With such implicit message echoing, we can in fact strengthen the above Δ-bounded assumption to the following:

> *Strong Δ-bounded assumption during periods of synchrony:* if an honest process *observes* a message in round r, then all honest processes will have observed it at the beginning of round $\max(GST, r + \Delta)$.

2.2 Blockchain Protocols

Processes participating in a blockchain protocol receive inputs (e.g. transactions) from an external environment. The processes are tasked to maintain an ordered log, called a *blockchain*, containing a sequence of strings called *blocks* (note that a blockchain protocol is allowed to define additional validity rules for its blockchain data structure). At any point of time, a process's output is the blockchain it maintains. Henceforth if a process p outputs some blockchain ch at some time, we also say that p considers ch *final*, or that ch is the *finalized* chain for process p.

Without loss of generality, we may assume that the protocol guarantees that a process p's output chain never decreases in length — given any protocol Π in which a process p's output chain may decrease in length, we can always compile it to a modified protocol Π' in which if any process p's output is ever going to shrink in length in Π, p would simply stick to the present, longer output.

We require that a blockchain protocol satisfies consistency and liveness as defined below with all but negligible (in some security parameter) probability over the choice of the random execution.

- *Consistency.* If two blockchains ch and ch' are ever considered final by two honest processes, it must be that either $\mathsf{ch} \preceq \mathsf{ch}'$ or $\mathsf{ch}' \preceq \mathsf{ch}$ where "$\preceq$" means "is a prefix of or equal to".
- *Liveness.* If some honest process receives some input txs in round r, txs will eventually be included in all honest processes' finalized chains.

[2] Dwork et al. [8] also propose another model of partial synchrony where Δ is unknown. The two models of partial synchrony are equivalent from a feasibility perspective [8]. Protocols in the unknown-Δ model typically try different guesses of Δ, doubling the guess each time until correct. Typically, the unknown-Δ model is more of theoretical interest, since this repeated trying technique makes the resulting protocols more complex and less practical.

3 A Blockchain Secure Against $< 1/3$ Corruptions

3.1 Definitions and Notations

Throughout, we use the notation $H : \{0,1\}^* \to \{0,1\}^\kappa$ to denote a hash function selected at random from a collision-resistant hash family.

Valid block and blockchain. A valid block is a tuple of the form $(d_{-1}, e, \mathsf{txs})$ where $d_{-1} \in \{0,1\}^\kappa$ is called the *parent hash*, $e \in \mathbb{N}$ is called the epoch number, and $\mathsf{txs} \in \{0,1\}^*$ is an application-specific payload string such as a set of transactions. A valid blockchain (or chain for short), often denoted $b_0 b_1 \ldots b_h$, is an ordered sequence of blocks such that[3]

- b_0 is a canonical "genesis" block of the form $(\perp, 0, \perp)$; and
- for every $i \in [h]$, b_i is a valid block and moreover $b_i.d_{-1} = H(b_0 b_1 \ldots b_{i-1})$. (Again, $b_i.d_{-1}$ refers to block b_i's parent hash value.) i.e., every block b_i's parent hash value must be a correct hash of the parent chain $b_0 b_1 \ldots b_{i-1}$.

The index of the block in a valid blockchain is called the *height*. Throughout the paper, we will use the notation b_h to denote a block at height h in some valid blockchain.

Notarization. Given a valid block b, a collection of valid signatures on the tuple $\langle \mathsf{notarize}, b \rangle$ from at least $2n/3$ processes is called a *notarization for* b. We assume that the globally unique genesis block b_0 is always notarized (even without any signatures).

Notations. We will make use of the following helpful notations.

- *Notarization and notarized chain.* Henceforth we will use the notation $[b]$ to denote the block b along with an arbitrary valid notarization for b (note that the notarization may not be unique).

  Given a valid blockchain $b_0 b_1 \ldots b_h$, we use the notation $[b_0][b_1] \ldots [b_h]$ to denote the same blockchain along with an arbitrary valid notarization for every block. Such a blockchain is also said to be a *notarized chain*.

- *Signature notation.* We will use the notation $\langle \mathsf{m} \rangle_p$ to denote a message m tagged with a valid signature by process p on m.

3.2 Protocol

Our protocol proceeds in incrementing epochs where each epoch is 2Δ rounds, i.e., long enough for honest processes to perform a network round-trip during a period of synchrony.

Each epoch e has a unique *leader* is elected according to some publicly known function. For the time being, we can imagine that epoch e's leader, henceforth denoted ℓ_e, is defined as as $\ell_e := H^*(e) \bmod n$, where H^* is a random oracle used for leader election, and we assume that H^* is randomly chosen after the adversary decides which processes to corrupt.

Our partially synchronous, 1/3-protocol is presented below — we also refer the reader to Section 1 for a slightly more narrative-style description of the same protocol.

[3]Although we do not require the epochs contained in a valid blockchain to be strictly increasing, it is not hard to show that indeed in any *notarized* chain, this constraint must hold.

Partially synchronous Streamlet blockchain secure against $< 1/3$ corruptions

Notarization for a block b. A collection of signatures on $\langle \mathsf{notarize}, b \rangle$ from at least $2n/3$ distinct processes.

For every epoch $e = 1, 2, \ldots$, a process p performs the following actions:

- *Propose.* If $p = \ell_e$, i.e., if p is the leader of the epoch:
  1. Let $[b_0][b_1] \ldots [b_{h-1}]$ be longest notarized chain seen by p (if there exist multiple, choose any.) Let txs be all transactions observed but not yet included in its finalized chain, and let $b_h = (H(b_0 \ldots b_{h-1}), e, \mathsf{txs})$.
  2. Multicast the proposal $\langle \mathsf{propose}, b_h \rangle_{\ell_e}$.
- *Vote.* On seeing the *first* valid proposal $\langle \mathsf{propose}, (d_{-1}, e, \mathsf{txs}) \rangle_{\ell_e}$ multicast $\langle \mathsf{notarize}, b_h \rangle_p$ iff the following hold:
  1. p has seen a notarized parent chain $[b_0] \ldots [b_{h-1}]$ such that $d_{-1} = H(b_0 \ldots b_{h-1})$;
  2. p has not seen a conflicting notarization for a conflicting block $b'_h \neq b_h$ at the same height h;

Finality rule. At any time, upon observing a notarized chain $[b_0][b_1] \ldots [b_{h-2}]\ [b_{h-1}][b_h]$ such that the last 3 blocks $b_{h-2}b_{h-1}b_h$ have consecutive epoch numbers, then prefix $b_0 b_1 \ldots b_{h-1}$ is considered *final*. A process always outputs the longest final chain it has observed so far.

3.3 Proofs

Additional terminology and conventions. We say that a message m is *in honest view* in some execution iff some honest process ever observes it at some point during the execution. Similarly, "a block is notarized (or finalized resp.) in honest view" in some execution iff some honest process ever observes a notarization (or finalization resp.) for the block at some point during the execution.

In our proofs, we consider only executions in which no signature forgery or hash collision happen. All the lemmas and theorems below hold for executions where these two bad events do not happen. By the security of the signature scheme and the collision resistance of the hash family, all the lemmas and theorems below hold except with negligible (in the security parameter) probability over the choice of the execution.

Fact 1. *Suppose that some block b_h is notarized or finalized in honest view. It must be that there exists one and only one blockchain in honest view $b_0 b_1 \ldots b_h$ that ends at b_h, and moreover, every block b_i in this chain for $i \in [0..h]$ must be notarized in honest view.*

Proof. The uniqueness of a blockchain ending at b_h stems from our assumption of no hash collision (or alternatively, ignoring the negligible fraction of executions in which hash collisions happen).

Now, if some prefix $b_0 b_1 \ldots b_i$ where $i < h$ is not notarized in honest view, then no honest process could have signed the next block b_{i+1}, and thus b_{i+1} cannot be notarized in honest view either. Therefore, through a simple induction, we can show that every block b_i in this chain where $i \in [0..h]$ must be notarized in honest view. □

3.3.1 Consistency Proof

We now give a consistency proof. It is important to observe that in the entire consistency proof, we never rely on network synchrony assumptions. For this reason, our 1/3 protocol can ensure consistency even when network delays can be arbitrarily long.

Fact 2. *If blocks b and b' both of epoch e are notarized in honest view, it must be that $b = b'$.*

Proof. Let $f < n/3$ be the number of corrupt processes. Let S be the set of at least $2n/3 - f$ honest processes that signed b in epoch e and let S' be the set of at least $2n/3 - f$ honest processes that signed b' in epoch e. The intersection of these two sets (of honest processes) is $|S \cap S'| \geq (2n/3 - f) + (2n/3 - f) - (n - f) = (n/3 - f) > 0$. $|S \cap S'| > 0$ implies that at least one honest process must have signed both b and b' in epoch e. This is a contradiction, since by the protocol definition, an honest process signs at most one unique block per epoch. □

Lemma 1. *Suppose that a notarized chain in honest view contains three consecutive blocks b_h, b_{h+1}, b_{h+2}, with consecutive epoch numbers $e, e+1$, and $e+2$ respectively. Then there cannot exist a conflicting block $b'_{h+1} \neq b_{h+1}$ such that b'_{h+1} is notarized in honest view.*

Proof. Assume for contradiction that $b'_{h+1} \neq b_{h+1}$ is notarized in honest view. Let e' denote the epoch number of b'_{h+1}. Then either $e' < e$, or $e' > e+2$, by Facts 1 and 2. In both cases we derive a contradiction — in the following let $f < n/3$ be the number of corrupt processes:

1. Let $e' < e$. At least $2n/3 - f$ honest processes signed the tuple $\langle \mathsf{notarize}, b'_{h+1} \rangle$ in epoch e'; let S denote this set. This implies that by the end of epoch e', every process in S observed a notarized parent chain ending at height h. By the protocol definition, no honest process in S will vote for any block whose height is $\leq h$ after the end of epoch e'. But since $e > e'$, this implies that b_h cannot gain notarization in honest view in epoch e, since at most $n - (2n/3 - f) = n/3 + f < 2n/3$ processes can vote for it. This in turn implies that b_{h+1} will not be notarized in honest view, because honest processes only vote for blocks with notarized parent blocks. This leads to a contradiction.

2. Let $e' > e + 2$. Since b_{h+2} is notarized in honest view, it must be that at least $2n/3 - f$ honest processes signed b_{h+2} in epoch $e + 2$; let S denote this set of honest processes. This implies that every process in S must have observed a notarized parent chain ending at b_{h+1} by the end of epoch $e + 2$. By our protocol definition, no process in S will vote for any competing block whose height is $\leq h + 1$ in epoch $e' > e + 2$. So again strictly fewer than $n - (2n/3 - f) = n/3 + f < 2n/3$ processes can vote for b'_{h+1}, which leads to a contradiction.

□

Theorem 2 (Consistency)**.** *If $\mathsf{ch} := [b_0][b_1] \ldots [b_{h-2}][b_{h-1}][b_h]$ and $\mathsf{ch}' := [b_0][b'_1] \ldots [b'_{k-2}][b'_{k-1}][b'_k]$ are two notarized chains in honest view and both chains end at three blocks with consecutive epochs — without loss of generality, assume that $h \leq k$. Then, it must be that $b_0 b_1 \ldots b_{h-1}$ is either a prefix of or equal to $b_0 b'_1 \ldots b'_{k-1}$.*

Proof. Let the epochs of b_{h-2}, b_{h-1} and b_h be denoted $e, e+1$, and $e+2$ respectively.

For the sake of contradiction, suppose that $b_0 b_1 \ldots b_{h-1}$ is not a prefix of or equal to $b_0 b'_1 \ldots b'_{k-1}$. This means that $b'_{h-1} \neq b_{h-1}$, due to Fact 1. By Lemma 1, since b_{h-2}, b_{h-1}, b_h are each notarized and have consecutive epochs, the conflicting notarized block $b'_{h-1} \neq b_{h-1}$ cannot exist in honest view, which is a contradiction.

□

3.3.2 Liveness Proof

We now present a liveness proof. Obviously when the network is partitioned and delays can be arbitrarily long, one cannot hope to have liveness. Therefore, our liveness guarantees hold after the GST, i.e., when the network eventually becomes synchronous. It can be easily seen that our liveness proofs do need to rely on the strong Δ-bounded network assumption after the GST.

Fact 3. *Suppose that some honest process p has observed a notarization for block b_h at the beginning of round r. It must be that at the beginning of round $\max(GST, r + \Delta)$, every honest process has seen a notarized chain $[b_0][b_1] \ldots [b_{h-1}][b_h]$ ending at the block b_h.*

Proof. Some honest process p' must have signed $\langle \mathsf{notarize}, b_h \rangle$ in or before round r, and p' must have observed the notarized prefix $[b_0][b_1] \ldots [b_{h-1}]$ by the beginning of round r.

By our strong Δ-bounded assumption during periods of synchrony (see Section 2.1), all honest processes will have observed the notarized prefix $[b_0][b_1] \ldots [b_{h-1}]$ and a notarization for b_h by $\max(GST, r + \Delta)$. □

Fact 4. *Suppose that after the GST, there are two epochs e and $e + 1$ both with honest leaders denoted ℓ_e and ℓ_{e+1} respectively, and suppose that ℓ_e and ℓ_{e+1} each proposes a block at height h_0 and h_1 respectively in epochs e and $e + 1$ respectively, it must be that $h_1 \geq h_0 + 1$.*

Proof. At the beginning of the round[4] Δ into epoch e, every honest process will have observed ℓ_e's proposal of epoch e for the block b_{h_0}. Moreover, due to the strong Δ-bounded assumption during periods of synchrony, the notarized parent chain $[b_0] \ldots [b_{h_0-1}]$ that triggered ℓ_e to propose b_{h_0} must have been observed by every honest process at the beginning of the round Δ into epoch e.

Every honest process will either sign $\mathsf{notarize}$ for this proposal unless by the beginning of round Δ into epoch e, it has already observed a conflicting notarization for some $b'_{h_0} \neq b_{h_0}$ at the same height. Now there are two cases:

1. If at least one honest process, say p, has observed a conflicting notarization for some $b'_{h_0} \neq b_{h_0}$ by round Δ of epoch e, then due to Fact 3, by the beginning of epoch $e + 1$, every honest process must have seen a notarized chain ending at height h_0.

2. If no honest process has signed and multicast saw a conflicting notarization at height h_0 by round Δ of the epoch, then all honest processes will sign ℓ_e's proposal in or before round Δ of the epoch e. Thus by the beginning of epoch $e + 1$, every honest process will have observed the notarized chain $[b_0] \ldots [b_{h-1}][b_h]$.

Therefore, ℓ_{e+1} must propose a block at height $h_0 + 1$ or greater. □

Lemma 3 (Main liveness lemma)**.** *After GST, suppose there are three consecutive epochs $(e, e + 1, e + 2)$ all with honest leaders denoted ℓ_e, ℓ_{e+1}, and ℓ_{e+2}, then the following hold — below we use b_{h_2} to denote the block proposed by ℓ_{e+2} during epoch $e + 2$:*

[4]The initial round of epoch is numbered round 0.

a) by the beginning of epoch $e + 3$, every honest process will observe a notarized chain ending at b_{h_2} (and b_{h_2} was not notarized before the beginning of epoch e);

b) furthermore, no conflicting block $b'_{h_2} \neq b_{h_2}$ at the same height will ever get notarized in honest view.

Proof. Let h_0, h_1, h_2 be the respective heights of the proposals made by ℓ_e, ℓ_{e+1}, and ℓ_{e+2} in epochs $e, e + 1$, and $e + 2$ respectively. By Fact 4, $h_2 > h_1 > h_0$. Let b_{h_2} be the block proposed by ℓ_{e+2} in epoch $e + 2$. We now argue that by the beginning of epoch $e + 3$, no honest process signed $\langle \mathsf{notarize}, b'_{h_2} \rangle$ for any conflicting $b'_{h_2} \neq b_{h_2}$. Suppose that this is not true and some honest process p^* signed $\langle \mathsf{notarize}, b'_{h_2} \rangle$ a conflicting $b'_{h_2} \neq b_{h_2}$ by the beginning of epoch $e + 3$. Now, p^* cannot have signed $\langle \mathsf{notarize}, b_{h_2} \rangle$ during epochs e, $e + 1$, or $e + 2$ since ℓ_e and ℓ_{e+1} cannot have proposed a block at height h_2 in epochs e and $e + 1$ respectively by Fact 4, and ℓ_{e+2} proposed $b_{h_2} \neq b'_{h_2}$ in epoch $e + 2$

Therefore, the honest process p^* must have signed $\langle \mathsf{notarize}, b_{h_2} \rangle$ before epoch e started, and at this time p^* must have observed a notarized parent chain ending at height $h_2 - 1$. Due to our strong Δ-bounded network assumption during periods of synchrony, this notarized parent chain ending at height $h_2 - 1$ must have been observed by all honest processes by the beginning of epoch $e + 1$. Therefore, ℓ_{e+1} must propose a block at height at least h_2. Thus we have reached a contradiction.

Since by the beginning of epoch $e + 3$, no honest process has signed $\mathsf{notarize}$ for any $b'_{h_2} \neq b_{h_2}$, at this time there cannot be a notarization for any $b'_{h_2} \neq b_{h_2}$ in honest view. Moreover, ℓ_{h+2}'s proposal and the notarized parent chain that triggered the proposal will be observed by all honest processes at the beginning of round Δ into epoch $e + 2$. Therefore all honest processes will sign and multicast $\langle \mathsf{notarize}, b_{h_2} \rangle$ in or before round Δ into epoch $e + 2$. Thus by the beginning of epoch $e + 3$, all honest processes will have seen a notarization for b_{h_2}. Thus, no honest process will ever sign $\mathsf{notarize}$ for any conflicting $b'_{h_2} \neq b_{h_2}$ after the start of epoch $e + 3$ either; and any conflicting $b'_{h_2} \neq b_{h_2}$ cannot ever gain notarization in honest view. □

Theorem 4 (Liveness)**.** *After GST, suppose that there are 5 consecutive epochs $e, e+1, \ldots, e+4$ all with honest leaders, then by the beginning of epoch $e + 5$, every honest process must have observed a new final block that was not final at the beginning of epoch e.*

Note that every time a new block proposed by an honest leader becomes final, this creates an opportunity for outstanding transactions to be incorporated into honest processes' finalized chain — specifically, in our protocol, an honest process always proposes a new block containing all outstanding transactions it has seen.

Proof. Due to Lemma 3, by the beginning of epoch $e + 5$, the blocks proposed by ℓ_{e+2}, ℓ_{e+3}, and ℓ_{e+4}, henceforth denoted b_{h_2}, b_{h_3}, and b_{h_4}, will be notarized in every honest process's local view. Moreover, these blocks cannot have been notarized in honest view before the beginning of epoch e, and no conflicting blocks can ever be notarized in honest view at heights h_2, h_3, and h_4.

We now show that b_{h_3} must extend from the parent chain ending at b_{h_2} and similarly b_{h_4} must extend from the parent chain ending at b_{h_3}. To see this, notice that by Lemma 3, at the beginning of epoch $e + 3$, the notarized parent chain ending at b_{h_2} must have been observed by all honest processes and no honest process can have observed any notarized chain that is longer, or any *different* notarized chain of the same length. Thus ℓ_{e+3}'s proposal must extend from the chain ending at b_{h_2}. Similarly ℓ_{e+4}'s proposal must extend from the chain ending at b_{h_3}.

Now, by the finality rule, at the beginning of epoch $e + 5$, the block proposed by ℓ_{e+3} will be considered final by every honest process. $\square$

Remark 1 (Leader election policies and confirmation time)**.** Our earlier presentation considered a random leader election policy where each epoch's leader is $H^*(e) \mod n$ where H^* is a random oracle. With such a random policy, assuming that fewer than $n/3$ processes are corrupt, it takes in expectation $O(1)$ number of rounds before we can have 5 consecutive epochs all with honest leaders. Alternatively, we may consider a policy where each randomly elected process serves as the leader for 5 consecutive epochs to improve the confirmation time by a constant when up to $< n/3$ processes can be corrupt. Note that in the optimistic case when all processes are honest, our protocol can confirm transactions in merely 2 epochs.

4 A Blockchain Secure Against Minority Corruptions

We now show that by making a couple minor tweaks to the protocol described earlier, we can obtain an honest-majority protocol that is secure in a synchronous network.

4.1 Protocol

The new protocol is described below with the modifications highlighted in red. Unless otherwise stated, in this section, all definitions and notations (including the definition of valid blocks and valid blockchains, leader election, etc.) are the same[5] as in Section 3.

[5]Although we do not impose any restrictions on the epoch numbers in a valid blockchain, it is not hard to show that for our synchronous honest-majority protocol, in any *notarized* chain, the epochs must be *non-decreasing*.

Synchronous Streamlet blockchain secure against minority corruptions

Notarization for a block b. A collection of signatures on $\langle \mathsf{notarize}, b\rangle$ from at least $n/2$ distinct processes.

For every epoch $e = 1, 2, \ldots$, a process p performs the following actions:

- *Propose.* If $p = \ell_e$, i.e., if p is the leader of the epoch:
  1. Let $[b_0][b_1]\ldots[b_{h-1}]$ be longest notarized chain seen by p (if there exist multiple, choose any.) Let txs be all transactions observed but not yet included in its finalized chain, and let $b_h = (H(b_0 \ldots b_{h-1}), e, \mathsf{txs})$.
  2. Multicast the proposal $\langle \mathsf{propose}, e, b_h\rangle_{\ell_e}$.
- *Vote.* On seeing the *first* valid proposal $\langle \mathsf{propose}, (d_{-1}, e, \mathsf{txs})\rangle_{\ell_e}$ multicast $\langle \mathsf{notarize}, b_h\rangle_p$ iff the following hold:
  1. p has seen a notarized parent chain $[b_0]\ldots[b_{h-1}]$ such that $d_{-1} = H(b_0 \ldots b_{h-1})$;
  2. p has not seen a notarization for a conflicting block $b'_h \neq b_h$ at the same height h;

Finality rule. At any time, upon observing a notarized chain $[b_0][b_1]\ldots[b_h]$ such that the last 6 blocks $b_{h-5}\ldots b_h$ have consecutive epoch numbers, and moreover no notarizations have been observed for any conflicting block at height $h' \in [h-5, h]$, then, the prefix $b_0 b_1 \ldots b_{h-5}$ is considered *final*. A process always outputs the longest final chain it has observed so far.

Remark 2 (A variant: 4 consecutive blocks, chop off trailing 3)**.** In the synchronous Streamlet blockchain protocol described above, we need to wait for 6 blocks with consecutive epoch numbers and chop off the trailing 5 for finalization.

We note that there are easy ways to tighten this constant 6; however, our philosophy in this paper is to convey a protocol that is easiest to understand rather than tightening this constant to the best it can be (e.g., by adding a little complexity to the protocol description). We believe that simple protocols should facilitate optimizations. To elaborate on this point, we point out a simple tweak to our above protocol allows for finalization when the last 4 blocks have consecutive epoch numbers, finalizing the first of those 4 blocks (if no competing notarizations were observed for any of those 4 blocks).

The tweak is as follows: insist that propose and vote are sent at the beginning of the first and second super-rounds of the epoch, respectively where each super-round is a span of Δ rounds. Now, it is not hard to show a tighter version of Fact 5 for this modified protocol, that is if an honest node votes for some block in epoch e, then all honest nodes will observe it by the beginning of the "Vote" super-round of epoch $e+1$ (instead of the current $e+2$). In this way, the liveness theorem would require only 6 consecutive epochs with honest leaders for progress to ensue.

4.2 Proofs

It is not hard to show that Fact 1 still holds for our synchronous, honest-majority protocol.

4.2.1 Consistency Proof

Fact 5. *Suppose that some block b_h of epoch e is notarized in honest view, it must be that by the beginning of epoch $e+2$, every honest process has observed a notarized chain $[b_0]\dots[b_{h-1}]$ that is a valid parent of b_h.*

Proof. Since b_h is notarized in honest view, at least one honest process p must have signed b_h in epoch e and p must have observed a notarized chain $[b_0]\dots[b_{h-1}]$ that is a valid parent for b_h in epoch e. By our strong Δ-bounded network assumption (see Section 2.1), every honest process will have observed $[b_0]\dots[b_{h-1}]$ by the beginning of epoch $e+2$. □

Lemma 5. *Suppose that there is a notarized chain in honest view containing two adjacent blocks $b_h b_{h+1}$ at consecutive epoch numbers e and $e+1$ respectively, then, no block of height h and epoch greater than $e+2$ can be notarized in honest view.*

Proof. By Fact 5, every honest process will have observed a notarized chain $[b_0][b_1]\dots[b_h]$ that is a valid parent of b_{h+1} by the beginning of epoch $e+3$. This means that no honest process will sign any block of height h or smaller in epoch $e+3$ or greater. Thus no block at height h and of epoch greater than $e+2$ can be notarized in honest view. □

Theorem 6 (Consistency)**.** *Suppose that some honest process finalizes $b_0 b_1 \dots b_{h-5}$ by applying the finality rule to the notarized chain $[b_0][b_1]\dots[b_h]$, and some honest process finalizes $b_0 b'_1 \dots b'_{k-5}$ by applying the finality rule to the notarized chain $[b_0][b'_1]\dots[b'_k]$. Without loss of generality, assume that $k \geq h$. It must be that $b_0 b_1 \dots b_{h-5}$ is a prefix of or equal to $b_0 b'_1 \dots b'_{k-5}$.*

Proof. Suppose that this is not true, by Fact 1, it must be that $b'_{h-5} \neq b_{h-5}$ and $b'_{h-4} \neq b_{h-4}$. Let $e-5, e-4, \dots, e$ be the epochs of $b_{h-5}, b_{h-4}, \dots, b_h$ respectively. By Lemma 5, b'_{h-4} must be of epoch $e' \leq e-2$. By Fact 5, by the beginning of epoch $e'+2 \leq e$, every honest process will have observed a notarized parent chain $[b_0][b'_1]\dots[b'_{h-5}]$. Obviously b_h cannot gain notarization in honest view before epoch e. This means that when any honest process observes a notarization for b_h, it must also have observed a notarization for b'_{h-5}. Thus no honest process will successfully apply the finalization rule to $[b_0][b_1]\dots[b_h]$ due to this conflicting notarization for b'_{h-5} and this leads to a contradiction. □

4.2.2 Liveness Proof

Fact 3, Fact 4, and Lemma 3 of Section 3.3.2 still hold (with identical proofs) for our new synchronous, honest-majority scheme — with the difference that now, GST starts at the beginning of the execution, i.e., the network synchrony assumption holds throughout.

Theorem 7 (Liveness)**.** *Suppose that there are 8 consecutive epochs $e, e+1, \dots, e+7$ all with honest leaders, then by the beginning of epoch $e+7$, every honest process must have observed a new final block that was not final at the beginning of epoch e.*

Proof. The proof follows in the same way as that of Theorem 4, additionally observing that there cannot ever be any conflicting notarizations in honest view at the same heights as the blocks proposed by ℓ_{e+2}, $\ell_{e+3}, \dots, \ell_{e+7}$ due to Lemma 3. □

References

[1] Ittai Abraham, Guy Gueta, and Dahlia Malkhi. Hot-stuff the linear, optimal-resilience, one-message BFT devil. *CoRR*, abs/1803.05069, 2018.

[2] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync hotstuff: Simple and practical synchronous state machine replication. Cryptology ePrint Archive, Report 2019/270, 2019. `https://eprint.iacr.org/2019/270`.

[3] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, 1999.

[4] T-H. Hubert Chan, Rafael Pass, and Elaine Shi. Pala: A simple partially synchronous blockchain. Manuscript, 2018. `https://eprint.iacr.org/2018/981`.

[5] T-H. Hubert Chan, Rafael Pass, and Elaine Shi. Pili: A simple, fast, and robust family of blockchain protocols. Cryptology ePrint Archive, Report 2018/980, 2018. `https://eprint.iacr.org/2018/980`.

[6] Jing Chen and Silvio Micali. Algorand: The efficient and democratic ledger. https://arxiv.org/abs/1607.01341, 2016.

[7] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016.

[8] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 1988.

[9] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Eurocrypt*, 2015.

[10] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 bft protocols. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pages 363–376, New York, NY, USA, 2010. ACM.

[11] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series: Consensus system. `https://dfinity.org/tech`.

[12] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Crypto*, 2017.

[13] Ramakrishna Kotla, Lorenzo Alvisi, Michael Dahlin, Allen Clement, and Edmund L. Wong. Zyzzyva: speculative byzantine fault tolerance. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, October 14-17, 2007*, pages 45–58, 2007.

[14] Jae Kwon. Tendermint: Consensus without mining. `http://tendermint.com/docs/tendermint.pdf`, 2014.

[15] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[16] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[18] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, page 305–320, USA, 2014. USENIX Association.

[19] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Eurocrypt*, 2017.

[20] Rafael Pass and Elaine Shi. Rethinking large-scale consensus. In *CSF*, 2017.

[21] Elaine Shi. Streamlined blockchains: A simple and elegant approach (a tutorial and survey). In *Asiacrypt*, 2019.

[22] Virgil Griffith Vitalik Buterin. Casper the friendly finality gadget. `https://arxiv.org/abs/1710.09437`.

A Another Variant

For the partially synchronous, 1/3-setting, we describe yet another variant, with a simpler consistency proof and better confirmation time. It leverages two types of votes per block but can confirm transactions in 3Δ rounds in the optimistic case (*c.f.*, the scheme in Section 3 confirms transactions in 4Δ rounds - 2 epochs of 2Δ each). The primary difference from the protocol in Section 3 is a new finalization rule: a process votes to finalize a notarized block if it has not notarized any other block at the same height. When at least $2n/3$ finalize-votes have been collected, a block becomes final.

In our new variant, it turns out that the blocks need not be tagged with an epoch number; thus we assume that a valid block is of the format $b := (d_{-1}, \mathsf{txs})$ where d_{-1} is the parent hash and txs is an application-specific payload. Blockchain validity is defined in the same way as before.

A notarize-finalize variant for the partially synchronous setting

The protocol proceeds in incrementing epochs where each epoch is $2\Delta + 1$ rounds.

For every epoch $e = 1, 2, \ldots,$, each process p performs the following:

1. If $p = \ell_e$, the leader of the epoch, multicast $\langle \mathsf{propose}, e, b_h \rangle_{\ell_e}$, choosing b_h as follows:

   let $[b_0][b_1]\ldots[b_{h-1}]$ be the longest notarized chain seen by p (if there exist multiple, choose any.) Choose txs to be all transactions observed but not yet included in p's finalized output chain, and let $b_h = (H(b_0 \ldots b_{h-1}), \mathsf{txs})$.

2. On seeing the *first* valid proposal $\langle \mathsf{propose}, e, b_h \rangle_{\ell_e}$, multicast $\langle \mathsf{notarize}, b_h \rangle_p$ iff

   (a) p has seen a notarized chain $[b_0] \ldots [b_{h-1}]$ such that

$b_h.d_{-1} = H(b_0 \dots b_{h-1})$, and

(b) p did not previously multicast $\langle \mathsf{finalize}, b'_h \rangle_p$ for a conflicting block $b'_h \neq b_h$ at height h.

Finality rule. At any time, upon observing a notarization for a block b_h, process p multicasts $\langle \mathsf{finalize}, b_h \rangle_p$ iff it has not previously multicast $\langle \mathsf{notarize}, b'_h \rangle_p$ for a conflicting block $b'_h \neq b_h$ at the same height. Upon observing at least $2n/3$ valid signatures for $\langle \mathsf{finalize}, b_h \rangle$ from distinct processes, p considers any valid blockchain ending at the block b_h to be *final.* At any time, p outputs the longest final chain seen so far.

Note that in the above protocol, an important invariant for consistency is that honest processes would never sign $\mathsf{notarize}$ for some block b_h and sign $\mathsf{finalize}$ for a differing block $b'_h \neq b_h$ at the same height.

A.1 Consistency Proof

We now present a consistency proof. Again, the consistency proof does not rely on any network synchrony assumption and it holds even when network delays can be arbitrarily long.

For convenience we define a *finalization* for a block b to be a collection of valid signatures on the tuple $\langle \mathsf{finalize}, b \rangle$ from at least $2n/3$ processes. A block is said to be finalized in honest view if there exists a finalization for it in honest view.

Lemma 8. *If there is a finalization for a block b_h at height h in honest view, then there cannot be a notarization for a conflicting block $b'_h \neq b_h$ at the same height in honest view.*

Proof. Let $f < n/3$ be the number of corrupt nodes. Suppose for the sake of contradiction that b_h is finalized in honest view and a conflicting $b'_h \neq b_h$ is notarized in honest view. Now, this means that exist in honest view a set of signatures on $\langle \mathsf{finalize}, b_h \rangle$ from at least $2n/3$ distinct processes — let this set of processes be S; and there exist in honest view a set of signatures on $\langle \mathsf{notarize}, b'_h \rangle$ from at least $2n/3$ distinct processes — let this set of processes be S'. Now, the intersection $|S \cap S'| \geq 2n/3 + 2n/3 - n = n/3$. Since $f < n/3$, it must be that there is an honest process $p \in S \cap S'$; and this honest process p must have signed both $\langle \mathsf{finalize}, b_h \rangle$ and $\langle \mathsf{notarize}, b'_h \rangle$. By the protocol definition, p cannot have signed $\langle \mathsf{notarize}, b'_h \rangle$ either before signing $\langle \mathsf{finalize}, b_h \rangle$ or after signing $\langle \mathsf{finalize}, b_h \rangle$, and thus we have reached a contradition. $\square$

It is not hard to see that Fact 1 of Section 3 still holds.

Theorem 9 (Consistency)**.** *Suppose that two blockchains $\mathsf{ch}_0 := b_0, b_1, \dots, b_{h_0}$ and $\mathsf{ch}_1 := b_0, b'_1, \dots, b'_{h_1}$ are both finalized in honest view and without loss of generality, assume that ch_1 is at least as long as ch_0. Then either $\mathsf{ch}_0 = \mathsf{ch}_1$ or ch_0 is a prefix of ch_1.*

Proof. Note that both $b_{h_0} = b'_{h_1}$ must be notarized by Fact 1. We consider two cases. First, if ch_0 and ch_1 are equal in length, s.t. $h_0 = h_1$, by Lemma 8, then $b_{h_0} = b'_{h_1}$. Second, consider $h_0 < h_1$. Let b'_{h_0} be the block in ch_1 at height h_0. By Fact 1, b'_{h_0} must be notarized in honest view. By Lemma 8, b'_{h_0} must be equal to b_{h_0}. $\square$

A.2 Liveness Proof

We now present a liveness proof. The liveness proof relies on the strong Δ-bounded network assumption after the GST.

Fact 6. *Suppose that some honest process p has observed a notarization or finalization for block b_h at the beginning of round r. It must be that at the beginning of round* $\max(GST, r + \Delta)$*, every honest process has seen a notarized chain $[b_0][b_1] \dots [b_{h-1}][b_h]$ ending at the block b_h.*

Proof. Similar to that of Fact 3. □

Fact 7. *Suppose that after GST, there are two epochs e and $e + 1$ both with honest leaders denoted ℓ_e and ℓ_{e+1} respectively, and suppose that ℓ_e and ℓ_{e+1} each proposes a block at height h_0 and h_1 respectively in epochs e and $e + 1$ respectively. Then $h_1 > h_0$.*

Proof. By the beginning of round Δ into epoch e, every honest process will have observed ℓ_e's proposal b_{h_0}. Moreover, ℓ_e must have observed a notarized parent chain $[b_0] \dots [b_{h_0-1}]$ when constructing the proposal. By Δ-bounded network assumption after the GST, the notarized parent chain will also be observed by every honest process by the beginning of round Δ into epoch e.

Thus every honest process will sign $\langle \mathsf{notarize}, b_{h_0} \rangle$, unless it has previously multicast $\langle \mathsf{finalize}, b'_{h_0} \rangle$ for some $b'_{h_0} \neq b_{h_0}$. Now there are two cases:

1. If at least one honest process, say p, previously multicast $\langle \mathsf{finalize}, b'_{h_0} \rangle$ for some $b'_{h_0} \neq b_{h_0}$ prior to round Δ into epoch e, then p must have observed a valid notarized chain $[b'_0], \dots, [b'_{h_0}]$ when it signed the $\mathsf{finalize}$ message. Now, by the Δ-bounded network assumption, by the beginning of epoch $e + 1$, every honest process must have seen a notarized chain ending at height h_0.

2. If no honest process previously multicast $\langle \mathsf{finalize}, b'_{h_0} \rangle$ for some $b'_{h_0} \neq b_{h_0}$, then all honest processes will sign ℓ_e's proposal by the end of round Δ into epoch e. Thus by the beginning of epoch $e+1$, every honest process will have observed the notarized chain $[b_0] \dots [b_{h_0-1}][b_{h_0}]$.

Therefore, ℓ_{e+1} must propose a block at height $h_0 + 1$ or greater. □

Theorem 10 (Liveness)**.** *After GST, suppose there are three consecutive epochs $(e, e + 1, e + 2)$ all with honest leaders denoted ℓ_e, ℓ_{e+1}, and ℓ_{e+2} respectively, then by the beginning of epoch $e + 3$, every honest process will observe a new finalized block (that was not finalized at the beginning of epoch e) that is proposed during epoch $e + 2$ by ℓ_{e+2}.*

Proof. Let h_0, h_1, h_2 be the respective heights of the proposals made by ℓ_e, ℓ_{e+1}, and ℓ_{e+2} in epochs $e, e + 1$, and $e + 2$ respectively. By Fact 7, $h_2 > h_1 > h_0$. Let b_{h_2} be the block proposed by ℓ_{e+2} in epoch $e + 2$.

We now argue that before the beginning of epoch $e+3$, no honest process signed $\langle \mathsf{notarize}, b'_{h_2} \rangle$ for any conflicting $b'_{h_2} \neq b_{h_2}$. For contradiction, suppose some honest process p^* signed $\langle \mathsf{notarize}, b'_{h_2} \rangle$ a conflicting $b'_{h_2} \neq b_{h_2}$ before the beginning of epoch $e + 3$. Now, p^* cannot have signed $\langle \mathsf{notarize}, b_{h_2} \rangle$ during epochs e, $e + 1$, or $e + 2$, since honest leaders ℓ_e and ℓ_{e+1} cannot have proposed a block at height h_2 in epochs e and $e + 1$ respectively (since $h_2 > h_1 > h_0$) and ℓ_{e+2} proposed $b_{h_2} \neq b'_{h_2}$ in epoch $e+2$ (here we rely on the fact that the proposals are tagged with the corresponding epoch to distinguish from proposals made by the same process when it was leader in the past). Therefore, the honest process p^* must have signed $\langle \mathsf{notarize}, b_{h_2} \rangle$ before the beginning of epoch e, and at this time p^* must have observed a notarized parent chain ending at height $h_2 - 1$. By the Δ-bounded network assumption, this notarized parent chain ending at height $h_2 - 1$ must be observed by all honest processes by the beginning of epoch $e + 1$. Therefore, ℓ_{e+1} must propose a block at height at least h_2, which is a contradiction.

Since no honest process signed $\mathsf{notarize}$ for conflicting block $b'_{h_2} \neq b_{h_2}$ before the beginning of epoch $e+3$, a conflicting notarization at height h_2 cannot exist in honest view and thus no honest process signed $\mathsf{finalize}$ for conflicting block $b'_{h_2} \neq b_{h_2}$ before the beginning of epoch $e+3$. Moreover, by Δ-bounded network assumption, both ℓ_{e+2}'s proposal in epoch $e+2$ and the notarized parent chain of b_{h_2} that triggered ℓ_{e+2} to propose b_{h_2} must be observed by all honest processes by the beginning of round Δ into epoch e. Thus all honest processes will sign and multicast $\langle \mathsf{notarize}, b_{h_2} \rangle$ by the end of round Δ into epoch $e+2$. By the Δ-bounded network assumption every honest process will observe a notarization for b_{h_2} by the beginning of round 2Δ into epoch $e+2$. Since every epoch is $2\Delta+1$ rounds, by the end of epoch $e+2$, all honest processes will sign and multicast $\langle \mathsf{finalize}, b_{h_2} \rangle$, which will eventually be delivered. The proof now follows due to Fact 6. $\square$