Streamlet: Textbook Streamlined Blockchains

Benjamin Y Chan[1] and Elaine Shi[2]

[1]Cornell University - byc@cs.cornell.edu
[2]CMU/Cornell - runting@gmail.com

September 12, 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 called Streamlet for constructing consensus protocols. 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 years of work on consensus partly driven by the cryptocurrency community — in particular, how remarkably simple the new generation of consensus protocols has become in comparison with classical mainstream approaches 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. For over a decade, companies like Google and Facebook have deployed consensus as part of their core infrastructure, to replicate mission-critical services such as Google Wallet and Facebook Credit. Consensus is also 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 protocols as *blockchain protocols*. The name "blockchain" gained popularity with Bitcoin. Before Bitcoin, such consensus protocols were commonly referred to as State Machine Replication protocols [3, 14, 24].

As we all know, by leveraging Proof-of-Work (PoW), Bitcoin's Nakamoto consensus [9, 19, 21, 23] was the first to achieve consensus in a "permissionless" 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 classical roots of consensus, that is, they 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 in time, the elected participants run a permissioned consensus protocol. This paradigm shift towards PoS rekindled the community's interest in classical, permissioned consensus.

In this paper, we focus on how to construct a simple, permissioned consensus protocol called Streamlet. Streamlet realizes the same abstraction as classical landmark protocols such as PBFT [3] and Paxos [14] — however, unlike the classical counterparts, Streamlet is absurdly simple, making it a perfect choice for pedagogy. Considering several decades of work that aimed to simplify the PBFT/Paxos family of protocols (notably, RAFT [20] and other efforts [17, 27]), it might be somewhat surprising at first that such a remarkably simple and natural protocol turns out to be possible.

1.1 Streamlet in a Nutshell and Our Contributions

It is perhaps most instructive to present the protocol upfront. Let us briefly recall the setting: there are n players participating in the consensus protocol, and their public keys are well-known (i.e., the classical "permissioned" setting). We assume that strictly fewer than $n/3$ players can be corrupt, and corrupt players can deviate from the prescribed protocol arbitrarily and maliciously. We assume that players have synchronized clocks[1] and that the protocol proceeds in synchronized *epochs*, each of which is, say, 1 second long (the "1 second" parameter can be replaced with other reasonable choices and we shall discuss how to choose this parameter shortly). Although we assume synchronized clocks, the protocol achieves consistency (i.e., safety) regardless of how long the message delays are or how badly the network might be partitioned.

[1] It is well-known that in a partially synchronous network, as long as players have clocks with bounded drift, it is possible to employ a simple clock synchronization procedure to establish a synchronized clock [8]. In this work, for simplicity, we assume synchronized clocks. The clock synchrony will only be needed for our liveness proof but not consistency.

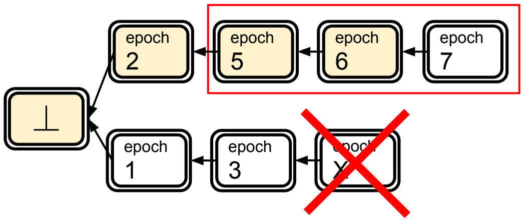

Figure 1: **Streamlet finalization example.** In this example, the prefix of the top chain up to the epoch-6 block is considered final.

The protocol. In STREAMLET, each epoch has a designated leader chosen at random by a publicly known hash function. We assume that a valid blockchain is a sequence of blocks cryptographically "chained" together by a hash function, i.e., each block contains a hash of its prefix. Now, the STREAMLET protocol works as follows:

- **Propose-Vote.** In every epoch:
  - The epoch's designated leader proposes a new block extending from the longest notarized chain it has seen (if there are multiple, break ties arbitrarily). The notion "notarized" is defined below.
  - Every player votes for the first proposal they see from the epoch's leader, as long as the proposed block extends from (one of) the longest notarized chain(s) that the voter has seen. A vote is a signature on the proposed block.
  - When a block gains votes from at least $2n/3$ distinct players, it becomes *notarized*. A chain is *notarized* if its constituent blocks are all notarized.
- **Finalize.** Notarized does not mean final. If in any notarized chain, there are three adjacent blocks with consecutive epoch numbers, the prefix of the chain up to the second of the three blocks is considered *final*. When a block becomes final, all of its prefix must be final too.

Importantly, the entire protocol follows a unified *propose-vote* paradigm, making it very natural and leaving (seemingly) no room for misinterpretation. In comparison, the classical mainstream approach [3, 10, 13, 14, 18, 18, 20] typically adopts a simple fast path that follows a "propose-vote" paradigm, but the fast path provides no liveness in the face of faults. To achieve liveness in the presence of faulty nodes, classical mainstream approaches resort to an additional, often complicated/subtle recovery path (see Section 1.2 for more discussion).

How does STREAMLET achieve both consistency and liveness with a unified *propose-vote* paradigm? The somewhat cute finalization rule is part of the "magic". An example of applying the finalization rule is shown in Figure 1. In this example, imagine all blocks are notarized: we see that there is a notarized chain with 3 adjacent blocks having consecutive epoch numbers 5, 6, and 7. Therefore, the entire prefix of the chain up to and including the epoch-6 block is considered final (i.e., all transactions contained in these blocks are confirmed and cannot be undone).

We shall prove later that in this case, there cannot be another conflicting block X notarized at the same length (i.e., position in the blockchain, or distance from the 'genesis block') as the epoch-6 block, and thus consistency is achieved. We now state the protocol's provable guarantees (slightly informally at this point).

Provable guarantees. In the remainder of the paper, we will describe the Streamlet protocol slightly more formally, and prove that this extremely simple protocol

1. achieves *consistency no matter what the actual network delays are*, even if the network is partitioned at times;

2. achieves *liveness with expected constant confirmation delay when network conditions are good*, i.e., during a period of time in which honest nodes can send messages back and forth to each other within a 1-second round-trip time.

The conditions under which liveness holds are often referred to as *a period of synchrony* [8]. Obviously, if the network is partitioned and nodes cannot deliver messages to each other, the protocol cannot make progress. The Streamlet protocol guarantees consistency nonetheless, no matter how bad or adversarial the network conditions are. In the literature, protocols satisfying such properties are often said to be *partially synchronous* [8] and *partition tolerant*. It is well-known that partially synchronous protocols cannot tolerate $1/3$ or more corruptions [8] and thus the protocol described above achieves optimal resilience.

How do we set the epoch length? The epoch length is typically set to be an estimate of the message round-trip time under normal/good network conditions.

Variants. With very minor tweaks, we obtain a couple variants:

- *An honest-majority, synchronous variant.* With a couple minor tweaks, we obtain a new protocol and prove it secure under honest majority in a synchronous network — note that due to the $1/3$ partially synchronous lower bound [8], network synchrony assumptions are indeed necessary for resisting minority corruptions.

  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[2]; furthermore, at finalization time, checking to make sure that the 6 blocks at consecutive epochs do not have conflicting notarizations at the same lengths.

- *A crash-fault tolerant, honest majority variant.* Taking the protocol above, changing the notarization threshold to $> n/2$ and removing the use of signatures, would give a crash-fault tolerant version secure under honest majority. This variant is also perfect for pedagogy: one can teach it without introducing digital signatures.

1.2 Comparison with Prior Approaches

Classical mainstream protocols: simple fast path + complicated recovery. Classical mainstream approaches such as PBFT [3], Paxos [14], and their numerous variants [10, 13, 18, 20], 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

[2] These constants can be reduced at the expense of introducing a couple more instructions to the protocol description; our goal in this paper is not to make these constants the smallest possible, but to make the protocol description as simple as possible. In Section 4, we will briefly discuss how one can make these constants smaller.

is corrupt; whereas the view change protocol must essentially embed a "full-fledged" consensus subprotocol that offers both consistency and liveness. As such, the complexity of classical protocols such as PBFT and Paxos arises due to the view change.

For decades, researchers have made it a top priority to simplify these consensus protocols, and various attempts have been made [15, 20, 27].

Recent wave of work motivated by decentralized blockchains. Very recently, due to interest in decentralized cryptocurrencies, various blockchain projects made independent efforts to simplify consensus protocols [2, 4, 5, 11, 25, 28]. As mentioned, more recently, the community's focus has been on permissioned consensus protocols due to the paradigm shift towards Proof-of-Stake. This line of work is known to be difficult to navigate partly due to the use of disparate terminology. Streamlet draws inspiration from many protocols [2, 4, 5, 26, 28] that came out during this wave. For example, our finalization rule is inspired by Casper [2], Hotstuff [28], Pili [5], and Pala [4].

Although the simplicity of the Streamlet protocol makes it kind of trivial and obvious in hindsight, it took the community *several years of effort and multiple iterations to eventually strip the protocol down to its current form.* Therefore we hope that the simplicity and the multiple iterations of efforts do not detract from our contribution.

Most closely related work. The closest related work is the very recent work by Shi [26]. Their work, which is in turn inspired by others' [2, 4, 5, 28], is an important step towards the same goal, i.e., to have a unified, simple protocol for both teaching and implementation. The main difference between our protocol and theirs is the following: in their protocol, blocks in a valid blockchain have an additional property called *freshness*. A block's freshness is determined by the epoch in which it is notarized. In Shi's protocol [26], a voter has to compare a proposed block against the *freshest notarized chain seen at the beginning of the previous epoch*, to determine whether it is okay to vote on the block. In private communication with the author, we learned that this protocol rule has caused confusion during lectures and also for engineers who implemented a protocol [4] similar to Shi's proposal [26].

Streamlet completely eliminates the freshness notion, and voters simply compare the proposed block with the longest notarized chain(s) that it has seen at the time of voting — a change that also simplifies the consistency proof.

Sequential composition of single-shot consensus. Finally, Byzantine Agreement or Byzantine Broadcast [16] is a single-shot consensus abstraction often studied in the theoretical literature. An alternative for constructing a blockchain is through sequential/parallel composition of single-shot consensus. However, cross-instance pipelining is difficult to accomplish in practice. Partly due to this reason, to the best of our knowledge, most practically deployed schemes employ the "direct blockchain construction" approach. Typically, in a directly constructed blockchain, the entire protocol is *streamlined* and there is no clear-cut boundary between single-shot instances. Directly constructed blockchains are often easier to optimize in practice.

Streamlet is a directly constructed blockchain, and we take the streamlining idea to its extreme, such that the entire protocol is a streamlined, unified path of "propose-vote" actions.

Non-goals. We are aware that some earlier works have strived to minimize the confirmation delay, e.g., Sync-Hotstuff [1]. We stress that minimizing the concrete constants related to confirmation delay is not a goal of our paper. In this paper, we instead take simplicity and understandability to be first-class principles. Having said this, we briefly discuss in Section 4 how to reduce the constants by introducing just a couple extra instructions to the Streamlet protocol.

In this paper, we focus on static security. We refer the reader to existing works [4, 7] that present somewhat generic approaches for upgrading from static security to adaptive security or mildly adaptive security [22], e.g., in a Proof-of-Stake context.

2 Execution Model and Definitions

Execution model. Let us briefly define the execution model. There are in total n nodes numbered $1, 2, \ldots, n$ respectively, and let $[n] := \{1, 2, \ldots, n\}$. All nodes have a public key that is common knowledge. Each node retains its own secret key for signing messages.

The adversary may control a subset of the nodes which are said to be *corrupt*, and the remaining nodes not under adversarial control are *honest*. Honest nodes always faithfully follow the prescribed protocol but corrupt nodes can deviate from the prescribed protocol arbitrarily (often called Byzantine faults). We assume that the adversary chooses which nodes 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 nodes. We require "periods of synchrony" for liveness, modeling them using the standard Global Stabilization Time (GST) approach [8]. Roughly speaking, before the GST, message delays can be arbitrary; however, after GST, messages sent by honest nodes 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 node sends a message in round r, an honest recipient is guaranteed to receive it by the beginning of round $\max(GST, r + \Delta)$.

Definition of a blockchain protocol. Nodes participating in a blockchain protocol receive inputs (e.g. transactions) from an external environment. The nodes are tasked with maintaining 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 node's output is the blockchain it maintains. Henceforth if a node p outputs some blockchain ch at some time, we also say that p considers ch *final*, or that ch is the *finalized* chain for node p.

Without loss of generality, we may assume that the protocol guarantees that a node p's output chain never decreases in length — given any protocol Π in which a node p's output chain may decrease in length, we can always compile it to a modified protocol Π' in which if any node 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 nodes, 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 node receives some input txs in round r, txs will eventually be included in all honest nodes' finalized chains.

3 The Streamlet Protocol: Partially Synchronous

In this section, we present our protocol formally and prove its correctness.

3.1 Epoch and Leader Rotation

Epochs: The protocol runs sequentially in synchronized epochs that are each 2Δ long. Recall that the parameter Δ determines when a period of synchrony occurs. During a period of synchrony, honest nodes can communicate with each other in at most Δ rounds.

Epoch leader: Every epoch e is mapped to a random leader using a public hash function $H : \{0,1\}^* \rightarrow [n]$ which is modeled as a random oracle. Specifically, epoch e's leader is computed as $H(e)$.

3.2 Blocks and Blockchain

Block: Each block is a tuple of the form (h, e, txs) where

- h is called the *parent hash*, i.e., a hash of the prefix of the chain;
- e is called the *epoch number* of the block, which records the epoch in which the block is proposed and voted on; and
- tx is a payload string (e.g., it may record a set of transactions to be confirmed).

A special block of the form $(\perp, e = 0, \perp)$ is called the *genesis block* which is the first block of every valid blockchain.

Blockchain: A blockchain chain is a sequence of blocks starting with the genesis block $\mathsf{chain}[0] := (\perp, e = 0, \perp)$, and with *strictly increasing* epoch numbers. Throughout we use $\mathsf{chain}[\ell]$ to refer to the ℓth block in chain, and $\mathsf{chain}[: \ell]$ to refer to the prefix of chain ending at (and including) the block $\mathsf{chain}[\ell]$.

For a blockchain chain to be valid, it must be that for every non-genesis block $\mathsf{chain}[\ell]$ where $\ell > 0$, $\mathsf{chain}[\ell]$ contains a parent hash equal to $H^*(\mathsf{chain}[: \ell - 1])$, where H^* denotes a hash function that was chosen from a collision-resistant hash family during a setup phase.

In a valid blockchain, we often say that a block $\mathsf{chain}[\ell]$ *extends from* the parent chain $\mathsf{chain}[: \ell - 1]$. Moreover, the block $\mathsf{chain}[\ell]$ is also said to have *length* ℓ, where ℓ is its 'distance' from the genesis block. The chain itself is also said to have a *length*, equivalent to the number of blocks in chain excluding the genesis block.

Remark 1 (The hash-chain data structure)**.** If we assume that the adversary did not find any hash collisions during the lifetime of the protocol, then the hash-chained data structure guarantees that *given a block, its prefix is uniquely determined.*

Remark 2 (A practical optimization)**.** Although we write $H^*(\mathsf{chain}[: \ell])$ for conceptual simplicity, in practice, H^* is typically implemented as an incremental hash (rather than having to hash the entire prefix chain for every block). In other words, each block's parent hash h may be computed simply as a hash of the parent block, which contains a hash of its own parent, and so on.

3.3 Votes and Notarization

A vote on a block is simply the node's signature on the block. A collection of at least $2n/3$ signatures from distinct nodes on the same block is called a *notarization* on the block. If a node i has observed a notarization for some block, that block is said to be *notarized* in i's view.

A valid blockchain is considered to be notarized by a node i if i has observed a notarization for every block in the blockchain (except for the genesis block).

3.4 Protocol

Although not explicitly stated, we make an implicit echoing assumption:

> *Implicit echoing:* upon observing a new transaction or message, a node always echos the transaction or message to everyone else.

We use the notation $\{m\}_{\mathsf{pk}_i^{-1}}$ to denote a message m, along with node i's signature on m. With this in mind, we now describe the Streamlet protocol[3].

The Streamlet blockchain protocol ($<1/3$ corrupt, partially synchronous)

For each epoch $e = 1, 2, \ldots$:

- **Propose:** At the beginning of epoch e, epoch e's leader L does the following: let chain be (any one of) the longest notarized chain(s) that the leader has seen so far, let $h := H^*(\mathsf{chain})$, and let txs be the set of unconfirmed pending transactions.

  The leader L sends to everyone the proposed block $\{(h, e, \mathsf{txs})\}_{\mathsf{pk}_L^{-1}}$ extending from the parent chain chain.

- **Vote:** During the epoch e, every node i does the following. Upon receiving the first proposal $\{(h, e, \mathsf{txs})\}_{\mathsf{pk}_L^{-1}}$ from epoch e's leader L, vote for the proposed block iff it extends from one of the longest notarized chains that node i has seen at the time.

  To vote for the proposed block (h, e, txs), node i simply sends to everyone $\{(h, e, \mathsf{txs})\}_{\mathsf{pk}_i^{-1}}$.

Finalize: On seeing three adjacent blocks in a notarized blockchain with consecutive epoch numbers, a node can finalize the second of the three blocks, as well as its entire prefix chain.

Note that when a block is *finalized* by a node i, the block and its entire prefix chain shows up the i's local finalized log; all the transactions contained in the block and its prefix are confirmed and can never be undone in the future.

In sum, the entire protocol follows a propose-vote paradigm, with a somewhat natural but also "magical" finalization rule. We give an example of the finalization rule below to aid understanding.

Example. To aid understanding, we illustrate the finalization rule in Figure 1. In this example, all blocks are notarized, and in the top chain, we have three adjacent blocks with consecutive epoch numbers $5, 6$, and 7. In this case, we can finalize the entire prefix chain "$\perp$ - 2 - 5 - 6".

[3] Throughout, we assume that if a node receives ill-formed or unanticipated messages, they are discarded or queued until they become anticipated. For example, an epoch-e proposal signed by a non-leader of epoch e is discarded.

To argue consistency, we will later prove that there cannot be a conflicting block notarized at the same *length* (i.e., distance from the genesis block) as the block with epoch 6, and thus the bottom chain "$\perp$ - 1 - 3" basically cannot grow further. In the figure, this is denoted as the crossed-out block for an arbitrary epoch X.

3.5 Consistency

For simplicity, henceforth we shall assume that both the signature scheme and the collision-resistant hash function H^* are ideal, i.e., there are no signature forgeries and no hash collisions.

We first give a simple proof for the above example — then the full, formal proof basically follows by changing the parameters in our argument to general ones. For completeness, we present the full proof in the Appendix (Section B).

In the example shown in Figure 1, we want to show that there cannot be any other conflicting block notarized at the same length as the block 6.

Suppose for the sake of contradiction that indeed some other conflicting block got notarized at the same length as block 6, e.g., the block with epoch number X. The following lemma says that X must either be greater than 7 or smaller than 5. We will use the term "in honest view" to mean that some honest node observes it at some time during the protocol execution.

Lemma 1 (Unique notarization per epoch)**.** *Suppose that $f < n/3$. Then, for each epoch e, there can be at most one notarized block for epoch e in honest view.*

Proof. Suppose that two blocks B and B', both of epoch e, are notarized in honest view. It must be that at least $2n/3$ nodes, denoted the set S, signed the block B, and at least $2n/3$ nodes, denoted the set S', signed the block B'. Since there are only n nodes in total, $S \cap S'$ must have size at least $n/3$, and thus at least one honest node is in $S \cap S'$. According to our protocol, every honest node votes for at most one block per epoch. This means that $\mathsf{B} = \mathsf{B}'$. □

Because of Lemma 1, the conflicting block notarized at the same length as block 6 must have an epoch number either greater than 7 or smaller than 5. We will therefore consider the two cases one by one, and the proofs for the two cases are nearly identical.

Case 1: $X < 5$. Since block X is notarized, it means that more than $n/3$ honest nodes, denoted S, voted for block X and not only so, at the time of the voting (that is, during epoch $X < 5$), they must have observed block 3 notarized. Now the honest nodes in S will not vote for block 5 during epoch 5, since it fails to extend a longest notarized chain seen, which is block 3 or longer. Since $f < n/3$, this means that block 5 can never get notarized in honest view. This leads to a contradiction.

Case 2: $X > 7$. Since block 7 is notarized, more than $n/3$ honest nodes (denoted the set S) must have seen a notarized block 6 by the time they vote for block 7 (i.e., by the end of epoch 7). As a result, by in epoch $X > 7$, the set S of nodes must have seen block 6 notarized and will not vote for block X, since block X now fails to extend the longest notarized chain seen (which is block 6 or longer). Therefore block X cannot gain $2n/3$ votes, and it cannot be notarized, which is a contradiction.

The above consistency proof was for our specific example earlier, but it can easily be generalized to the following statement.

Lemma 2 (Main consistency lemma)**.** *If some honest node sees a notarized chain with three adjacent blocks* $\mathsf{B}_0, \mathsf{B}_1, \mathsf{B}_2$ *with consecutive epoch numbers* $e, e+1$*, and* $e+2$*, then there cannot be a conflicting block* $\mathsf{B} \neq \mathsf{B}_1$ *that also gets notarized in honest view at the same length as* B_1*.*

Proof. Essentially the same as the argument above, but with generalized parameters rather than 6, 7, and 8. We defer the proof to the Appendix, in Section B Lemma 14. □

Lemma 2 leads to the following consistency theorem. Recall that "$\preceq$" means "is a prefix of or equal to", and that the $\mathsf{chain}[:\ell]$ notation refers to a prefix of chain ending at (and including) block $\mathsf{chain}[\ell]$.

Theorem 3 (Consistency)**.** *Suppose that* chain *and* chain' *are two notarized chains in honest view both ending at three blocks with consecutive epoch numbers. Denote the length of* chain *as* ℓ*, and the length of* chain' *as* ℓ'*. Moreover, suppose that* $\ell \leq \ell'$*. It must be that* $\mathsf{chain}[:\ell-1] \preceq \mathsf{chain}'[:\ell'-1]$*.*

Proof. Suppose $\mathsf{chain}[:\ell-1]$ is not a prefix of or equal to $\mathsf{chain}'[:\ell'-1]$. Then it must be that $\mathsf{chain}[\ell-1] \neq \mathsf{chain}'[\ell-1]$; in other words, they have different blocks at length $\ell-1$. However, Lemma 2 precludes both of $\mathsf{chain}[\ell-1]$ and $\mathsf{chain}'[\ell-1]$ from getting notarized in honest view. □

The consistency proof holds regardless of network timing. By examining the entire proof, it is not hard to see that the consistency proof holds no matter what the actual network message delays are, and even if honest nodes' clocks are not synchronized (as long as the epoch numbers are monotonically increasing). Of course, if the network is partitioned and honest nodes' messages are being held up, then we cannot guarantee progress. As explained in the following section, liveness ensues during "periods of synchrony", i.e., during a period of time in which honest nodes can deliver messages back and forth within a round-trip time of 1 second (i.e., equal to the epoch length).

3.6 Liveness

3.6.1 Liveness Theorem

Recall that after GST, the network enters a period of synchrony, i.e., honest nodes can deliver messages to each other within Δ rounds.

The following theorem states that after GST, whenever there are 5 consecutive epochs all with honest leaders, liveness ensues.

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 node must have observed a new final block that was not final at the beginning of epoch* e*. Moreover, this new block was proposed by an honest leader.*

Before we prove the liveness theorem, let us first try to understand it intuitively. Recall that earlier in our protocol, we need a notarized chain to have three adjacent blocks with consecutive epoch numbers to finalize blocks. You might wonder why in the liveness theorem above, we require 5 consecutive epochs with honest leaders (during a period of synchrony) to make progress. At a very high level, we first need a couple honest leaders to "undo" the damage that corrupt leaders have done, so that the following three honest leaders can successfully notarize three consecutive blocks. We formalize this intuition below.

Since the epoch leaders are randomly chosen by a hash function (modeled as a random oracle), a fixed sequence of 5 consecutive epochs will all have honest leaders with $\Theta(1)$ probability. In other words, after GST, transactions can be confirmed in an *expected constant* number of rounds.

3.6.2 Liveness Proof

In the remainder of this section, we will focus on proving the liveness theorem (i.e., Theorem 4).

First, because of the aforementioned implicit echoing as well as the Δ-bounded assumption during periods of synchrony, the following fact is immediate:

Fact 1 (Strong Δ-bounded message delivery for periods of synchrony)**.** *If an honest node observes a message in round r, then all honest nodes will have observed the message at the beginning of round* $\max(GST, r + \Delta)$.

Fact 2. *Suppose that some honest node i has observed a notarization for a block B at the beginning of round r. It must be that at the beginning of round* $\max(GST, r + \Delta)$*, every honest node has seen a notarized chain ending at B.*

Proof. Some honest node j must have signed the block B in or before round r, and thus j must have observed B's parent chain notarized (henceforth denoted ch) by the beginning of round r.

By Fact 1, all honest nodes will have observed the notarized parent chain ch as well as a notarization for B by $\max(GST, r + \Delta)$. □

Fact 3. *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 lengths ℓ_0 and ℓ_1 respectively in epochs e and $e+1$ respectively, it must be that $\ell_1 \geq \ell_0 + 1$.*

Proof. At the beginning of the round[4] Δ into epoch e, every honest node will have observed L_e's proposal for epoch e comprising of a block B at length ℓ_0. Moreover, due to Fact 1, the parent chain of B that triggered L_e to propose B must have been observed by every honest node at the beginning of the round Δ into epoch e.

Every honest node will sign this proposal *unless* by the beginning of round Δ into epoch e it has already observed a conflicting notarized chain of length ℓ_0 (in other words, B does not extend from one of the longest notarized chains seen so far). Now there are two cases:

1. If at least one honest node, say i, has observed a conflicting notarized chain of length ℓ_0 by round Δ of epoch e, then due to Fact 1, by the beginning of epoch $e+1$, every honest node must have seen a notarized chain of length ℓ_0.

2. If no honest node has seen a conflicting notarized chain of length ℓ_0 by round Δ of the epoch, then all honest nodes 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 node will have observed the notarized chain ending at B of length ℓ_0.

Therefore, the honest L_{e+1} must propose a block at length $\ell_0 + 1$ or greater. □

Lemma 5 (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 holds — below we use B to denote the block proposed by L_{e+2} during epoch $e+2$:*

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

a) by the beginning of epoch $e + 3$, every honest node will observe a notarized chain ending at B *(and* B *was not notarized before the beginning of epoch e);*

b) furthermore, no conflicting block $\mathsf{B}' \neq \mathsf{B}$ *with the same length as* B *will ever get notarized in honest view.*

Proof. Let ℓ_0, ℓ_1, ℓ_2 be the respective lengths 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 3, $\ell_2 > \ell_1 > \ell_0$. Recall that B is the block proposed by L_{e+2} in epoch $e + 2$ whose length is ℓ_2. We now argue that by the beginning of epoch $e + 3$, no honest node signed a conflicting $\mathsf{B}' \neq \mathsf{B}$ at the same length as B. Suppose that this is not true and some honest node i^* signed a conflicting $\mathsf{B}' \neq \mathsf{B}$ at length ℓ_2 by the beginning of epoch $e+3$. Now, i^* cannot have signed B during epochs e, $e + 1$, or $e + 2$ since L_e and L_{e+1} cannot have proposed a block at length ℓ_2 in epochs e and $e + 1$ respectively by Fact 3, and L_{e+2} proposed $\mathsf{B} \neq \mathsf{B}'$ in epoch $e + 2$.

Therefore, the honest node i^* must have signed B before epoch e started, and at this time i^* must have observed a notarized parent chain ending at length $\ell_2 - 1$. Due to Fact 1, this notarized parent chain ending at length $\ell_2 - 1$ must have been observed by all honest nodes by the beginning of epoch $e + 1$. Therefore, L_{e+1} must propose a block at length at least ℓ_2. Thus we have reached a contradiction.

Since by the beginning of epoch $e + 3$, no honest node has signed any $\mathsf{B}' \neq \mathsf{B}$ at length ℓ_2, at this time there cannot be a notarization for any $\mathsf{B}' \neq \mathsf{B}$ at length ℓ_2 in honest view. Moreover, L_{e+2}'s proposal and the notarized parent chain that triggered the proposal will be observed by all honest nodes at the beginning of round Δ into epoch $e + 2$. Therefore all honest nodes will sign and multicast B in or before round Δ into epoch $e + 2$. Thus by the beginning of epoch $e + 3$, all honest nodes will have seen a notarization for B. Thus, no honest node will ever sign any conflicting $\mathsf{B}' \neq \mathsf{B}$ at length ℓ_2 after the start of epoch $e + 3$ either; and any conflicting $\mathsf{B}' \neq \mathsf{B}$ at length ℓ_2 cannot ever gain notarization in honest view. □

Now, we can restate and prove Theorem 4.

Additional notation. Henceforth if B is a block, the notation $|\mathsf{B}|$ denotes the length of B.

Theorem 6 (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 node must have observed a new final block that was not final at the beginning of epoch e. Moreover, this new block was proposed by an honest leader.*

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 nodes' finalized chain — specifically, in our protocol, an honest node always proposes a new block containing all outstanding transactions it has seen.

Proof. Due to Lemma 5, by the beginning of epoch $e + 5$, the blocks proposed by L_{e+2}, L_{e+3}, and L_{e+4}, henceforth denoted B_2, B_3, and B_4, will be notarized in every honest node'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 the lengths $\ell_2 := |\mathsf{B}_2|, \ell_3 := |\mathsf{B}_3|$, and $\ell_4 := |\mathsf{B}_4|$.

We now show that B_3 must extend from the parent chain ending at B_2 and similarly B_4 must extend from the parent chain ending at B_3. To see this, notice that by Lemma 5, at the beginning of epoch $e + 3$, the notarized parent chain ending at B_2 must have been observed by all honest nodes and no honest node 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_2. Similarly L_{e+4}'s proposal must extend from the chain ending at B_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 node. □

4 Streamlet: Synchronous, Honest Majority

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. A *synchronous network* is one in which the Δ-bounded message delivery assymption always holds, i.e., the entire execution is during a period of synchrony. Due to the lower bound by Dwork et al. [8], such a network synchrony assumption is necessary (for consistency) to tolerate $n/3$ or more corruptions.

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.

The Streamlet blockchain protocol (honest majority, synchronous)

Notarization for a block B: a collection of signatures on B at least $n/2$ distinct nodes.

For every epoch $e = 1, 2, \ldots$:

- **Propose.** At the beginning of epoch e, epoch e's leader L does the following: let chain be (any one of) the longest notarized chain(s) that the leader has seen so far, let $h := H^*(\mathsf{chain})$, and let txs be the set of unconfirmed pending transactions.

  The leader L sends to everyone the proposed block $\{(h, e, \mathsf{txs})\}_{\mathsf{pk}_L^{-1}}$ extending from the parent chain chain.

- **Vote.** During the epoch e, every node i does the following. Upon receiving the first valid proposal $\{(h, e, \mathsf{txs})\}_{\mathsf{pk}_L^{-1}}$ from epoch e's leader L, vote for the proposed block iff it extends from one of the longest notarized chains it has seen at the time.

  To vote for the proposed block (h, e, txs), node i simply sends to everyone $\{(h, e, \mathsf{txs})\}_{\mathsf{pk}_i^{-1}}$.

Finalize: At any time, upon observing a notarized chain such that the last 6 blocks have consecutive epoch numbers, and moreover no notarizations have been observed for any conflicting

[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*.

block at the same lengths of the last 6 blocks, then, the prefix of chain removing the last 5 blocks is considered *final*.

Remark 3 (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 couple more instructions 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 the proposal and votes 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 4 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

4.2.1 Consistency Proof

Unlike Section 3, for our honest majority protocol, the consistency proof must now rely on network synchrony.

Fact 4. *Suppose that some block* B *of epoch* e *is notarized in honest view, it must be that by the beginning of epoch* $e+2$*, every honest node has observed the parent chain of* B *notarized.*

Proof. Since B is notarized in honest view, at least $n/2$ nodes have signed B in epoch e, and since $f < n/2$, then at least one honest node i must have signed B in epoch e. Moreover, i must have observed the parent chain for B notarized in epoch e. By our Fact 1, every honest node will have observed the parent chain notarized by the beginning of epoch $e+2$. □

Lemma 7. *Suppose that there is a notarized* chain *in honest view containing two adjacent blocks* $\mathsf{chain}[\ell]$ *and* $\mathsf{chain}[\ell+1]$ *at consecutive epoch numbers* e *and* $e+1$ *respectively, then, no block of length* ℓ *and epoch greater than* $e+2$ *can be notarized in honest view.*

Proof. By Fact 4, every honest node will have observed a notarized chain $\mathsf{chain}[:\ell]$ that is a valid parent of $\mathsf{chain}[\ell+1]$ by the beginning of epoch $e+3$. This means that no honest node will sign any block of length ℓ or smaller in epoch $e+3$ or greater. Thus no block at length ℓ and of epoch greater than $e+2$ can be notarized in honest view. □

Theorem 8 (Consistency)**.** *Suppose that two notarized chains* chain *and* chain' *of lengths* ℓ *and* ℓ' *respectively both triggered the finalization rule in some honest node's view, that is,* $\mathsf{chain}[:\ell-5]$ *and* $\mathsf{chain}'[:\ell'-5]$ *are finalized in honest view. Without loss of generality, suppose that* $|\mathsf{chain}| \leq |\mathsf{chain}'|$*. It must be that* $\mathsf{chain}[:\ell-5] \preceq \mathsf{chain}'[:\ell'-5]$*.*

Proof. Suppose that this is not true, it must be that $\mathsf{chain}'[\ell-5] \neq \mathsf{chain}[\ell-5]$ and thus $\mathsf{chain}'[\ell-4] \neq \mathsf{chain}[\ell-4]$. Let $e-5, e-4, \ldots, e$ be the epochs of $\mathsf{chain}[\ell-5], \mathsf{chain}[\ell-4], \ldots, \mathsf{chain}[\ell]$ respectively. By Lemma 7, $\mathsf{chain}'[\ell-4]$ must be of epoch $e' \leq e-2$. By Fact 4, by the beginning of epoch $e'+2 \leq e$, every honest node will have observed a notarized parent chain $\mathsf{chain}'[:\ell-5]$. Obviously $\mathsf{chain}[\ell]$ cannot gain notarization in honest view before epoch e. This means that when any honest node observes a notarization for $\mathsf{chain}[\ell]$, it must also have observed a notarization for $\mathsf{chain}'[\ell-5]$. Thus no honest node will successfully apply the finalization rule to chain due to this conflicting notarization for $\mathsf{chain}'[\ell-5]$ and this leads to a contradiction. □

4.2.2 Liveness Proof

Fact 2, Fact 3, and Lemma 5 of Section 3 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 9 (Liveness)**.** *Suppose that there are 8 consecutive epochs $e, e+1, \ldots, e+7$ all with honest leaders, then by the beginning of epoch $e+7$, every honest node 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 6, additionally observing that there cannot ever be any conflicting notarizations in honest view at the same lengths as the blocks proposed by L_{e+2}, $L_{e+3}, \ldots$, L_{e+7} due to Lemma 5. □

5 Acknowledgements

This work is in part supported by NSF under grant number CNS-1561209, and by a Packard Fellowship.

References

[1] 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.

[2] Vitalik Buterin and Vlad Zamfir. Casper. https://arxiv.org/abs/1710.09437, 2017.

[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. Cryptology ePrint Archive, Report 2018/981, 2018. https://eprint.iacr.org/2018/981.

[5] T-H. Hubert Chan, Rafael Pass, and Elaine Shi. Pili: An extremely simple synchronous blockchain. 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: Robustly reconfigurable consensus and applications to provably secure proofs of stake. In *Financial Crypto*, 2019.

[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 seriesconsensus 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] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 1998.

[15] Leslie Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58, December 2001.

[16] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 1982.

[17] Butler Lampson. The abcd's of paxos. In *PODC*, page 13, 2001.

[18] Jean-Philippe Martin and Lorenzo Alvisi. Fast byzantine consensus. *IEEE Trans. Dependable Secur. Comput.*, 3(3), 2006.

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

[20] 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.

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

[22] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, 2017.

[23] Rafael Pass and Elaine Shi. Rethinking large-scale consensus (invited paper). In *CSF*, 2017.

[24] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, December 1990.

[25] Elaine Shi. Streamlined blockchains: A simple and elegant approach (A tutorial and survey). In *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, pages 3–17, 2019.

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

[27] Robbert Van Renesse and Deniz Altinbuken. Paxos made moderately complex. *ACM Comput. Surv.*, 47(3), February 2015.

[28] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 347–356, 2019.

A Streamlet for Crash Faults

In this section, we present a version of Streamlet for the *crash fault* setting, tolerating $f < n/2$ crash faults in a partially synchronous setting, and forgoing the need for cryptography. This is also the optimal resilience for crash faults in a partially synchronous network [8]. This crash-tolerant variant can be taught without needing to teach digital signatures, and is also well-suited for performance-hungry applications where faults are benign and the delay of digital signing may be unacceptable.

A.1 Execution Model

In prior sections, we considered faulty nodes that are malicious in nature and can deviate arbitrarily from the protocol. In this section, we consider a weaker adversary that can only cause *crash faults*, also known as stopping failures.

The model is the same as in Section 2, with two main differences. First, nodes no longer have cryptographic keys. We do, however, continue to assume that the n nodes are numbered $1, 2, \ldots, n$ respectively, and that each node knows its own number, henceforth referred to as its *id*.

Second, the adversary now has the following, more restricted power. The adversary (as before) can corrupt a subset of the nodes partaking in the protocol prior to the start of the protocol — these nodes are said to be *faulty*, while the remaining nodes are said to be *honest*. All nodes follow the prescribed protocol, but faulty nodes may *crash* at a time of the adversary's choice during the protocol execution. A crashed node halts all further local execution. A node may crash in the middle of sending a message to other nodes - so it is possible that during a round in which a message sender crashes, only a subset of the intended messages get delivered, even during periods of synchrony. For instance, a faulty node may try to send a vote to every other node in some round r, but if it crashes in round r, it may be that only a subset of those votes are actually sent.

We continue to use the same model of partial synchrony as before, guaranteeing the Δ-bounded message delivery assumption after *GST* only for honest (or non-faulty) nodes.

A.2 Streamlet for Crash Faults

We now present a variant of Streamlet that tolerates $f < n/2$ *crash* faults, and this crash-tolerant variant does not use cryptography. We first define votes and notarizations.

Definition 1 (Vote)**.** A *vote* on a block B is the tuple $(\mathsf{vote}, \mathsf{B}, id)$, where id is the id of the sender. We say that a node 'votes' for a block if it sends a vote for that block to every other node.

Definition 2 (Notarization)**.** A collection of at $> n/2$ votes with distinct node ids for the same block is called a *notarization* for the block. A valid blockchain is considered to be *notarized* by a node i if i has observed a notarization for every block (except the genesis block).

Other definitions remain the same. We remind the reader that the length of a block B, denoted $|\mathsf{B}|$, is equivalent to the length of the valid blockchain that terminates at block B (excluding the genesis block).

Protocol. As before, we make an implicit echoing assumption:

> *Implicit echoing:* upon observing a new transaction or message, a node always echos the transaction or message to everyone else.

The crash-tolerant version of Streamlet is shown in the figure below. Essentially the protocol is the same as in Section 3, except that *the threshold of notarization is changed to* $>$ $n/2$ *and messages are no longer signed.*

Streamlet Protocol ($< 1/2$ crash failures, partially synchronous)

The protocol proceeds in incrementing epochs where each epoch is 2Δ rounds.

For each epoch $e = 1, 2, \ldots,$

1. *Propose.* At the beginning of epoch e, epoch e's leader L sends to everyone a new proposed block $(H^*(\mathsf{chain}), e, \mathsf{txs})$, where the parent chain is any of the longest notarized chains seen so far by L, and where txs is the set of unconfirmed pending transactions. In addition, L sends to everyone the notarized parent chain.
2. *Vote.* Every player does the following during epoch e: When a player receives a proposed block from L for epoch e for the first time, along with its notarized parent chain, the player 'votes' for the proposed block i.f.f. it is strictly longer than any notarized chain that the player has seen thus far.

Finality rule. If at any time, a player sees three consecutive notarized blocks with consecutive epoch numbers, the player finalizes the second of the three blocks, and its entire parent chain. Whenever requested, the player outputs the longest finalized chain it has seen thus far.

A.3 Consistency

In this section we present a consistency proof for the crash-fault tolerant protocol. The proof itself is essentially the same as the analysis for the malicious setting - with modifications to account for the new fault-tolerance threshold and a different notarization threshold.

We will use the term "in honest view" to mean that some honest (i.e. non-faulty) node observes the specified message at some time during the protocol execution.

Lemma 10 (Unique notarization per epoch)**.** *If blocks* B *and* B' *with the same epoch number* e *are each notarized in honest view, it must be that* $\mathsf{B} = \mathsf{B}'$*.*

Proof. Let S be the set of $> n/2$ nodes that voted for B and let S' be the set of $> n/2$ nodes that voted for B'. The intersection of S and S' has size $|S \cap S'| > n/2 + n/2 - n = 0$, implying that at least one node must have voted for both B and B'. Assume for the sake of contradiction that $\mathsf{B} \neq \mathsf{B}'$. This is absurd, since by the protocol definition, a node votes for at most one block with the epoch number e (by virtue of voting at most once during epoch e, and only voting for proposals with an epoch number that matches the current epoch). □

Lemma 11. *If some node sees a notarized chain with three adjacent blocks* $\mathsf{B}_0, \mathsf{B}_1, \mathsf{B}_2$ *with consecutive epoch numbers* $e, e+1,$ *and* $e+2$*, then there cannot be a conflicting block* $\mathsf{B} \neq \mathsf{B}_1$ *that also gets notarized in honest view with the same length as* B_1*.*

Proof. Denote the lengths of $\mathsf{B}_0, \mathsf{B}_1$, and B_2 as $\ell, \ell+1$, and $\ell+2$ respectively. Denote the epoch number of B as e_{B}.

Assume for the sake of contradiction that some block $\mathsf{B} \neq \mathsf{B}_1$ with the same length as B_1 is indeed notarized in honest view. By Lemma 10, e_{B} cannot equal $e, e+1$, or $e+2$, since $\mathsf{B} \notin \{\mathsf{B}_0, \mathsf{B}_1, \mathsf{B}_2\}$. Then there are two cases:

1. Case $e_{\mathsf{B}} < e$. Let S be the set of $> n/2$ nodes that voted for B and let S' be the set of $> n/2$ nodes that voted for B_0. Again, the intersection of S and S' has size $|S \cap S'| > 0$, implying that at least one node $i \in S \cap S'$ voted for both B and B_0.

   Now notice that node i must have voted for B by the end of epoch e_{B}, and thus before the beginning of epoch e. This implies that node i must have observed a notarized chain of length ℓ before the beginning of epoch e (by observing B's notarized parent chain, which is a prerequisite for voting). However, in epoch e, node i voted for B_0 which has length ℓ. This is a contradiction because by the protocol description, i cannot vote for B_0 since it has already seen a notarized chain of length ℓ by the time it votes.

2. Case $e_{\mathsf{B}} > e+2$. The argument here is nearly identical. This time, let S be the set of $> n/2$ nodes that voted for B and let S' be the set of $> n/2$ nodes that voted for B_2. Again, the intersection of S and S' has size $|S \cap S'| > 0$, implying that at least one node $i \in S \cap S'$ voted for both B and B_2.

   Node i must have voted for B_2 by the end of epoch $e+2$, and thus i observes a notarized parent chain of length $\ell+1$ by the end of epoch $e+2$. However, during epoch $e_{\mathsf{B}} > e+2$, node i votes for B which has length $\ell+1$. This is a contradiction because by the protocol description, i cannot vote for B since it has already seen a notarized chain of length $\ell+1$ by the time it votes.

□

Theorem 12 (Consistency)**.** *Suppose that* chain *and* chain' *are two notarized chains in honest view both ending at three blocks with consecutive epoch numbers. Denote the length of* chain *as* ℓ*, and the length of* chain' *as* ℓ'*. Moreover, suppose that* $\ell \leq \ell'$*. It must be that* $\mathsf{chain}[:\ell-1] \preceq \mathsf{chain}'[:\ell'-1]$*.*

Proof. The proof from Theorem 3 of the partially synchronous setting works here, replacing Lemma 2 with Lemma 11. □

A.4 Liveness

The liveness argument from Section 3 still works here word-for-word. So Fact 1, Fact 2, Lemma 5 and Theorem 6 from Section 3 hold and the protocol has liveness:

Theorem 13 (Liveness)**.** *After GST, suppose that there are 5 consecutive epochs $e, e+1, \ldots, e+4$ all with honest leaders. By the beginning of epoch $e+5$, every honest node must have observed a new final block that was not final at the beginning of epoch e. Moreover, this new block was proposed by an honest leader.*

B Additional Proofs

Here, we restate Lemma 2 and provide a complete proof. Recall for context that we are in the partially synchronous setting, and $f < n/3$.

Lemma 14 (Main consistency lemma)**.** *If some honest node sees a notarized chain with three adjacent blocks $\mathsf{B}_0, \mathsf{B}_1, \mathsf{B}_2$ with consecutive epoch numbers $e, e+1$, and $e+2$, then there cannot be a conflicting block $\mathsf{B} \neq \mathsf{B}_1$ that also gets notarized in honest view at the same length as B_1.*

Proof. Denote the lengths of $\mathsf{B}_0, \mathsf{B}_1$, and B_2 as $\ell, \ell+1$, and $\ell+2$ respectively. Denote the epoch number of B as e_{B}.

Assume for the sake of contradiction that some block $\mathsf{B} \neq \mathsf{B}_1$ with the same length as B_1 is indeed notarized in honest view. By Lemma 1, e_{B} cannot equal $e, e+1$, or $e+2$, since $\mathsf{B} \notin \{\mathsf{B}_0, \mathsf{B}_1, \mathsf{B}_2\}$. Then there are two cases:

1. Case $e_{\mathsf{B}} < e$. Let S be the set of $\geq 2n/3$ nodes that voted for B and let S' be the set of $\geq 2n/3$ nodes that voted for B_0. The intersection of S and S' has size $|S \cap S'| \geq 2n/3 + 2n/3 - n = n/3 > f$, implying that at least one *honest* node $i \in S \cap S'$ voted for both B and B_0.

   Now notice that node i must have voted for B by the end of epoch e_{B}, and thus before the beginning of epoch e. This implies that node i must have observed a notarized chain of length ℓ before the beginning of epoch e (by observing B's notarized parent chain, which is a prerequisite for voting). However, in epoch e, node i voted for B_0 which has length ℓ. This is a contradiction because by the protocol description, i cannot vote for B_0 since it has already seen a notarized chain of length ℓ by the time it votes.

2. Case $e_{\mathsf{B}} > e+2$. The argument here is nearly identical. This time, let S be the set of $\geq 2n/3$ nodes that voted for B and let S' be the set of $\geq 2n/3$ nodes that voted for B_2. Again, the intersection of S and S' has size $|S \cap S'| \geq 2n/3 + 2n/3 - n = n/3 > f$, implying that at least one *honest* node $i \in S \cap S'$ voted for both B and B_2.

   Node i must have voted for B_2 by the end of epoch $e+2$, and thus i observes a notarized parent chain of length $\ell+1$ by the end of epoch $e+2$. However, during epoch $e_{\mathsf{B}} > e+2$, node i votes for B which has length $\ell+1$. This is a contradiction because by the protocol description, i cannot vote for B since it has already seen a notarized chain of length $\ell+1$ by the time it votes.

□