Banyan: Fast Rotating Leader BFT

Yann Vonlanthen
yvonlanthen@ethz.ch
ETH Zurich
Switzerland

Jakub Sliwinski
jsliwinski@ethz.ch
ETH Zurich
Switzerland

Massimo Albarello
massimo@onfabric.io
ETH Zurich
Switzerland

Roger Wattenhofer
wattenhofer@ethz.ch
ETH Zurich
Switzerland

Abstract

This paper presents Banyan, the first rotating leader state machine replication (SMR) protocol that allows transactions to be confirmed in just a single round-trip time in the Byzantine fault tolerance (BFT) setting. Based on minimal alterations to the Internet Computer Consensus (ICC) protocol and with negligible communication overhead, we introduce a novel dual mode mechanism that enables optimal block finalization latency in the fast path. Crucially, the modes of operation are integrated, such that even if the fast path is not effective, no penalties are incurred. Moreover, our algorithm maintains the core attributes of the ICC protocol it is based on, including optimistic responsiveness and rotating leaders without the necessity for a view-change protocol.

We prove the correctness of our protocol and provide an open-source implementation of it. Banyan is compared to its predecessor ICC, as well as other well known BFT protocols, in a globally distributed wide-area network. Our evaluation reveals that Banyan reduces latency by up to 30% compared to state-of-the-art protocols, without requiring additional security assumptions.

CCS Concepts: • **Computer systems organization** → **Distributed architectures**; • **Security and privacy** → *Distributed systems security*.

Keywords: Consensus, Blockchain, Byzantine fault tolerance, Fast Path, State Machine Replication

MIDDLEWARE '24, December 2–6, 2024, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-0623-3/24/12
https://doi.org/10.1145/3652892.3700788

ACM Reference Format:
Yann Vonlanthen, Jakub Sliwinski, Massimo Albarello, and Roger Wattenhofer. 2024. Banyan: Fast Rotating Leader BFT. In *24th International Middleware Conference (MIDDLEWARE '24), December 2–6, 2024, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3652892.3700788

1 Introduction

Byzantine fault tolerance (BFT) is a class of protocols that provide guarantees in the presence of arbitrary faults, such as a powerful adversary controlling both a share of participants and the network scheduling [46]. BFT protocols can be applied whenever a fixed (also called permissioned) set of untrusted parties desires to reach an agreement.

This core primitive has enabled the creation of decentralized protocols that are akin to a world computer, enabling anyone to run Turing-complete programs. Various such world computers exist today [10, 16, 29, 31], with varying trade-offs regarding performance, security, and inclusiveness. At each system's core, lays an ordering protocol assuring that resource accesses are consistent across all participants (called replicas), thus guaranteeing deterministic and secure execution of the computation.

Byzantine agreement (BA) protocols typically provide consensus on a single decision, while state machine replication protocols (SMR) focus on making efficient use of this costly primitive. In practice, the most widely used consensus protocols (such as Bitcoin [51], Ethereum [16], and Algorand [35]) make use of an elected leader to propose a batch of transactions, called a block. A total order across leaders is then obtained by chaining blocks.

There are a few considerations that are especially crucial to such leader-based protocols. Firstly, leaders can misbehave and propose conflicting blocks. While this can be detected, the resulting change of the leader (called *view-change* in the literature) leads to notoriously high complexity. Honest, but slow leaders are just as problematic, as they cannot easily be called out, but can stall the effective throughput nonetheless [24]. Moreover, relying on a single leader leads to uneven load, both in terms of computation and communication [25], and might facilitate censorship. Finally, leaders sometimes extract value from their control of the block creation. On the

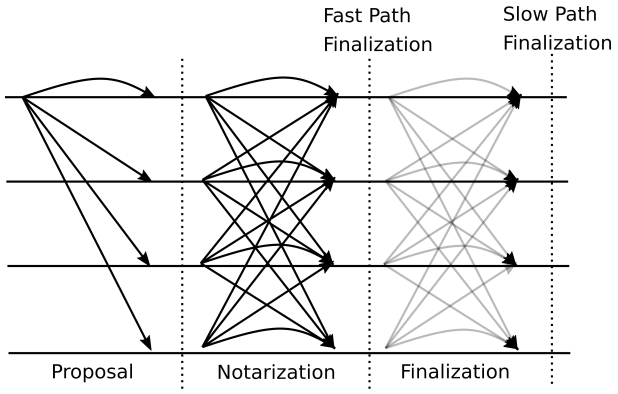


Figure 1. Banyan can terminate after just two communication steps. Existing rotating leader BFT protocols require at least three communication steps.

Ethereum blockchain, for example, miner-extractable value (MEV) is a major source of income for replicas and leads to potentially dangerous incentives to deviate from honest behavior [26]. These issues can at least partially be resolved by employing rotating or random leaders [19], which is the class of protocols we aim to improve upon in this work.

Specifically, we address what we believe to be a major roadblock in the adoption of decentralized computers: high latency. In contrast to privacy and security, latency is at the forefront of the user experience. While centralized and trusted applications (such as credit card payments or social media) provide almost instantaneous services, their Byzantine fault-tolerant counterpart typically comes at the cost of a noticeable delay, in some cases in the order of minutes or even hours (such as in Bitcoin). Furthermore, contrary to other metrics such as throughput, latency cannot be readily improved by hardware upgrades.

In this work, we present the first rotating leader SMR protocol that allows transactions to be confirmed in just a single round-trip time. This is optimal, as to achieve fault tolerance the block must 1) reach a fraction of replicas, that have to 2) respond to acquiesce to the processing [55]. This is visualized in Figure 1.

We are able to achieve this by uniting two lines of research. The first deals with the theoretical bounds of latency, and has established optimally fast single-shot protocols in the good case [3, 6, 43, 44, 50]. The second, more well-known line of research, provides a class of consensus protocols that solve state machine replication, in an efficient and fair way, by employing rotating or random leaders. Prominent examples include HotStuff, Tendermint, Casper, Algorand and Bullshark [14, 17, 35, 57, 61].

As Chan and Pass point out [21], rotating leader protocols can be differentiated by either favoring *proposal confirmation time*, meaning block finalization latency, or *block creation time*, corresponding to the delay between the blocks and thus the associated throughput. The Internet Computer Consensus family of protocols [19] excels in both of these metrics.

Our contribution. We present the Banyan protocol that achieves optimal consensus latency while preserving the state-of-the-art characteristics of ICC.

1. Banyan is the first SMR protocol with rotating leaders supporting two-step chain growth and two-step finalization under partial synchrony. We prove the safety and liveness of Banyan for $n \geq 3f + 2p^* - 1$, where n is the total number of replicas in the network, and f is the maximum number of Byzantine replicas tolerated. Additionally, $p^* \geq 1$ is a parameter that can be set freely ($p^* \leq f$), and determines the effectiveness of the fast path. We prove that as long as no more than p^* replicas are unresponsive, Banyan finalization latency is just two network delays.
2. We provide a proof-of-concept Banyan implementation written in Golang, built on top of the Bamboo BFT framework [33].
3. We measure the proposal finalization time in three different wide area networks. We empirically demonstrate that, compared to ICC, HotStuff, and Streamlet, Banyan achieves the fastest proposal finalization time, improving by up to 30% over the runner-up. Further, we show that Banyan does not suffer from increased variance in its latency, and behaves as ICC under crash-faults.

2 Related Work

Byzantine fault tolerance was introduced by Lamport et al. in 1982 [46]. A long line of work studies consensus protocols in the permissioned setting under various synchrony assumptions [9, 12, 45]). The partially synchronous network model was first introduced in [30], together with the first consensus protocol for this setting. The first truly practical protocol in the partially asynchronous setting is the PBFT protocol, as described in [20].

Fast Consensus. Our algorithm relies on a long line of established work on reaching consensus in two communication steps (or one round trip time), also referred to as fast or early-stopping consensus [13, 32, 36, 43, 50, 55]. As shown by Brasileiro et al. [13] in the crash-fault model, it is possible to terminate early if "enough" acknowledgments are received. A year later, a fast path was presented by Kursawe [43], in which a two-step fast path is paired with a subprotocol in the slow path. Upon the expiration of a timer, a fallback subprotocol is used to ensure liveness.

Song et al. describe Bosco [55], an algorithm that provides a two-step fast path to any underlying consensus algorithm. Their fast path does not rely on synchrony assumptions, and, assuming $n > 5f$, is triggered when all servers are in pre-agreement, i.e., propose the same value. Assuming $n > 7f$, the fast path is triggered when all honest servers are in

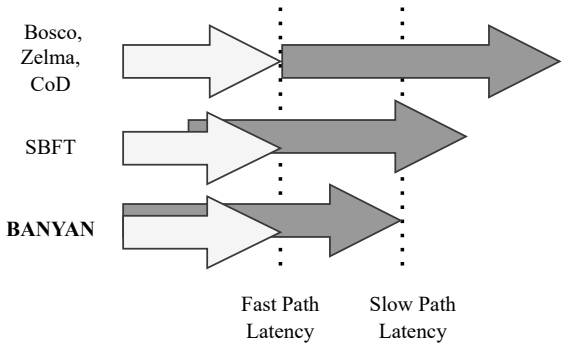


Figure 2. The different approaches of fast path protocols. In many protocols, the slow path starts after the fast path fails. In BANYAN, the fast path is integrated into the slow path.

pre-agreement instead. Using similar building blocks it was later shown that consensus can be performed "on demand", i.e., the fallback is only used when the fast path does not succeed [54]. As Figure 2 shows, these approaches come at the cost of a longer recovery during the fallback on the slow path though, as the slow path can only be started once the fast path round is over. In this work, we show that both the (single) fast path round and the first slow path round can happen simultaneously (with only constant message overhead).

Earlier, Martin and Alvisi introduce Fast Byzantine Paxos (FaB Paxos) [50], a family of protocols parameterized not only by f but also p ($0 \leq p \leq f$), the number of (non-leader) replicas that are not needed for the fast path to succeed. Their algorithms provide Byzantine agreement, when $n \geq 3f+2p+1$, where the fast path triggers as long as $n-p$ servers behave honestly and are in pre-agreement. Instead of relying on a variant of PBFT in the slow path, we integrate our protocol with ICC, sidestepping the complexity introduced by view changes.

The lower bound by Martin and Alvisi [50] has recently been improved to $n \geq max(3f+2p-1, 3f+1)$ by Kuznetsov et al. [44] and Abraham et al. [3]. Their insight is that misbehaving (i.e., equivocating) leaders can be reliably detected, and thus acceptors can wait for $n-f$ votes, *excluding* the malicious leader ($n-f+1$ votes in total). To the best of our knowledge, we present the first protocol that matches this lower bound without the need for a *view-change* protocol.

Fast SMR. Ideas from the FaB protocol were borrowed by the Zyzzyva protocol [42] as well, which turns the single-shot FaB consensus into an SMR protocol. Improving upon Zyzzyva, Aliph [6] reduces the latency to two steps. These protocols do not allow running both the fast path and a slow path concurrently, though, and thus suffer from switching costs (see Figure 2). Zyzzyva safety issues were pointed out by [1] and led to the development of Zelma [2], combining FaB and Zyzzyva's benefits.

Zelma is also at the core of the SBFT protocol [37]. SBFT is the first protocol running both pessimistic and optimistic paths simultaneously in a dual mode. However, the fast path of SBFT has one more communication step than BANYAN, and assumes $n \geq 3f + 2p + 1$. Moreover, the slow path of SBFT is not optimistically responsive, as it triggers only after a time-out (fig. 2). A large body of work has based itself on the 3-phase (6-step) HotStuff [61] protocol and presented improvements upon it [38, 40, 48]. Jolteon and Ditto [34] have improved the latency while adding an asynchronous fallback to enhance its performance during epoch synchronization (see Table 1).

Most recently Malkhi and Nayak propose a 2-phase (4-step) protocol and a simpler fast path [48]. These protocols reach no lower than 3δ proposer latency in the fast path.

Rotating Leader BFT protocols. Many recent advances in BFT protocols rely on rotating leaders [14, 15, 22, 61], as originally introduced by Veronese et al. [60]. Mir-BFT [58] and its follow-up work [59] for example, improve upon sequential-leader approaches, by running PBFT instances on a set of leaders. When a leader is slow, it is replaced. Other SMR protocols are designed to perform well under attack [5, 7]. However, in the permissioned blockchain setting it is often assumed that during periods of synchrony, misbehavior can be punished, either by exclusion or slashing [17]. As such, we focus on the latency in the optimistic case and satisfy the same pessimistic liveness as ICC, which was shown to be adequate in [21]. Recent concurrent work by Chan and Pass [21] provides a theoretical framework for this class of algorithms, and Table 1 is partially inspired by it. Their proposed protocol called Simplex falls into the same category as the ICC and BANYAN algorithms but improves *pessimistic liveness* by only allowing a single leader per round. We believe that a technique similar to the one shown in this work could be applied to bring our fast path to Simplex too. In [4] Abraham et al. prove that 2Δ is the upper and lower bound for the good-case latency of rotating leader BFT protocols, in the synchronous setting. In this work, we provide a matching upper bound in the partially synchronous setting.

DAG-based BFT protocols. Another line of work proposes to improve the throughput of leader-based protocols by allowing all parties to broadcast transactions simultaneously. Many so-called DAG-based protocols have emerged recently [8, 41, 49, 57].

One main idea is to disconnect transactions broadcasting from finalization [27], and another is to allow the finalization of blocks outside the main blockchain by causally referencing them [41, 47]. In Bullshark [57], all replicas are parallel leaders and broadcast one batch of transactions in each round using Byzantine Consistent Broadcast. While doing so, they include references to at least $n-f$ blocks from the previous round. A more recent protocol called BBCA-Chain [49] reduces Bullshark's latency by removing the need for specialized block layers and instead broadcasts blocks using a

	Block Finalization Latency	Block Finalization Requirement	Block Creation Latency	Block Creation Requirement	Number of Replicas *equals*	Supports Rotating Leaders
Casper FFG [17]	$O(\Delta)$	$2f+1$	$O(\Delta)$	N/A[†]	$3f+1$	✓
Fast HotStuff [38][‡]	5δ	$2f+1$	2δ	$2f+1$	$3f+1$	
Jolteon [34]	5δ	$2f+1$	2δ	$2f+1$	$3f+1$	
PaLa [23]	4δ	$2f+1$	2δ	$2f+1$	$3f+1$	
Zelma [2]	2δ	$3f+p+1$	2δ	$2f+p+1$	$3f+2p+1$	
SBFT [37]	3δ	$3f+p+1$	3δ[§]	$2f+p+1$	$3f+2p+1$	
Streamlet [22]	6Δ	$2f+1$	2Δ	$2f+1$	$3f+1$	✓
Bullshark [57]	4δ[¶]	$2f+1$	2δ	$2f+1$	$3f+1$	✓
BBCA-Chain [49]	3δ	$2f+1$	3δ	$2f+1$	$3f+1$	✓
ICC [19] / Simplex [21]	3δ	$2f+1$	2δ	$2f+1$	$3f+1$	✓
Mysticeti [8]	3δ	$2f+1$	1δ	$2f+1$	$3f+1$	✓
Banyan	2δ	$3f+p^*-1$	2δ	$2f+p^*$	$3f+2p^*-1$	✓

Table 1. Popular and Fast State Machine Replication Protocols. To simplify comparison, we assume that the number of replicas is equal to the respective lower bound. Δ denotes the message delivery time upper bound, while δ is the true message delivery time. [†] Non-equivocation is enforced by slashing. [‡] We consider the pipelined version of Fast HotStuff. [§] To the best of our knowledge no pipelining is specified for SBFT. [¶] We consider Bullshark's best case latency (anchor blocks). [*] For simplicity, we replace p by $p^*, p^* \geq 1$.

primitive called Byzantine Broadcast with Complete-Adopt (BBCA). Mysticeti [8] forgoes the confirmation of blocks and instead relies solely on the DAG edges as input to the protocol, leading to better optimistic latency. ICC does not allow causally referencing non-leader blocks to finalize them and thus does not benefit from the same throughput advantages. However, we believe ICC is a good fit to describe our fast path mechanism, and hypothesize that it applies to DAG protocols too.

In Table 1 an overview of state-of-the-art protocols is shown. For both the block finalization latency and the block creation latency, we have included the required number of replicas that have to respond in order to proceed in the optimistic case (synchronous round with a block proposed by an honest leader). While irrelevant in the calculation of the theoretical latency bounds, we found that small changes in these requirements could have large effects in practice, especially on global-scale deployments as they are seen today. (Intuitively, it is not uncommon for a few outlier replicas to have a higher latency. If progress can be made safely without them, a large performance increase can be experienced.)

3 Model

We consider a network of $n \geq \max(3f+2p-1, 3f+1)$ participants called replicas. Our protocol is safe and live with up to f replicas being Byzantine. Additionally, we guarantee *fast termination* when up to p replicas are not cooperating. In other words, the parameter $p \in [0, f]$ denotes the maximum number of replicas that are not needed for the fast path to be successful. Note that there is no reason to choose $p = 0$, as our protocol will be strictly faster with $p = 1$, and require the same number of replicas. We can thus also write $n \geq 3f + 2p^* - 1$, where $p^* \in [1, f]$.

By setting $p = 1$, we reach the upper bound on the number of Byzantine replicas permissible [30], i.e. $n \geq 3f+1$. In this case, the fast path will be used when the leader and all but one replica behave honestly. On the other side of the spectrum, given an honest leader, the fast path can be made robust against Byzantine behavior, by setting $p = f$.

We consider the same communication model used in ICC, in which replicas communicate over a partially synchronous network [30]. In a synchronous network, there is a known fixed upper bound Δ on the time required for a message to be sent from one party to another. In an asynchronous network, no fixed upper bound Δ exists. We use δ to denote the *true message delivery time* of replicas, i.e., the unknown time it takes for one communication step across all replicas. In partial synchrony, the network alternates intervals of synchrony, in which the bound $\delta < \Delta$ holds, to ones of asynchrony.

Replicas communicate via all-to-all authenticated links. We assume the existence of a public-key infrastructure (PKI), secure digital signatures, and collision-resistant hash functions. Additionally, we assume replicas have access to shared randomness, e.g., through a safe and live random beacon protocol [52].

4 Internet Computer Consensus

The slow path of the Banyan algorithm corresponds almost precisely to the Internet Computer Consensus protocol (ICC), which we thus introduce first.

Overview. The goal of the ICC protocol is to provide a total ordering (i.e., chain) of blocks among replicas in a network. As the protocol advances, a tree of blocks is constructed, starting from a genesis block that is at the root of the tree. Blocks are added to the block-tree if they are safe to be extended. Each replica may have a different, partial view of the block-tree. Replicas make progress, by *finalizing* at most one block per tree height. By traversing the tree edges, a path of finalized blocks is established. This path constitutes the blockchain. Honest replicas are guaranteed to have a common prefix of this path.

Creating a Block-Tree through Notarization. The ICC protocol proceeds in **rounds**. In the k-th round, one or more blocks are added to a block-tree at height k. To be added to the tree, a block must be *notarized*. In ICC, a block b becomes **notarized** for a replica, once at least $n - f$ *notarization votes* for block b are received. A **notarization vote** is a BLS signature sent by a replica u for a block b, implying that replica u validated block b. Notarization votes can be aggregated to a single multi-signature, that can be efficiently verified [11]. As soon as block b is validated by at least $n - f$ replicas, it becomes notarized, and can be added to the block-tree.

Block Proposal. In principle, in every round k, each replica can propose a block and all notarized blocks make it into the block-tree. Therefore, at any height k there can be up to n blocks. However, this makes it difficult for the replicas to agree on which block should be part of the blockchain, as at most one block per height should be included. To reduce the number of blocks proposed in each round, a *random beacon* is used to generate a random permutation of the n replicas. The permutation defines a different **rank** $r \in [0, n-1]$ for each replica for that particular round. The replica with the lowest rank, i.e., the **rank-0 replica**, is the **leader** of that round. At the beginning of a round, each replica computes its rank and starts a timer. The leader of that round proposes the block immediately, while the other replicas wait for a time directly proportional to their rank before proposing a block. Let r be the rank of a replica in a given round, the replica delays the proposal of its block by a proposal delay $\Delta_{prop}(r) = 2\Delta \times r$. In round k, when deciding which block in the block-tree to extend, the replica considers only notarized blocks at height $k - 1$.

Notarization. In round k, before sending a notarization vote for a block b proposed by a replica u with rank $r^{(u)}$, the replica v receiving the block b waits a notarization delay of $\Delta^{(u)}_{notary} = 2\Delta \times r^{(u)}$ after starting round k. Therefore, once replica v receives a block b from replica u, it checks if it is time to send a notarization vote for a block of rank $r^{(u)}$. If so, it broadcasts the notarization vote to all other replicas, otherwise, it waits until $\Delta^{(u)}_{notary}$ has passed. This way, blocks of lower rank should become notarized and added to the block-tree before others. Once a block b in round k becomes notarized for replica u, replica u broadcasts the notarization to other replicas, stops sending notarization votes for blocks of round k, and starts round $k + 1$. Thus, when the leader is honest, the network is in a phase of synchrony and the block proposal and notarization delays are set accordingly, only the block proposed by the rank-0 replica will be added to the tree at height k. If the leader is not honest or the network is asynchronous, some other replicas of higher rank may also propose blocks, and also have their blocks added to the tree.

Finalization. As there might be multiple notarized blocks at the same height, we must guarantee that all replicas agree on the same *finalized* block and that all *finalized* blocks are part of the same blockchain. ICC uses a mechanism similar to notarization to guarantee that no consistency violation across rounds is possible. If the replica did not broadcast notarization votes for any other block than b in the same round k, the replica will also broadcast another BLS-signature for block b, called **finalization vote**. If a replica u receives at least $n - f$ finalization votes for the same block b, the replica u can aggregate them into a **finalization**. At this point, the replica u is guaranteed that all replicas will eventually include block b in the blockchain. Once a finalization is created for a given block b, block b is said to be **explicitly finalized** and can be output. The finalization is then sent to all the other replicas.

The rule for broadcasting finalization votes implies that in some rounds, no block might become finalized by the aggregation of at least $n - f$ finalization votes. However, as soon as a block in a later round is explicitly finalized, all its ancestors in the tree, until the last explicitly finalized block, automatically become **implicitly finalized**. An illustration of the steps necessary before reaching finalization is shown in Figure 3.

We point out that if a block b at height k becomes finalized, no other block at height k can be notarized. Since only notarized blocks are extended (see beginning of Section 4), this implies that all blocks at height greater than k will have b as an ancestor and thus, all *finalized* blocks are part of the same blockchain. This sketches the safety proof of the ICC algorithm.

Remark 4.1 (Finalization Latency)**.** In rounds in which the network is synchronous, *finalization* of a block can be reached in three times the message delivery time $\delta < \Delta$. In the first round the block proposal, in the second the *notarization votes*, and in the third the *finalization votes* are broadcast (see Figure 1).

Remark 4.2 (Timeouts)**.** In the simplest implementation of the ICC protocol, we can assume that the communication delay bound Δ is an explicit parameter. In practice, instead,

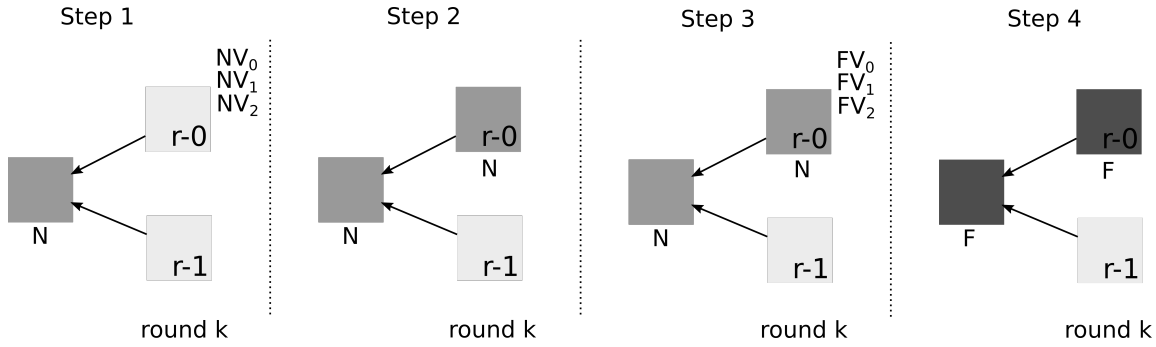


Figure 3. The diagram shows the steps necessary to reach the finalization of a block. Black blocks are *finalized*, dark gray blocks are *notarized*, while light gray blocks are not. In this example, $n = 4$ and $f = 1$. Initially, $n - f = 3$ replicas send notarization votes (NV) for the rank-0 block of round k. Replicas will not send a NV for higher rank proposals, in this round. As soon as the NV are received, the replica combines them in a notarization (N). Then, the replicas that only sent an NV for the rank-0 block will also send a finalization vote (FV) for it. Once the three FV are received, they will be combined into a finalization (F) and both the r-0 block and its ancestor(s) are finalized.

the protocol is modified to adaptively adjust to an unknown communication delay bound.

5 Problem Statement

The purpose of SMR protocols is to totally order blocks containing an arbitrary payload so that all replicas output the payload in the same order.

The properties we want the Banyan algorithm to satisfy, are first and foremost the same that Internet Computer Consensus provides [19]:

- **Deadlock Freeness**: Each round eventually terminates and increases the block-tree height by 1.
- **Safety**: If some honest replica finalizes block b in round k, and another honest replica finalizes block b' in round k, then $b = b'$.
- **Liveness**: If the network is momentarily synchronous and the leader is honest, then the block proposed by the leader is added to the block-tree and finalized.

Additionally, Banyan satisfies a stronger property:

- **Fast Termination**: If the network is momentarily synchronous, the leader is honest, and $n - p$ replicas behave momentarily honestly, then the block proposed by the leader is added to the block-tree and finalized in a single round trip time.

Remark 5.1. Deadlock freeness is also a *liveness* property. It provides chain growth even in periods of asynchrony, while Liveness guarantees that the entire progress is finalized in periods of synchrony. This distinction underlines the two modes of operation [19].

6 Intuition

The goal of the Banyan fast path is to finalize a block as soon as possible while guaranteeing agreement among replicas. In the ICC protocol a block is finalized at the fastest three communication steps after being proposed. The fast path added in Banyan reduces the finalization latency down to two communication steps, as shown in Figure 1.

Definition 6.1. In order to distinguish the two possible finalization scenarios, we call **Fast Path finalized** (or **FP-finalized**) the blocks that are explicitly finalized via the Banyan fast path, while we call **Slow Path finalized** (or **SP-finalized**) the blocks that are explicitly finalized by receiving at least $n - f$ finalization votes, as in the ICC protocol.

The idea behind the fast path is that if "enough" replicas send their **first** notarization votes for the same block b, i.e., replicas are in pre-agreement, then b can be immediately finalized without waiting for the finalization votes to be sent and received. In other words, replicas finalize blocks through finalization votes only if there are too many replicas that do not agree on the first block added to the block-tree at a given height, or if the slow path is faster.

Definition 6.2 (Fast Vote)**.** To determine if enough replicas are in pre-agreement, and try to finalize a block via the fast path, replicas broadcast a **fast vote** for each round's first block they send a notarization vote for. A block that receives $n - p$ fast votes becomes explicitly finalized via the fast path, and fast votes can be combined into a **fast finalization**.

Safety considerations. As we have sketched in Section 4, the rules of ICC ensure that if a block b at height k becomes SP-finalized, no other block at the same height will be notarized. This guarantees that the tree will not contain any block at height k besides block b. Therefore, only block b will be extended. In Banyan however, if a block b gets *FP-finalized*, we cannot guarantee that there will not be another notarized blocks at the same height.

Hence, we introduce a new concept, similar to notarization. Intuitively, we want to guarantee that whenever a block b at

height k gets FP-finalized, then no other block at height k can be extended. To this end, each replica marks blocks as *unlocked* when they are safe to be extended.

As we will show, the conditions of a block being *unlocked* almost never restrict the original ICC algorithm (see Remark 8.3) and only a few additions are sufficient to guarantee the safety of Banyan.

7 The Banyan Algorithm

The Banyan algorithm extends the Internet Computer Consensus algorithm and enables *fast path finalization* in periods of synchrony. The protocol allows explicit finalization to be achieved as soon as $n-p$ *fast votes* are received. The fast path runs alongside the Internet Computer Consensus protocol and is integrated into existing messages. In case the fast path cannot be used, there is no "switching cost" to revert to the slow path. When there is no pre-agreement, blocks will still be notarized and eventually finalized (either explicitly or implicitly) just as in the ICC protocol.

The following definitions are written from the point of view of a replica u, at round k:

Definition 7.1. Let $\text{blocks}(k)$ be the set of blocks that have been received in round k. For a block $b \in \text{blocks}(k)$, we define $\text{supp}(b)$ to be the set of replicas from which u has received a fast vote. Further, for any set $\mathcal{B} \subseteq \text{blocks}(k)$, we define $\text{supp}(\mathcal{B})$ to be the set of *distinct* replicas from which u has received a *fast vote* for a block in $\mathcal{B}$:

$$\text{supp}(\mathcal{B}) = \bigcup_{b \in \mathcal{B}} \text{supp}(b)$$

Definition 7.2. At any given time, $\text{max}(k)$ is a rank-0 block, such that no other rank-0 block has a larger support:

$$\text{max}(k) = \left(\underset{b \in \text{blocks}(k), b.rank=0}{\arg\max} |\text{supp}(b)| \right).pop()$$

Remark 7.3. There can be multiple rank-0 blocks when the leader is Byzantine.

Definition 7.4. At any given time, $\text{nonLeaderBlocks}(k)$ is the set of blocks which u has received with a rank larger than 0:

$$\text{nonLeaderBlocks}(k) = \{b \in \text{blocks}(k) \mid b.rank \neq 0\}$$

Definition 7.5. At any given time, $\text{nonMaxBlocks}(k)$ is the set of blocks which u has received, excluding a rank-0 block that has the most support:

$$\text{nonMaxBlocks}(k) = \{b \in \text{blocks}(k) \mid b \neq \text{max}(k)\}$$

Using these definitions, we can pin down exactly what blocks are safe to extend for a replica u, in the presence of the fast path. We call such a block *unlocked*.

Definition 7.6 (Unlocked Blocks)**.** Finalized blocks are **unlocked** by definition. Additionally, at any point during a round k, for a valid block $b \in \text{blocks}(k)$:

1. if $|\text{supp}(b)| + |\text{supp}(\text{nonLeaderBlocks}(k))| > f + p$, then b is unlocked.
2. if $|\text{supp}(\text{nonMaxBlocks}(k))| > f + p$, all current and future blocks of round k are unlocked.

An example of the above definitions is presented in Figure 4.

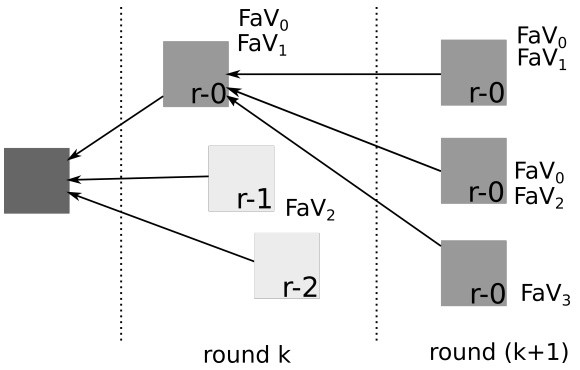


Figure 4. Fast votes (FaV) received for each block are shown in the block-tree. Black blocks are *finalized*, dark gray blocks are *unlocked*, while light gray blocks are not. Supposing $n = 4$, $f = 1$, and $p = 1$, for round k, Condition 1 is met, and the r-0 block is *unlocked*. Instead, for round $(k + 1)$, Condition 2 is met, and all blocks in this round are *unlocked*.

The Banyan algorithm is defined by the following changes to the slow path algorithm (ICC) presented in Section 4. Line numbers refer to Algorithms 1 and 2 that contain the full Banyan pseudocode.

Restriction 1	Block proposals can only extend an unlocked block. Similarly, notarization votes, fast votes, and finalization votes are only sent for blocks that extend an unlocked block. (We change the validity condition on line 67.)
Restriction 2	Replicas move to the next round once an unlocked block has been notarized, and they have sent a fast vote (line 53).
Addition 1	When replicas move to the next round, an *unlock proof* of the notarized block is broadcast (line 32).
Addition 2	When a block is proposed, an *unlock proof* of the parent block and a fast vote for the current block is broadcast alongside the proposed block (lines 29 and 32).
Addition 3	When the first round k notarization vote is broadcast, a fast vote for the same block is broadcast alongside it (line 43).
Addition 4	If a rank-0 block has received $n - p$ fast votes, it is considered FP-finalized (line 61). Fast votes are aggregated into a fast finalization and broadcast (lines 62 and 63).

Definition 7.7 (Unlock Proof)**.** An **unlock proof** (Addition 1) is the collection of valid fast votes, that prove that b is unlocked according to Definition 7.6.

Unlock proofs can be implemented naively by aggregating the fast votes for each block using BLS multi-signatures [11]. In the worst case, condition 2) of definition 7.6 might only be met after receiving $2f + 2p + 1$ fast votes, which might attest $f + p + 2$ unique blocks, leaving little improvements from aggregation. In practice, however, we expect only a few different blocks to receive fast votes. The description of a more targeted mechanism to achieve small unlock proofs in the worst case is left to future work.

Remark 7.8. It is possible to omit sending a corresponding notarization vote when a fast vote is sent. A notarization then consists of two multi-signatures, on for notarization and one for fast votes. For the sake of simplicity, we do not consider this version of Banyan in our description and analysis.

8 Protocol Analysis

In this section, we present proof sketches that explain the correctness of Banyan. These are not complete proofs but are intended to provide a conceptual understanding of the protocol's security.

8.1 Deadlock Freeness

The ICC protocol guarantees deadlock freeness as each honest replica receives at least one notarized block in each round. Therefore, each honest replica can start the next round and the block-tree keeps growing. However, in Banyan a block that was notarized is not guaranteed to be extended as, due to Restriction 1, in order for this to happen, the block must also be unlocked.

The following lemma shows how, regardless, deadlock freeness is not impacted. Intuitively, Restriction 2 makes sure that replicas only move to a higher round once a block can be extended, and Addition 1 guarantees that each replica does so eventually.

Lemma 8.1. *If each honest replica broadcasts a fast vote in round k, then Banyan guarantees that each honest replica eventually observes at least one unlocked block.*

Proof. Assume towards contradiction that in round k no block is unlocked for an honest replica u. Consider the set $\mathcal{S}$, defined to be the support for leader blocks different from $\textsc{Max}(k)$, i.e.,

$$\mathcal{S} = \textsc{supp}(\textsc{nonMaxBlocks}(k) \setminus \textsc{nonLeaderBlocks}(k))$$

If $\mathcal{S} = \emptyset$, a contradiction is reached immediately, as

$$|\textsc{supp}(\textsc{max}(k))| + |\textsc{supp}(\textsc{nonLeaderBlocks}(k))|$$
$$= |\textsc{supp}(\textsc{blocks}(k))| > n - f > f + p$$

and thus by Item 1 of Definition 7.6 $\textsc{max}(k)$ would be unlocked. (This corresponds to rounds with an honest leader.)

Algorithm 1 Banyan (Part 1)

Implements:
  Pseudocode for round k at replica u

Uses:
  RandomBeacon, **instance** beacon
  BestEffortBraodcast, **instance** broadcast

Parameters:
  Δ_{prop} ▷ Proposal delay
  Δ_{notary} ▷ Notarization delay
  k ▷ Current round number
  $kMax$ ▷ Highest round finalized so far
  $payload$ ▷ Set of transactions for this round
  r_u ▷ Rank, permutation derived from beacon
  b_p ▷ Notarized and unlocked round $(k-1)$ block

upon event ⟨banyan.Init⟩ **do**
  $fastVoteSent \leftarrow$ `False`;
  $proposed \leftarrow$ `False`; ▷ Proposed a block
  $t_0 \leftarrow clock()$; ▷ Time at the start of the round
  $D \leftarrow \emptyset$; ▷ Set of disqualified ranks
  $N \leftarrow \emptyset$; ▷ Set of blocks for which a notarization vote was sent

upon $\neg proposed$ **and** $clock() \geq t_0 + \Delta_{prop}(r_u)$ **do**
  create a new round k block:
  $b \leftarrow (k, u, hash(b_p), payload, signature_u)$
  $proposed \leftarrow$ `True`
  if $r_u = 0$ **then**
    broadcast b, b_p's notarization, b_p's unlock proof, fast vote for b
    $fastVoteSent \leftarrow$ `True`
  end if
  broadcast b, b_p's notarization, b_p's unlock proof

upon exists *a valid round k block b of rank r* **such that** $b \notin N$, **and** $r \notin D$ **and** $clock \geq t_0 + \Delta_{notary}(r)$ **and** $\nexists$ *valid round k block b′ of rank r′* $\in [r] \setminus D$ **do**
  if $r_u \neq r$ **then**
    broadcast b, b_p's notarization, b_p's unlock proof
  end if
  if some block in N has rank r **then**:
    $D \leftarrow D \cup \{r\}$;
  else
    $N \leftarrow N \cup \{b\}$
    if $\neg fastVoteSent$ **then**:
      broadcast fast and notarization vote for b;
      $fastVoteSent \leftarrow$ `True`;
    else
      broadcast notarization vote for b
    end if
  end if

Algorithm 2 BANYAN (Part 2)

50: **upon exists** *a valid round k block b* **such that** *b not notarized* **and** $\lceil \frac{n+f+1}{2} \rceil$ *notarization votes received* **do**
51: **combine** notarization votes into a notarization for b
52:
53: **upon exists** *a valid round k block b* **such that** *b is notarized* **and** *b is unlocked* **and** $fastVoteSent$ **do**
54: **combine** fast votes into a unlock proof
55: **broadcast** notarization and unlock proof for b
56: **if** $N \subseteq \{b\}$ **then**
57: **broadcast** finalization vote for b
58: **end if**
59: $k \leftarrow k + 1$ ▷ Move to next round
60:
61: **upon exists** *(fast) finalization for a round k block b with* $k > kMax$ **or** *receive* $\lceil \frac{n+f+1}{2} \rceil$ *finalization votes* **or** (*receive* $n - p$ *fast finalization votes* **and** $b.rank = 0$) **such that** $valid(b)$ **and** *b is not finalized* **do**
62: **combine** (fast) finalization votes into a (fast) finalization for b, if necessary
63: **broadcast** (fast) finalization for b
64: **output** payloads of the last $k - kMax$ blocks in the chain ending at b
65: $kMax \leftarrow k$
66:
67: **procedure** $valid(b)$ **is**
68: **return** b extends a notarized and unlocked round $(k-1)$ block b_p **and** b is signed correctly **and** contains a fast vote from the proposer if $b.rank = 0$
69:

Instead, if $\mathcal{S} \neq \emptyset$, there are at least two rank-0 blocks. By Addition 2 (Line 29), they each contain a fast vote from the (Byzantine) leader. Since u also receives fast votes from $n - f \geq 2f+2p-1$ honest replicas, at least $2f+2p+1$ fast votes will be received, i.e., $|\text{SUPP}(\text{MAX}(k) \cup \text{NONMAXBLOCKS}(k))| = 2f + 2p + 1$. By the pigeonhole principle, either $|\text{SUPP}(\text{MAX}(k))| \geq f + p + 1$ or $|\text{SUPP}(\text{NONMAXBLOCKS}(k))| \geq f + p + 1$. In both cases at least one block is unlocked according to Definition 7.6, thus leading to a contradiction. □

Theorem 8.2. *BANYAN satisfies* ***deadlock freeness****.*

Proof. We prove that each round eventually terminates, and that the block-tree of each honest replica keeps growing in height by induction on the block-tree height k. Specifically, we show that for each round k, at least one notarized and unlocked block will be added to each honest replica's block-tree.

Base case: All honest replicas agree on the root of the block-tree, the genesis block, which is also notarized and finalized by definition. By Definition 7.6 the genesis block is thus unlocked. Thus, in round 0, a notarized and unlocked block is added to each replica's block-tree.

Induction step: Assuming a notarized and unlocked block exists in round k, we prove that each replica observes a notarized and unlocked block in round $k+1$. By the induction hypothesis, all replicas will broadcast a fast vote and thus execute the procedure that starts on line 53. Thus, all honest replicas will enter round $k+1$.

Each honest replica will receive at least one round $k+1$ block, together with a notarization and unlock proof for the extended parent block (Addition 2, line 32). Thus, each honest replica will send a fast vote for one block (Addition 3, line 43). Thus, by Lemma 8.1, at least one block round $k+1$ block will be unlocked. As no honest replica moves to a higher round without an unlocked block being notarized (Restriction 2, line 53), and replicas keep notarizing blocks, one unlocked block will be notarized. Finally, honest replicas moving from one round to the next broadcast the notarization and unlock proof (Addition 1, line 55), guaranteeing that all honest replicas can add a notarized and unlocked block to their block-tree. □

Remark 8.3. BANYAN is at least as fast as ICC. Any scenario in which a notarized block for round k exists before it is unlocked can only occur due to message reordering. Hence, Restriction 1 and Restriction 2 cannot cause latency concessions if the communication channel precludes message reordering (which is the case in practice when TCP/QUIC is used).

8.2 Safety

ICC derives its safety from the fact that in rounds with an explicetly finalized block, no other blocks can be notarized. We start our way towards sketching the safety proof of BANYAN by showing that an equivalent property holds for BANYAN.

Lemma 8.4. *If a round k block b is SP-finalized, then BANYAN guarantees that no round k block* b', $b' \neq b$ *is notarized for any replica.*

Proof. Let block b be SP-finalized by some replica, and let block b', $b' \neq b$ be notarized. A replica must have received a quorum of at least $\lceil \frac{n+f+1}{2} \rceil$ finalization votes for b, $\lceil \frac{n-f+1}{2} \rceil$ of which are from honest replicas. Moreover, some replica must have received a quorum of $\lceil \frac{n+f+1}{2} \rceil$ notarization votes for b', $\lceil \frac{n-f+1}{2} \rceil$ of which are from honest replicas. The two quorums must be disjoint, since each honest replica sends a finalization vote for b only if it did not send any notarization vote for other blocks (in particular b') at height k (line 56). Thus, there must be at least $\lceil \frac{n-f+1}{2} \rceil + \lceil \frac{n-f+1}{2} \rceil \geq n - f + 1$ honest replicas. By definition, there are only $n - f$ honest replicas,

hence one honest replica must occur in both quorums, a contradiction. □

Therefore, any other block at height k will be neither extended nor finalized (as both cases require it to be at least notarized). Similarly, let b be a block at height k which is FP-finalized, we prove that b is the only unlocked block at height k.

Lemma 8.5. *If a round k block b is FP-finalized, then Banyan guarantees that no round k block b', $b' \neq b$ is unlocked for any replica.*

Proof. Assume towards contradiction that there exists an FP-finalized block b and an unlocked block b', $b' \neq b$ at height k. For b' to be unlocked, either Condition 1 or Condition 2 of Definition 7.6 must be satisfied.

If Condition 1 is satisfied, then

$$|\text{supp}(b')| + |\text{supp}(\text{nonLeaderBlocks}(k))| > f + p$$

This implies that blocks different from b have received more than $f + p$ fast votes, more than p of which coming from honest replicas. By lines 42 to 44, each honest replica broadcasts at most one fast vote per round. Thus b received less than $n - p$ fast votes, a contradiction, as $n - p$ are required for FP-finalization (line 61).

If Condition 2 is satisfied, then

$$|\text{supp}(\text{nonMaxBlocks}(k))| > f + p$$

It follows that $|\text{supp}(\text{max}(k))| < n - p$. This implies that no block can be FP-finalized, a contradiction. □

Theorem 8.6. *Banyan satisfies **safety**.*

Proof. Towards contradiction, assume an honest replica finalizes round k block b, while another honest replica finalizes round k block b', $b' \neq b$. Without loss of generality, we assume b was explicitly finalized.

Consider the case where b and b' were both explicitly finalized. SP-and FP-finalization imply the reception of $\lceil \frac{n+f+1}{2} \rceil$ unique notarization and fast votes for a block, respectively. This implies the existence of two respective Byzantine quorums, a contradiction [18]. Thus, without loss of generality, assume b' was implicitly finalized. If b was SP-finalized, by Lemma 8.4 b' cannot have been notarized, and thus no honest replica would have extended b'. Instead, if b was FP-finalized, by Lemma 8.5 then b' was not unlocked. Thus, honest replicas would have sent fast, notarization, or finalization votes for a block that was not unlocked, a contradiction (see line 34 and the validity definition on line 67). □

8.3 Liveness

We define liveness such that whenever the network is synchronous for a "long enough" interval, if the leader is honest, only the leader's block will be added to the block-tree for the corresponding round and this block will be finalized by all honest replicas.

Theorem 8.7. *Banyan satisfies **liveness**.*

Proof. Let T be the time when the first honest replica u enters round k. Suppose that the leader l of round k is honest and proposes block b. Moreover, suppose that the network is δ-synchronous between time T and $T + 2\delta$. Furthermore, assume $\delta \leq \Delta$, which implies that $\Delta_{notary}(1) \geq 2\delta$, with the functions Δ_{notary} and Δ_{prop} as defined in Section 4. We want to prove that under these assumptions, eventually, each honest replica will finalize block b.

Before the honest replica u enters round k at time T, the replica u notarized an unlocked block in rounds $1, \ldots, k-1$ and broadcast the notarization and unlock proof (Addition 1, line 55). By the synchrony assumption, the other replicas, l included, receive the notarization and unlock proof for rounds $1, \ldots, k-1$ by time $T + \delta$, and enter round k.

Note that $\Delta_{prop}(0) = 0$. Thus, replica l broadcasts b by time $T + \delta$, and other replicas will receive it by $T + 2\delta$. By assumption, no other honest replica enters round k before time T and as $\Delta_{notary}(1) \geq 2\delta$, every honest replica will broadcast a fast vote and a notarization vote for b (Addition 3, line 43). Therefore, all honest replicas eventually receive at least $n - f$ fast votes for b. When $p = f$, these fast votes are enough to FP-finalize b (Addition 4, line 61). When $p < f$, $\lceil \frac{n+f+1}{2} \rceil \leq n - f$ notarization votes are sufficient to notarize b instead. As argued previously, honest replicas will not have notarized another block and thus will broadcast a finalization vote (line 57). Eventually, after an additional communication round, honest replicas will receive $\lceil \frac{n+f+1}{2} \rceil \leq n - f$ finalization votes, which leads to SP-finalization (line 61). □

8.4 Fast Termination

Theorem 8.8. *Banyan satisfies **fast termination**.*

Proof. With the additional assumption that $n - p$ replicas behave honestly during a phase of synchrony, we can guarantee $n - p$ fast votes for the rank-0 block reaching every replica. Thus, after a single round-trip time, each replica will be able to FP-finalize the rank-0 block. □

9 Evaluation

9.1 Implementation

We implement Banyan in Golang, utilizing the Bamboo BFT framework, which is designed for prototyping and evaluating chained BFT algorithms [33]. This framework already supports the two well-known protocols, Streamlet [22] and HotStuff [61].

To enhance the performance and reliability of the framework, we introduced a few modifications. For instance, by using non-blocking message-passing channels internally, and by forwarding blocks that extend the tip of the chain, we drastically improve the performance of all algorithms implemented with Bamboo. The impact of the latter change is

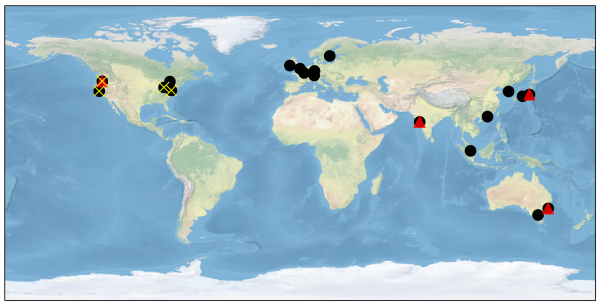


Figure 5. The red triangles mark the coordinates of the 4 AWS datacenters used in Section 9.3, while the yellow crosses show the coordinates of 4 datacenters in the US used in Section 9.4. The 19 different coordinates of the deployed EC2 instances from Section 9.5 are marked by black circles.

especially surprising to us, as originally only Streamlet pursued this strategy in the provided implementation, leading to an unexpected advantage over all other protocols, both in terms of throughput and latency. Henceforth, we have been careful to treat all protocols equally.

To further increase predictability and transparency of the protocol operations, we have replaced the random beacon leader election with a round-robin rotation. Our code and benchmarking scripts are made openly available [53].

9.2 Methodology

Application subnets of the Internet Computer feature 13 replicas [28]. To test the potential of Banyan as a drop-in replacement for ICC, and since $n = 19$ is optimal for both $f = 6$, $p = 1$ and $f = 4$, $p = 4$ experimental scenarios, we perform experiments with up to 19 replicas around the globe (see Figure 5). Replicas run on a testbed comprised of AWS `t3.large` EC2 instances (with 2vCPUs, 8 GB of memory, running Ubuntu 22.04).

We wish to demonstrate that (i) requiring only two rounds of communication instead of three has a positive impact and does not increase variance, while (ii) not suffering any more throughput or latency degradation than ICC under crash-faults. Finally, we want to show (iii) that Banyan is performing consistently as well as or better than ICC and other state-of-the-art protocols in a global network topology.

As the goal of our evaluation is to measure the precise impact of our changes, we define latency as the average proposal finalization time, measured at the respective proposer using their system clocks. In turn, throughput is measured as the average number of committed bytes per second at any (non-faulty) replica. We measure the protocols' behavior under varying load by controlling the size of the payload, where the payload is a bit vector randomly generated by the leader.

As required by the protocol, we set the proposal (Δ_{prop}) and notarization (Δ_{notary}) delays for Banyan and ICC to be larger than the message delay experienced without network disruptions, such that our experimental results correspond to regular network conditions and only one block is proposed per round if there are no faults. We proceed in the same way to set appropriate timeouts for HotStuff and Streamlet. Each experiment is run for 120 seconds, which is shown to be sufficient, as the measurements show remarkable regularity (e.g., see Figure 6c).

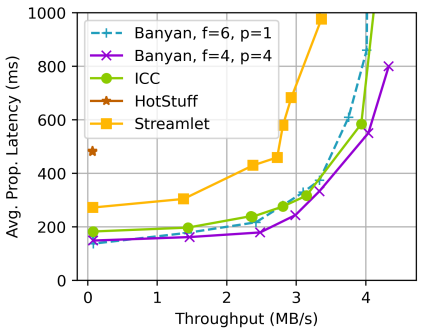
9.3 Performance Evaluation

We distribute 19 replicas across 4 datacenters, as shown by the red triangles in Figure 5. Each datacenter hosts 5 replicas, except for one that hosts only 4. Figure 6a shows the throughput with varying block sizes. Primarily, we note that Banyan with $p = 1$ performs consistently better or as well as ICC. For blocks of size 400KB, ICC averages a proposal finalization time of 239ms, while Banyan $p = 1$ averages a finalization time of 216ms. This improvement of about 10% is expected to be less than the theoretical maximum of 33%, because Banyan must hear from all the data centers in the fast path. In the slow path, replicas can collect two consecutive quorums without needing to communicate with the furthest datacenter. This observation, namely that more communication rounds with smaller quorum sizes are sometimes faster in practice than fewer rounds with larger quorum sizes, has been made before in similar contexts [39, 56]. Moreover, by setting $p = f = 4$, we observe that Banyan now has an average proposal finalization time of 179ms at the same block size, an improvement of 25.1%, that is much closer to the theoretical maximum of 33%. We suggest that this is due to the situation where the 4 co-located replicas are the furthest, in which case the fast path is employed, and thus lowers the average latency.

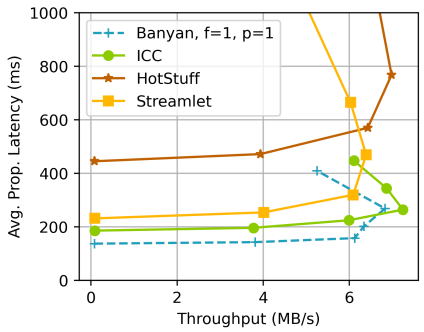
Next, we measure the potential of the Banyan fast path with a low number of replicas, i.e., we use only a single replica at each datacenter ($n = 4$). Crucially, in this case the fast path fires with the same conditions as regular notarization, i.e., after receiving just 3 replies. Figure 6b shows the throughput with block size increments of 500KB. At block sizes of 1MB, ICC averages a proposal finalization time of 224ms, which Banyan manages to improve by 29.9% to 157ms. We emphasize that this large performance increase does not come at the cost of higher variance in latency, as is shown in Figure 6c.

9.4 Effect of Crash-faults

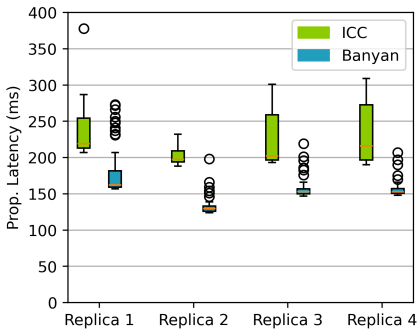
In this experiment, we show the effect that crashes have on Banyan and Internet Computer Consensus. We distribute 19 replicas across four US AWS datacenters, as shown by the yellow crosses in Figure 5. Results are presented in Figure 6d. As has been noted in literature, rotating leader protocols are sensitive to crashes (or malicious behavior), as at least one full timeout duration must expire before progress can be made in the case of a faulty leader. The timeout parameter



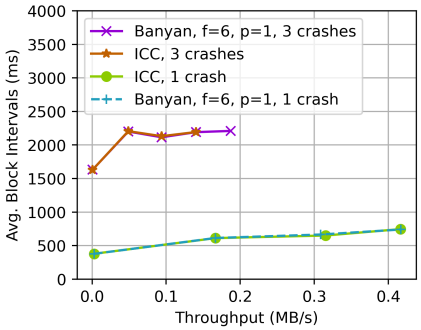
(a) Throughput vs. proposal latency for n=19 replicas spread across 4 global datacenters.



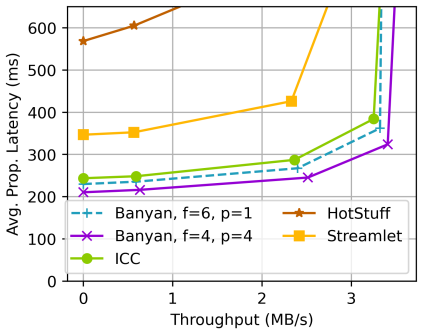
(b) Throughput vs. proposal latency for n=4 replicas spread across 4 global datacenters.



(c) Variance of Banyan and ICC proposal latencies compared with 1 MB payload and n=4.



(d) Effect of crash-faults on throughput and block intervals for n=19 replicas, spread across 4 US datacenters.



(e) Throughput vs. proposal latency for n=19 replicas spread across a global network.

Figure 6. Evaluation results.

will crucially impact the results of such scenarios. In the presented experiment, the timeout was set to 3 seconds. As seen in Figure 6d, there are no penalties in trying to take the fast path. When they are failures, the performance of Banyan is exactly the one of ICC.

9.5 Global Network

In this experiment, we simulate a worldwide deployment. We include almost all AWS datacenter locations available to us and distribute 19 replicas across the globe, as shown by the black circles in Figure 5. In this setting and with 1MB payloads, the average proposal finalization time measured for the ICC implementation was 384ms. By running Banyan with $f = 6, p = 1$, the average proposal finalization time is reduced by 5.8% to 362ms for "free". Further, for Banyan with $f = 4, p = 4$, the number drops further by 16% to 324ms (Figure 6e).

10 Conclusion

The proposed Banyan protocol improves upon current BFT protocols by providing a simultaneous dual mode, without incurring extra cost, thus closing the gap between state-of-the-art SMR protocols and classical fast consensus literature.

We show that it is possible to achieve optimally fast proposal finalization time for a rotating leader SMR protocol. We experimentally demonstrate the effectiveness of the fast path by showing consistent latency improvements compared to ICC, HotStuff and Streamlet. Ultimately, the effectiveness of the fast path depends largely on the network topology, as well as the parameter $p \in [0, f]$. Still, as we can set $p = 1$ at essentially no cost, we hypothesize that Banyan can have large positive impact in practice, especially in networks with fewer replicas. In the context of permissioned blockchains, and future decentralized Layer-2 sequencers, this setting is undoubtedly compelling, since it is possible to detect adversarial behavior and remove misbehaving replicas [17, 19].

11 Acknowledgements

We thank the anonymous Middleware reviewers for their extensive and valuable reviews, that among other things lead to the improvement of the resilience of Banyan. We further extend our gratitude to Prof. Christian Cachin for his insightful feedback.

References

[1] Ittai Abraham, Guy Gueta, Dahlia Malkhi, Lorenzo Alvisi, Rama Kotla, and Jean-Philippe Martin. 2017. Revisiting fast practical byzantine fault tolerance. *arXiv preprint arXiv:1712.01367* (2017).

[2] Ittai Abraham, Guy Gueta, Dahlia Malkhi, and Jean-Philippe Martin. 2018. Revisiting fast practical byzantine fault tolerance: Thelma, velma, and zelma. *arXiv preprint arXiv:1801.10022* (2018).

[3] Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. 2021. Good-case latency of byzantine broadcast: A complete categorization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*. 331–341.

[4] Ittai Abraham, Kartik Nayak, and Nibesh Shrestha. 2021. Optimal good-case latency for rotating leader synchronous bft. *Cryptology ePrint Archive* (2021).

[5] Yair Amir, Brian Coan, Jonathan Kirsch, and John Lane. 2010. Prime: Byzantine replication under attack. *IEEE transactions on dependable and secure computing* 8, 4 (2010), 564–577.

[6] Pierre-Louis Aublin, Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. 2015. The next 700 BFT protocols. *ACM Transactions on Computer Systems (TOCS)* 32, 4 (2015), 1–45.

[7] Zeta Avarikioti, Lioba Heimbach, Roland Schmid, Laurent Vanbever, Roger Wattenhofer, and Patrick Wintermeyer. 2023. FnF-BFT: A BFT protocol with provable performance under attack. In *Structural Information and Communication Complexity: 30th International Colloquium, SIROCCO 2023, Alcalá de Henares, Spain, June 6–9, 2023, Proceedings*. Springer, 165–198.

[8] Kushal Babel, Andrey Chursin, George Danezis, Lefteris Kokoris-Kogias, and Alberto Sonnino. 2023. Mysticeti: Low-Latency DAG Consensus with Fast Commit Path. *arXiv preprint arXiv:2310.14821* (2023).

[9] Michael Ben-Or. 1983. Another advantage of free choice (extended abstract) completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*. 27–30.

[10] Sam Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, et al. 2023. *Sui Lutris: A Blockchain Combining Broadcast and Consensus.* Technical Report. Technical Report. Mysten Labs. https://sonnino. com/papers/sui-lutris. pdf.

[11] Dan Boneh, Manu Drijvers, and Gregory Neven. 2018. Compact multi-signatures for smaller blockchains. In *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II*. Springer, 435–464.

[12] Gabriel Bracha. 1987. Asynchronous Byzantine agreement protocols. *Information and Computation* 75, 2 (1987), 130–143.

[13] Francisco Brasileiro, Fabíola Greve, Achour Mostéfaoui, and Michel Raynal. 2001. Consensus in one communication step. In *Parallel Computing Technologies: 6th International Conference, PaCT 2001 Novosibirsk, Russia, September 3–7, 2001 Proceedings 6*. Springer, 42–50.

[14] Ethan Buchman. 2016. *Tendermint: Byzantine fault tolerance in the age of blockchains.* Ph. D. Dissertation. University of Guelph.

[15] Yehonatan Buchnik and Roy Friedman. 2019. Fireledger: A high throughput blockchain consensus protocol. *arXiv preprint arXiv:1901.03279* (2019).

[16] Vitalik Buterin. 2013. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper.

[17] Vitalik Buterin and Virgil Griffith. 2017. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437* (2017).

[18] Christian Cachin, Rachid Guerraoui, and Luís Rodrigues. 2011. *Introduction to Reliable and Secure Distributed Programming* (2nd ed.). Springer Publishing Company, Incorporated.

[19] Jan Camenisch, Manu Drijvers, Timo Hanke, Yvonne-Anne Pignolet, Victor Shoup, and Dominic Williams. 2022. Internet computer consensus. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*. 81–91.

[20] Miguel Castro, Barbara Liskov, et al. 1999. Practical byzantine fault tolerance. In *OsDI*. 173–186.

[21] Benjamin Y Chan and Rafael Pass. 2023. Simplex Consensus: A Simple and Fast Consensus Protocol. Cryptology ePrint Archive, Paper 2023/463. https://eprint.iacr.org/2023/463 https://eprint.iacr.org/2023/463.

[22] Benjamin Y Chan and Elaine Shi. 2020. Streamlet: Textbook streamlined blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. 1–11.

[23] TH Hubert Chan, Rafael Pass, and Elaine Shi. 2018. Pala: A simple partially synchronous blockchain. *Cryptology ePrint Archive* (2018).

[24] Allen Clement, Edmund Wong, Lorenzo Alvisi, Mike Dahlin, and Mirco Marchetti. 2009. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation* (Boston, Massachusetts) *(NSDI'09)*. USENIX Association, USA, 153–168.

[25] Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. 2018. DBFT: Efficient Leaderless Byzantine Consensus and its Application to Blockchains. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. 1–8. https://doi.org/10.1109/NCA.2018.8548057

[26] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 910–927.

[27] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*. 34–50.

[28] Dfinity. Accessed 2024. Subnets - ICP Dashboard. https://dashboard.internetcomputer.org/subnets.

[29] Dfinity. Accessed 2024. World Computer build on the network. https://internetcomputer.org/.

[30] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. 1988. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)* 35, 2 (1988), 288–323.

[31] Filecoin. Accessed 2024. Filecoin - A decentralized storage network. https://filecoin.io/.

[32] Roy Friedman, Achour Mostefaoui, and Michel Raynal. 2005. Simple and efficient oracle-based consensus protocols for asynchronous Byzantine systems. *IEEE Transactions on Dependable and Secure Computing* 2, 1 (2005), 46–56.

[33] Fangyu Gai, Ali Farahbakhsh, Jianyu Niu, Chen Feng, Ivan Beschastnikh, and Hao Duan. 2021. Dissecting the performance of chained-bft. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 595–606.

[34] Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. 2022. Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*. Springer, 296–315.

[35] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*. 51–68.

[36] Rachid Guerraoui and Marko Vukolić. 2007. Refined quorum systems. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*. 119–128.

[37] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. 2019. Sbft: a scalable and decentralized trust infrastructure. In *2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 568–580.

[38] Mohammad M Jalalzai, Jianyu Niu, Chen Feng, and Fangyu Gai. 2020. Fast-hotstuff: A fast and resilient hotstuff protocol. *arXiv preprint arXiv:2010.11454* (2020).

[39] Flavio Junqueira, Yanhua Mao, and Keith Marzullo. 2007. Classic paxos vs. fast paxos: caveat emptor. *Proceedings of USENIX Hot Topics in System Dependability (HotDep)* (2007).

[40] Dakai Kang, Suyash Gupta, Dahlia Malkhi, and Mohammad Sadoghi. 2024. HotStuff-1: Linear Consensus with One-Phase Speculation. *arXiv preprint arXiv:2408.04728* (2024).

[41] Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. 2021. All you need is dag. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*. 165–175.

[42] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. 2007. Zyzzyva: speculative byzantine fault tolerance. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*. 45–58.

[43] Klaus Kursawe. 2002. Optimistic byzantine agreement. In *21st IEEE Symposium on Reliable Distributed Systems, 2002. Proceedings.* IEEE, 262–267.

[44] Petr Kuznetsov, Andrei Tonkikh, and Yan X Zhang. 2021. Revisiting Optimal Resilience of Fast Byzantine Consensus. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC'21)*. Association for Computing Machinery, New York, NY, USA, 343–353. https://doi.org/10.1145/3465084.3467924

[45] Leslie Lamport. 2003. Lower bounds for asynchronous consensus. *Future Directions in Distributed Computing: Research and Position Papers* (2003), 22–23.

[46] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (1982), 382–401.

[47] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. 2015. Inclusive block chain protocols. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*. Springer, 528–547.

[48] Dahlia Malkhi and Kartik Nayak. 2023. HotStuff-2: Optimal Two-Phase Responsive BFT. *Cryptology ePrint Archive* (2023).

[49] Dahlia Malkhi, Chrysoula Stathakopoulou, and Maofan Yin. 2023. BBCA-CHAIN: One-Message, Low Latency BFT Consensus on a DAG. *arXiv preprint arXiv:2310.06335* (2023).

[50] J.-P. Martin and L. Alvisi. 2006. Fast Byzantine Consensus. *IEEE Transactions on Dependable and Secure Computing* 3, 3 (2006), 202–215. https://doi.org/10.1109/TDSC.2006.35

[51] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf.

[52] Mayank Raikwar and Danilo Gligoroski. 2022. Sok: Decentralized randomness beacon protocols. In *Australasian Conference on Information Security and Privacy*. Springer, 420–446.

[53] Jakub Sliwinski and Yann Vonlanthen. 2024. Banyan. https://github.com/kobi-lizard/banyan.

[54] Jakub Sliwinski, Yann Vonlanthen, and Roger Wattenhofer. 2022. Consensus on demand. In *Stabilization, Safety, and Security of Distributed Systems: 24th International Symposium, SSS 2022, Clermont-Ferrand, France, November 15–17, 2022, Proceedings*. Springer, 299–313.

[55] Yee Jiun Song and Robbert van Renesse. 2008. Bosco: One-step byzantine asynchronous consensus. In *Distributed Computing: 22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008. Proceedings 22*. Springer, 438–450.

[56] João Sousa and Alysson Bessani. 2015. Separating the WHEAT from the chaff: An empirical design for geo-replicated state machines. In *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 146–155.

[57] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. 2022. Bullshark: The Partially Synchronous Version. *arXiv preprint arXiv:2209.05633* (2022).

[58] Chrysoula Stathakopoulou, Tudor David, and Marko Vukolic. 2019. Mir-bft: High-throughput bft for blockchains. *arXiv preprint arXiv:1906.05552* (2019), 92.

[59] Chrysoula Stathakopoulou, Matej Pavlovic, and Marko Vukolić. 2022. State machine replication scalability made simple. In *Proceedings of the Seventeenth European Conference on Computer Systems*. 17–33.

[60] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. 2009. Spin one's wheels? Byzantine fault tolerance with a spinning primary. In *2009 28th IEEE International Symposium on Reliable Distributed Systems*. IEEE, 135–144.

[61] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. 2019. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. 347–356.