Quintus: Two-round Good-case Information Theoretic BFT for $n = 5f + 1$

Chenyang Liu[1], Dahlia Malkhi[2], Kartik Nayak[1], and Nibesh Shrestha[3]

[1] Duke University, Durham, NC, USA
chenyang.liu@duke.edu, kartik@cs.duke.edu
[2] University of California, Santa Barbara, CA, USA
dahliamalkhi@ucsb.edu
[3] Independent Researcher
nibeshrestha2@gmail.com

Abstract. We present Quintus, information-theoretic BFT protocols for tolerating $f < n/5$ Byzantine faults among n parties. We present two protocols:

1. The first protocol, Quintus-Fixed, is in a fixed view regime where views advance at a cadence 3Δ time. This protocol incurs a good-case latency of 2δ time where δ indicates actual network delay and message complexity of $O(n^2)$ in a view.
2. The second protocol, Quintus-Responsive, is an optimistically responsive protocol with good-case latency of 2δ time, $O(n^2)$ message complexity, and $2\Delta + 2\delta$ worst-case view latency where Δ denotes a pessimistic network delay parameter under synchrony.

1 Introduction

Byzantine consensus is a fundamental building block for fault-tolerant distributed systems and blockchains. In the partially synchronous model, classical results show that consensus can be achieved among $n = 3f + 1$ parties despite up to f Byzantine faults [14]. Modern protocols typically rely on digital signatures and quorum certificates to transfer evidence of previous decisions across "views". Using these approaches, they primarily optimize three key metrics: *good-case latency*, i.e., the time to commit a block when proposed by an honest "leader" and the network is synchronous, the *worst-case view latency*, i.e., the worst-case time after GST between the last party entering a view to all parties entering the next view, and *message complexity*, which is the number of messages sent by honest parties in a view.

In the presence of signatures, these protocols with $n = 3f + 1$ are able to obtain the optimal good-case latency of 3-rounds (i.e., 3 network hops) [4], a message complexity of $O(n^2)$ while requiring worst-case view latency of $3\Delta + \delta$ where Δ is a pessimistic network delay parameter that is respected for all messages when the network is synchronous [23,10,8,22], and δ represents the actual network delay. On the other hand, when tolerating $n = 5f + 1$, these

protocols achieve a good-case latency of 2-rounds [23,20,16,18,12], $O(n^2)$ message complexity while requiring a worst-case latency of $2\Delta + \delta$ [12].

Motivated by growing interest in both scaling down computational overhead and in removing reliance on digital signatures that may not be post-quantum secure, recent work has revisited the design of information-theoretic/signature-free Byzantine consensus protocols under partial synchrony [27,10,1,2,22,25,6,24]. These protocols replace cryptographic certificates with communication-based mechanisms, relying on authenticated channels to disseminate the state of the protocol. Prior known information-theoretic protocols operate at the optimal resilience of $n = 3f + 1$, and incur a 3-round good-case latency, which is known to be optimal when tolerating $f < n/3$ faults [27,10,1,2], and the best known protocol incurs a worst-case latency of $3\Delta + 2\delta$ [2].

In this work, we ask whether we can obtain *information-theoretic protocols with 2-round good-case latency for $n = 5f + 1$ while simultaneously minimizing worst-case view latency and message complexity?*

We answer this question affirmatively. We present two protocols, Quintus-Fixed and Quintus-Responsive, in two different regimes. The first regime is a "fixed view" regime where start times and end times of a view are fixed in advance and known to all parties. This regime is adopted by several practical blockchains such as Solana [18] and Ethereum [9]. The second regime is an "optimistically responsive" regime where the time spent in a view can change depending on network conditions and adversarial behavior. This regime is adopted by several practical blockchains such as Monad [17] and Espresso Systems [19].

Quintus-Fixed. Our first contribution is a protocol in the fixed view regime that proceeds through three phases of proposal, voting, and a state dissemination phase in each view. The protocol achieves 2-round good-case latency, a 3-round view, and incurs message complexity of $O(n^2)$ per view. More formally, our protocol obtains the following result:

Theorem 1 (Quintus-Fixed). *Assuming clocks are synchronized across parties, there exists an information-theoretic, deterministic, partially synchronous Byzantine broadcast protocol among n parties that tolerates f Byzantine faults for $n = 5f + 1$ in a fixed-view regime, that achieves a good-case latency of 2δ, has a view duration of 3Δ, and requires $O(n^2)$ message complexity per view.*

Quintus-Responsive. Our second contribution is an optimistically responsive protocol that advances at the speed of the network. Thus, instead of starting views at predetermined intervals, parties advance views earlier when messages from previous views arrive sooner. Such protocols tend to outperform fixed view protocols when the actual network delay $\delta \ll \Delta$. Quintus-Responsive achieves good-case 2δ latency, with $2\Delta + 2\delta$ worst-case latency, and incurs a message complexity of $O(n^2)$. More formally, we show:

Theorem 2 (Quintus-Responsive). *There exists an information-theoretic, deterministic, partially synchronous Byzantine broadcast protocol among n parties that tolerates f Byzantine faults with $n = 5f + 1$. The protocol achieves a*

good-case latency of 2δ*, worst-case view latency of* $2\Delta + 2\delta$*, incurs* $O(n^2)$ *message complexity, is optimistically responsive, and incurs optimistic* 2δ *latency between views.*

2 Preliminaries

We consider a system consisting of n parties $(P_1, \ldots, P_n)$, where up to $f < n/5$ parties may be Byzantine. Byzantine parties can behave arbitrarily. A party that remains fault-free throughout the execution and follows the protocol correctly is referred to as *honest*.

We assume the partial synchrony model introduced by Dwork et al. [14]. In this model, the system begins in an asynchronous state during which the adversary can arbitrarily delay messages between honest parties. After an unknown point in time, known as the *Global Stabilization Time* (GST), the network becomes synchronous: all messages sent by honest parties are guaranteed to be delivered within a fixed delay Δ. We use δ to denote the actual (and possibly variable) message transmission delays, and note that $\delta \leq \Delta$ after GST. Furthermore, we assume that parties' local clocks exhibit *no drift* but may have *arbitrary skew*.

Definition 1 (Partially Synchronous Byzantine Broadcast [4]). *A partially synchronous Byzantine broadcast protocol provides the following properties.*

- ***Agreement.*** *If two honest parties commit values* m *and* m' *respectively, then* $m = m'$.
- ***Validity.*** *If the designated broadcaster is honest and GST*$= 0$*, then all honest parties commit the broadcaster's value.*
- ***Termination.*** *All honest parties commit and terminate after GST.*

Measures. The *good-case latency* of a partially synchronous broadcast protocol is the time for all honest parties to commit (over all executions and adversarial strategies), given the designated broadcaster is honest and GST $= 0$; *worst-case view latency* is measured after GST from the time that the last honest party enters a view until all honest parties advance to the next view. *Message complexity* is the number of messages exchanged among honest parties in each view.

Optimistic Responsiveness. Responsiveness requires a consensus protocol to make progress in time proportional to the actual network delay (δ) and independent of any known upper bound delay when a leader is honest [21]. Optimistic responsiveness requires this same guarantee, but only when certain optimistic conditions hold. Several variations [26,15,5,1] have been formulated in the literature. We make use of the following definition in this paper.

Definition 2 (Optimistic Responsiveness [26]). *After GST, any correct leader, once designated, needs to wait just for the first* $n - f$ *responses to guarantee that it can create a proposal that will make progress. This includes the case where a leader is replaced.*

Table 1. Comparison of prior works on information-theoretic BFT protocols.

	Fault tolerance	Good-case latency	Worst-case view latency†	Message complexity	Optimistic Responsiveness
PBFT [10, Sec. 3]	$3f+1$	3δ	9Δ	$O(n^3)$	✓
IT-HS (blog) [25]	$3f+1$	4δ	10Δ	$O(n^2)$	×
IT-HS [6]	$3f+1$	6δ	$11\Delta+\delta$	$O(n^2)$	✓
TetraBFT [27]	$3f+1$	5δ	$9\Delta+\delta$	$O(n^2)$	✓
Simple-IT (Opt-RBC) [28]	$3f+1$	$5\delta/3\delta$	$8\Delta+4\delta$	$O(n^2)$	✓
Simple-IT (Bracha-RBC) [28]	$3f+1$	4δ	$5\Delta+2\delta$	$O(n^2)$	✓
Forget-IT [1]	$3f+1$	3δ	$4\Delta+3\delta$	$O(n^2)$	✓
IT-Kuplex (blog) [2]	$3f+1$	3δ	$3\Delta+2\delta$	$O(n^2)$	✓
IT-Kuplex (blog) [2]	$4f+1$	3δ	$3\Delta+\delta$	$O(n^2)$	✓
Floor-IT [3]	$5f-1$	2δ	$2\Delta+2\delta$	$O(n^2)$	×‡
Quintus-Fixed	$5f+1$	2δ	3Δ	$O(n^2)$	×
Quintus-Responsive	$5f+1$	2δ	$2\Delta+2\delta$	$O(n^2)$	✓

†: Entries of previous works expressed solely in Δ are not reported by the original papers; we infer them from each protocol's timeout and the number of rounds in its view change. All of them are substantially larger than those of Quintus.
‡: We note that Floor-IT [3] claims optimistic responsiveness under a weaker definition that measures latency only from the leader's proposal, after which the protocol completes in $O(\delta)$ time in the good case. However, the leader's proposal itself is preceded by an $\Omega(\Delta)$ waiting period.

We note that in [26], the term "make progress" means that all honest nodes will vote for the correct (honest) leader's proposal.

3 Related Work

There has been a long line of work on designing efficient BFT protocols in the information-theoretic setting. Table 1 compares our protocols with prior information-theoretic BFT protocols. In comparison with these works, both of our protocols tolerate $f < n/5$ Byzantine faults while achieving $O(n^2)$ message complexity and 2δ good-case latency. Furthermore, Quintus-Fixed achieves a worst-case view latency of 3Δ. Similarly, Quintus-Responsive achieves a worst-case view latency of $2\Delta + 2\delta$ while providing optimistic responsiveness.

PBFT [10, Sec. 3] achieves a good-case latency of 3δ under the optimal resilience bound $n = 3f+1$. However, its view-change phase incurs $O(n^3)$ message complexity, and its worst-case view latency is 9Δ.

Information-Theoretic HotStuff (IT-HS) [25,6] adapts HotStuff to the signature-free setting. The simplified blog construction [25] achieves 4δ good-case latency and $O(n^2)$ message complexity, using a boosting mechanism to guarantee totality of *accept* messages during view change. However, the construction is not responsive, as the leader must wait 2Δ to collect suggested locks from all honest parties. The paper version [6] eliminates this fixed waiting period by introducing a three-key quorum mechanism, thereby restoring responsiveness. This comes at the cost of two additional buffer rounds on the critical path, increasing the good-case latency to 6δ.

TetraBFT [27] achieves responsive consensus with 5δ good-case latency, $9\Delta + \delta$ worst-case view latency, and $O(n^2)$ message complexity. Instead of relying on lock-based safety, TetraBFT carries prior voting information in view-change messages, which prevents honest parties from echoing unsafe values and removes one communication round compared with IT-HS.

Forget-IT [1] achieves the optimal 3δ good-case latency under $n = 3f + 1$ with $O(n^2)$ message complexity. However, its three-graded-echo design incurs $4\Delta + 3\delta$ worst-case view latency, making view changes costly under consecutive faulty leaders or asynchronous executions. Forget-IT also considers a different model that permits message loss and crash recovery with loss of volatile state, relying on repeated broadcasts and recovery before GST. Under this model, it achieves an $O(1)$-word bound on persistent state per process.

IT-Kuplex [2] is an improvement over Forget-IT on view length. IT-Kuplex reduces the view length, from $4\Delta + 3\delta$ to $3\Delta + 2\delta$ under $n = 3f + 1$ and $3\Delta + \delta$ under $4f + 1$ by assuming synchronized clocks across honest parties. Under $n = 4f + 1$, due to the fact that quorum size $2f + 1$ is a majority of $4f + 1$, no two conflicting messages can reach that quorum of the same view, this can reduce one round of amplification of messages.

Simple-IT [28] leverages the optimistic reliable broadcast (RBC) for leaders' proposal, while a primitive called Reliable Notification (RN) for skipping rounds like in Simplex [11]. By leveraging Opt-RBC [24], Simple-IT can achieve good-case latency of 5δ, 3δ optimistically; by leveraging Bracha-RBC [7], Simple-IT can achieve good-case latency of 4δ.

Concurrent work. Floor-IT [3] achieves 2δ good-case commit latency at resilience $n = 5f - 1$, for $f \geq 2$. Its communication and recovery model differs from ours: Floor-IT permits message loss and uses repeated broadcasts whose delivery guarantees depend on continued retransmission. It also explicitly supports crashes that erase volatile state, assuming correct processes recover before GST. In this model, Floor-IT proves an $O(1)$-word bound on persistent protocol state per process. Quintus assumes eventual delivery of honest messages and does not separately model crash recovery or establish a constant persistent-storage bound. These distinctions are relevant when comparing storage and communication guarantees.

Although both Floor-IT and Quintus-Responsive achieve 2δ good-case commit latency, they differ in the responsiveness of view changes. Floor-IT delays issuing *adopt* messages for the current view until its timer reaches 2Δ. Consequently, the earliest correct entries into consecutive views first entered after GST are separated by at least 2Δ, even with honest leaders. Its vote-based catch-up rule allows an individual process to advance without waiting for its own timer, but uses votes for view V to enter that same view V. It cannot produce the first correct entry into a new view, which must instead use an adopt-based rule. Floor-IT's responsiveness therefore concerns commit latency, while its view progression remains constrained by Δ.

Two-round finality. Several protocols achieve two-round finality, either under optimistic conditions or hybrid fault models that distinguish crash from Byzantine

Each party stores: (i) $\mathcal{S}$: a set of `status` messages received for the previous view, and (ii) `key`: a local variable, initially set to $\perp$.

1. **Propose.** $\triangleright\ 3V\Delta$
   If $P_i = l_V$:
   - If $V = 0$, send $(\texttt{propose}, v, V)$ for any value v.
   - If $|\mathcal{S}| \geq n - f$:
     - If there exists $2f + 1$ `status` supporting the same non-$\perp$ value v, choose any value v that does.
     - Otherwise, can choose any value v.

     Send $(\texttt{propose}, v, V)$
   - Otherwise, no proposal in this view.
2. **Vote.** $\triangleright\ 3V\Delta + \delta$
   Upon receiving the first proposal $(\texttt{propose}, v, V)$ from the leader l_V before $(3V + 1)\Delta$:
   If $V = 0$, send $(\texttt{vote}, v, V)$ and set $\texttt{key} = v$.
   Otherwise, if $|\mathcal{S}| \geq n - f$:
   - If there exists $2f + 1$ `status` in $\mathcal{S}$ on some value $x \in \{v, \perp\}$, then send $(\texttt{vote}, v, V)$, and set $\texttt{key} = v$.
3. **Notify.** $\triangleright\ (3V + 1)\Delta$
   Broadcast $(\texttt{notify}, \texttt{key}, V)$.
4. **Commit.** $\triangleright\ 3V\Delta + 2\delta$
   Upon receiving $\geq n - f$ $(\texttt{vote}, v, V)$ messages from distinct parties, commit v.
5. **Status.** $\triangleright\ (3V + 2)\Delta$
   Upon receiving $\geq n - f$ `notify` messages from distinct parties,
   - If there exists $2f + 1$ `notify` messages for the same value x, send $(\texttt{status}, x, V)$ to all parties;
   - If there does not exist any set of $2f + 1$ `notify` messages on the same value, send $(\texttt{status}, \perp, V)$ to all parties.

   Status Amplification: At any time, on receiving $f+1$ $(\texttt{status}, v, V)$, broadcast $(\texttt{status}, v, V)$ if haven't done so.

Fig. 1. Quintus-Fixed: An $n = 5f+1$ fixed-view protocol for party P_i in view V.

faults [23,20,16,18,13], or under $n = 5f + 1$ [12]. All of them assume a computationally bounded adversary and rely on transferable certificates. We also note the DGV protocol [13], which operates in the $n = 5f - 1$ setting and achieves 2δ good-case latency. However, it is signature-free only for "in-view" messages and relies on digital signatures during view changes.

4 Quintus-Fixed: A Fixed View $5f + 1$-Protocol

Protocol Description. Figure 1 presents a signature-free partially synchronous Byzantine broadcast protocol for $n = 5f + 1$ parties. The protocol relies only on authenticated channels and does not require threshold signatures, quorum

certificates, or any cryptographic aggregation mechanism. The protocol proceeds in views where each view requires three rounds and thus, runs in 3Δ time, assuming parties have synchronized clocks.

Each party maintains a set $\mathcal{S}$ of `status` messages received for the previous view, reset at the start of each view, and a variable `key`, initially $\perp$, holding the value it currently supports, namely the value it most recently voted for.

The proposal phase of view V starts at time $3V\Delta$ (line 1). The designated leader l_V checks if it has received at least $n - f$ `status` messages from the previous view $V - 1$ for $V \geq 1$. If it has not, then the leader does not propose a value. If it has, then among these `status` messages, if a non-$\perp$ value appears at least $2f + 1$ times, the leader proposes such a value; otherwise, it may propose any value.

In the vote phase (line 2), before $(3V+1)\Delta$, upon receiving the first proposal, parties first process the `status` messages it received from the previous view to decide whether to vote for the proposal. If the party receives $2f + 1$ status messages among $\geq n - f$ received `status` messages matching the value leader sent or for $\perp$, the party can vote for the proposal. After voting, the party updates its key accordingly and broadcasts a `notify` message on the `key` at $(3V + 1)\Delta$ (line 3).

In the commit phase (line 4), a value is committed once a party receives $n - f = 4f + 1$ votes for that value in the current view.

In the status phase (line 5) if each party received $n - f$ `notify` messages for the current view by 2Δ into the view, it will broadcast the `status` message for view change. Once a party sees $2f + 1$ `notify` messages on the same value, it sends a `status` on that value; otherwise, it sends a status on $\perp$. At any time, upon receiving $f + 1$ $(\texttt{status}, v, V)$ messages, a party amplifies the message.

Key Intuition. Observe that safety holds within a view since any two sets of $n - f$ votes intersect in more than one value. Safety holds across views since, once a value v has been committed in a view, there exists a set of $n-2f = 3f+1$ honest parties who have their `key` set to v. Subsequently, no key $v' \neq v$ can be notified by $> 2f$ parties, and thus, no honest party will send a `status` for $v' \neq v$. Due to this, no honest party observes $2f + 1$ `status` on $v' \neq v$ in subsequent views and will not vote on an equivocating value.

Liveness requires all honest parties to vote on an honest leader's proposal after GST. In particular, if the leader of view V is honest, then we show that this holds so far as GST $\leq (3(V - 1) + 1)\Delta$. Observe that, since the network is synchronous, all honest parties `notify` and `status` messages will be received by all honest parties. Among the $\geq n - f$ `notify` messages received by an honest party, at least one of the two `status` sending conditions must hold. Consequently, all honest parties will send at least one `status` message, allowing an honest leader to propose.

Observe that, since a leader proposes based on $2f+1$ matching `status` messages, at least $f+1$ of these are from honest parties, and the status amplification step ensures that all honest parties amplify this status. This ensures that all hon-

est parties receive sufficiently many status messages on the value proposed by the leader. Thus, they will vote for the leader's proposal and commit the value.

Latency and message complexity. The protocol incurs a good-case latency [4] of 2-rounds (2δ), since a value can be committed as soon as it is proposed and voted on when the leader is honest and GST = 0.

The total message complexity per view is $O(n^2)$. The proposal step requires only $O(n)$ messages sent from the leader. Sending the `vote`, `notify`, and `status` message incurs $O(n^2)$ communication, because at most n parties multicast constant number of such messages to at most n parties.

4.1 Security Proofs

Lemma 1. *If some honest party commits value v in view V, (i) no other value can be committed in view V by any other honest party, (ii) there exists a set $\mathcal{Q}$ of at least $3f+1$ honest parties that have their key set to v by time $(3V+1)\Delta$.*

Proof. For part (i), observe that for any two values v and v' for $v \neq v'$ to be committed, each of them must be voted for by at least $n-f$ parties. Any two such quorums intersect in at least $3f+1$ honest parties, and none of these parties will vote for two different values in the same view.

For part (ii), observe that committing a value v requires $4f+1$ votes, out of which $3f+1$ are from honest parties. By the voting rule, the vote is only allowed before $(3V+1)\Delta$, so every party in $\mathcal{Q}$ should have set its `key` to v. □

Lemma 2. *If some honest party commits value v in view V, (i) a value $v' \neq v$ proposed in any view $> V$ will not be voted upon by any honest party, and (ii) no honest party will flip their key from v to v'.*

Proof. We prove this lemma by induction.
Base case. If some honest party commits value v in view V, then $\geq 3f+1$ honest parties (set $\mathcal{Q}$) have set their `key` to v at time $(3V+1)\Delta$. This holds by Lemma 1.

Due to this, observe that only up to $2f$ parties (f Byzantine, and f honest not in $\mathcal{Q}$) can send `notify` messages on values $v' \neq v$.

Now, observe that any honest party that sends `status` message must have collected $\geq n-f$ `notify` messages; this set intersects with set $\mathcal{Q}$ at $2f+1$ parties. Thus, no honest party will send $(\texttt{status}, v', V)$ for $v' \neq v$

At $3(V+1)\Delta$, if any party collects `status` messages from view V with $|\mathcal{S}| \geq n-f$, $n-2f = 3f+1$ of these `status` messages will be from honest parties, and each of those messages can only support v. This proves the base case for part (i), in view $V+1$, a value $v' \neq v$ proposed in the view $V+1$ will not be voted upon by any honest party.

Part (ii) follows immediately, because according to the voting rule, `key` can be changed only when a party votes. Therefore, in view $V+1$, no honest party will flip their key from v to v'.
Induction hypothesis. The lemma statement holds for some $V' > V$.
Induction step. Follows from identical reasoning as in the Base case. □

Theorem 3 (Safety). *If two honest parties P_i and P_j commit v and v' respectively, then $v = v'$.*

Proof. Suppose P_i and P_j commit v and v' respectively in views V and V'. WLOG, suppose $V' \geq V$.

If $V = V'$, then by Lemma 1, $v = v'$, a contradiction.

If $V' > V$, then by Lemma 2, any value $v'' \neq v$ will not be voted for by any honest party in view V'. Thus, $v' \neq v$ will not receive more than f votes, and thus it cannot be committed. Thus, $v = v'$. A contradiction. □

Theorem 4 (Liveness). *All honest parties will commit a value in the first view V such that $GST \leq (3(V-1)+1)\Delta$ and l_V is honest.*

Proof. At the time $t = (3(V-1)+1)\Delta$, every honest party will send an `notify` message on their `key`. By the fact the time t has passed GST, at time $(3(V-1)+2)\Delta$, all honest parties will receive `notify` messages from all honest parties (total size $\geq n-f$). We refer to the message set of honest `notify` as $\mathcal{M}_h$. In $\mathcal{M}_h$, at least one of the two conditions are satisfied to broadcast a `status` message according to the status sending rule.

Now we claim: there exists at least one value (including $\perp$), such that l_V and all parties obtain at least $2f + 1$ `status` messages on that value.

We can consider the following cases based on the value distribution in $\mathcal{M}_h$ in view $V - 1$:

- If all values appeared $< 2f + 1$ times in $\mathcal{M}_h$, all honest parties will send `status` on $\perp$.
- If there exist a value v appeared $\geq 2f+1$ times in $\mathcal{M}_h$, every honest parties will send `status` on v; and any honest party will not send `status` on $\perp$.

Given $n - f > 2f + 1$, the claim is proven.[4]

Consider the state of the honest leader l_V and the state of all honest parties after the status amplification step, we can discuss the two cases separately:

- First, suppose the leader proposed v by receives $2f + 1$ `status` messages for some non-$\perp$ value v. At least $f + 1$ of these messages were sent by honest parties and are therefore received by every honest party by $3V\Delta$. Consequently, the amplification rule ensures that every honest party also receives $2f+1$ `status` messages for v. The leader proposes v, and the voting condition for v is satisfied at every honest party.
- Otherwise, the leader propose an arbitary value based on $2f+1$ $\perp$. Following the above argument, all honest parties should receive $2f + 1$ `status` on $\perp$ after one round of message by amplification.

In either case, all honest parties vote for the honest leader's proposal in the view V. After these votes are delivered, every honest party receives a commit quorum and commits the proposed value by the end of view V. □

[4] We note that, due to this reasoning, we could require the leader to propose based on requiring $n - f$ `status` messages on the same value too; this would obviate the need for status amplification step.

Adapting the protocol to satisfy partially synchronous Byzantine Broadcast (BB). The above protocol can be easily adapted to partial synchrony BB by letting the first leader be the designated sender, and adding a termination condition that requires every party to report when they commit, and terminating upon $f + 1$ such commits. Agreement follows from the safety argument, termination follows from liveness, and validity trivially follows.

5 Quintus-Responsive: A Responsive Protocol for $n = 5f + 1$

We present an information-theoretic responsive partially synchronous Byzantine broadcast protocol in the $n = 5f + 1$ setting in Figure 2. At a high level, the protocol follows a similar reasoning sketch as in the previous section but achieves optimistic responsiveness, and a worst-case view latency of $2\Delta + 2\delta$.

5.1 Protocol description

The protocol proceeds in numbered views, each associated with a designated leader. Every view consists of a normal-case agreement phase followed by a view change phase. Each party maintains two local variables: the current view `curview`, initially 0, and the `key`, initially $\perp$. The `key` stores the value most recently voted for in any view and is used during the view change phase.

Normal-case agreement. To initiate view V, the leader waits until receiving $2f + 1$ identical `attest` messages or $3f + 1$ identical `vote` messages from view $V-1$. These messages summarize the state until the previous view. If it identifies a value $x \neq \perp$, the leader proposes $v = x$; otherwise, it may propose any value v. The leader then broadcasts a $(\texttt{propose}, v, V)$ message (line 1). The leader also starts a timer of 2Δ when it enters the view.

Upon receiving the proposal, a party votes only if it is still in view V (timer for the view V was set and does not expire), the proposal is compatible with the set of view $(V - 1)$-`attest` messages it received, i.e., either $2f + 1$ `attest` messages on the proposed value or on $\perp$, and if the party has not voted before in this view. The party records v as its current key and broadcasts a $(\texttt{vote}, v, V)$ message (line 3). A party also votes if it observes $f + 1$ $(\texttt{vote}, v, V)$ messages, if the party has not sent a `vote` or `attest` or `timeout` on an equivocating value in view V or higher (line 4).

A party commits a value once it collects $n-f$ matching vote messages (line 5).

View change. Parties enter a view upon receiving $3f + 1$ `attest` messages or $n - f$ `vote` messages for the same value x (line 2). At this point, the party sets a timer of 2Δ for the view.

When the timer for view V expires, if a party did not vote already, then it broadcasts a $(\texttt{timeout}, \texttt{key}, V)$ message.

A party sends an `attest` message in view V under two conditions. First, if it receives $2f + 1$ messages on x that are either `vote` or `timeout` from view V,

$\texttt{curview} = 0$
$\texttt{key} = \perp$

1. **View advancement for $P_i = l_V$.** Upon receiving $2f + 1$ $(\texttt{attest}, x, V - 1)$ or $3f + 1$ $(\texttt{vote}, x, V - 1)$ messages while $\texttt{curview} \leq V - 1$:
   Set $\texttt{curview} = V$
   Start a timer for 2Δ
   If $x \neq \perp$, $v = x$, else set $v =$ any value
   Send $(\texttt{propose}, v, V)$ to all parties
2. **View advancement for $P_i \neq l_V$.** Upon receiving $3f + 1$ $(\texttt{attest}, x, V - 1)$ or $4f + 1$ $(\texttt{vote}, x, V - 1)$ messages while $\texttt{curview} \leq V - 1$:
   Set $\texttt{curview} = V$,
   Start a timer for 2Δ
3. **Direct vote.** Upon receiving $(\texttt{propose}, v, V)$ from leader l_V and $2f + 1$ $(\texttt{attest}, x, V - 1)$ messages for some $x \in \{v, \perp\}$, if $\texttt{curview} = V$, $\texttt{timer} > 0$, and the party has not yet voted in view V:
   Set $\texttt{key} = v$ and send $(\texttt{vote}, v, V)$
4. **Vote amplify.** Upon receiving $f + 1$ $(\texttt{vote}, v, V)$ messages, if the party has not sent any $(\texttt{attest}, v', V')$ or $(\texttt{timeout}, v', V')$ or $(\texttt{vote}, v', V')$ with $v' \neq v$ and $V' \geq V$, then set $\texttt{key} = v$ and send $(\texttt{vote}, v, V)$, provided no $(\texttt{vote}, v, V)$ has been sent yet.
5. **Commit.** Upon receiving $n - f$ $(\texttt{vote}, v, V)$ messages, commits the value v for view V.
6. **Timeout.** If timer $= 0$ and $\texttt{curview} = V$, send $(\texttt{timeout}, \texttt{key}, V)$ if the party has not voted in view V.
7. **Attest.** Party P_i sends an $\texttt{attest}$ message for view V under any of the following conditions:
   - Upon receiving $n - f + k$ messages (for $k \geq 0$) each of which is either $(\texttt{vote}, *, V)$ or $(\texttt{timeout}, *, V)$ and provided that no value appears in $2f + 1 + k$ of those messages, then send $(\texttt{attest}, \perp, V)$.
   - Upon receiving $2f + 1$ messages each of which is either $(\texttt{vote}, x, V)$ or $(\texttt{timeout}, x, V)$, send $(\texttt{attest}, x, V)$ if $(\texttt{attest}, x, V)$ message has not been sent yet.
8. **Attest amplify.** Upon receiving $f + 1$ $(\texttt{attest}, x, V)$, send $(\texttt{attest}, x, V)$ if $(\texttt{attest}, x, V)$ message has not been sent yet.

Fig. 2. Quintus-Responsive: An $n = 5f + 1$ responsive protocol for party P_i with good-case 2δ view-change.

then it sends an $\texttt{attest}$ message for x. Second, if it receives $n - f$ messages, out of which no value appears $2f + 1$ times. In this case, it sends an $\texttt{attest}$ message for $\perp$. Note that, it is possible that both of these conditions are satisfied and a party sends multiple $\texttt{attest}$ messages. Finally, a party amplifies an $\texttt{attest}$ message on receiving $f + 1$ $\texttt{attest}$ messages for it in a view (line 8).

5.2 Key Intuition

Safety and Liveness. Observe that safety follows based on quorum intersection arguments within and across views. If a party commits on a value v in view V, it obtains $n-f$ votes, and thus, $n-2f$ honest votes. Any two such $n-2f$ honest sets intersect in $2f+1 > 0$ honest parties who will not vote on equivocating values. Across views, once a party commits v, at least $n-2f \geq 3f+1$ honest parties will have their key set to v. Subsequently, no other value, including $\perp$ can obtain $2f+1$ `attest` messages in the next view, and inductively, in any subsequent view. Due to this, no honest party votes on any other value in subsequent views. We note that, to ensure liveness, there is an indirect voting rule that allows parties to vote based on $f+1$ votes if they have not sent a conflicting `vote`, `attest`, or `timeout` message but this requires a first honest party to send one of these messages, which will not happen once a value has been committed.

For liveness, we need to ensure that a value that is proposed by an honest leader after GST will be committed. First, once an honest party enters a view after GST, other honest parties enter the view soon enough. In particular, observe that, once a non-leader party enters a view, it enters based on receiving $3f+1$ `attest` messages or $4f+1$ `vote` messages for the same value, and thus, all honest parties will obtain $2f+1$ `attest` or $3f+1$ `vote` messages within Δ time. The leader can enter the view at this time. For other parties, the `attest` messages will be amplified within another Δ time. Similarly, if the leader enters a view based on $2f+1$ `attest` or $3f+1$ `vote` messages, by similar reasoning, due to amplification, then all honest parties enter the view within 2Δ time. Thus, if some honest party moves to a higher view, all other parties will follow soon enough.

Second, to ensure that parties do not stay stuck in the same view, we require parties to all send a `vote` or `timeout` message within the view. Among these, if the same value appears $2f+1$ times, then parties send an `attest` for it, and on obtaining $2f+1$ such `attest` messages, parties can move to the next view. If no value appears $2f+1$ times, then parties send an `attest` on $\perp$, and parties can move to the next view on collecting $2f+1$ `attest` on $\perp$ messages. Importantly, since there exist $4f+1$ honest `attest` messages, it ensures that at least one of these cases is triggered, and all parties can move to the next view after GST.

Finally, after GST, we need to ensure that parties vote on an honest leader's proposal. Since the leader enters within Δ time of any honest party, they will not time out of the view before receiving the proposal because of the 2Δ timer. Thus, if they are still in the view, the leader's view entry condition ensures the requirements for voting are met, and they will directly vote for the leader's proposal. The only remaining scenario is if an honest party has already moved to a higher view before voting. This is because they either observed $3f+1$ `attest` or $4f+1$ `vote` messages for the same value. In either case, this implies at least $f+1$ honest parties have voted, and thus, the party will vote indirectly with the caveat that it has not voted, sent `timeout` on, or sent `attest` on an equivocating value in a higher view. Note that this cannot happen because $2f+1$ `attest` on an equivocating message can never be formed in view V (leader is honest, and

we are after GST) and, thus, in any subsequent views. Thus, an honest party will not vote on an equivocating value in a higher view after GST.

Latency. Observe that the protocol has a good-case latency of 2δ. When GST $= 0$, and the leader is honest, all honest parties will vote on the leader's proposal, and $n - f$ votes will be generated within 2δ time at all honest parties.

In addition, the worst-case view latency is the time between the last honest party entering a view to all honest parties entering the next view. Suppose the last honest party enters a view at time t. Then, by time $t+2\Delta$, all honest parties will have sent a `vote` or `timeout` message. By time $t+2\Delta+\delta$, every honest party would have received a `vote` or `timeout` for every honest party. At this point, either some value will appear $2f + 1$ times or no value will appear $2f + 1$ times. Thus, every party will send the same `attest` message, which will arrive at all parties within δ time. Thus, the worst-case latency is $2\Delta + 2\delta$ time.

Finally, in good cases, i.e., when consecutive leaders are honest, and after GST, parties switch between views in 2δ time since parties can enter the next view based on a quorum of `vote` messages.

Message complexity. Observe that, for every step of the protocol, every party sends $O(1)$ messages to all other parties. In particular, since an `attest` message on $x \neq \perp$ relies on the existence of $2f + 1$ and thus, $f + 1$ honest, `vote` or `timeout` messages, an honest party can `attest` only $O(1)$ messages in a view. Thus, the message complexity of the protocol is $O(n^2)$ per view.

5.3 Security Analysis

Claim 1. *If an honest party commits value v in view V, then (i) no value $v' \neq v$ can be committed in view V, and (ii) there exists a set $\mathcal{Q}$ of at least $3f+1$ honest parties that send $(\texttt{vote}, v, V)$ and set* `key` *to v when doing so.*

Proof. For part (i), suppose, for contradiction, that two distinct values v and v', with $v \neq v'$, are committed in view V. Both cases imply that each value must receive votes from at least $n - f$ parties. Any two such quorums intersect in at least $n - 2f = 3f + 1$ parties, including at least $2f + 1$ honest parties. Since an honest party votes for at most one value in a given view, no two distinct values can both be committed in view V.

For part (ii), committing v requires receiving $n - f = 4f + 1$ $(\texttt{vote}, v, V)$ messages. Since at most f parties are Byzantine, at least $n - 2f = 3f + 1$ of these votes are sent by honest parties. Let $\mathcal{Q}$ denote this set of honest voters. By the voting rule, every party in $\mathcal{Q}$ sets its `key` to v. □

Lemma 3. *If an honest party commits value v in view V, then no honest party votes for any value $v' \neq v$ in view $W > V$.*

Proof. By Claim 1, if an honest party commits v in view V, then there exists a set $\mathcal{Q}$ of at least $3f + 1$ honest parties that send $(\texttt{vote}, v, V)$ and set their `key` to v when doing so.

We prove the lemma by contradiction. Let a conflicting value v' be any value such that $v' \neq v$, including $\perp$. Suppose one of the following events occurs:

1. an honest party votes for a value v' in some view $W > V$;
2. a party in $\mathcal{Q}$ sends $(\texttt{timeout}, v', W)$ in some view $W \geq V$;
3. an honest party sends $(\texttt{attest}, v', W)$ in some view $W \geq V$;

Let m be the first such event in causal order.

First, suppose m is the event that an honest party votes for v' in a view $W > V$. If this vote is sent by vote amplification, then the party must have received at least $f + 1$ $(\texttt{vote}, v', W)$ messages. Since at most f parties are Byzantine, this threshold contains at least one earlier honest vote for v', contradicting the choice of m. Thus, the vote must be a direct vote.

For a direct vote in view W, the party must have received a proposal $(\texttt{propose}, v', W)$ and $2f + 1$ $(\texttt{attest}, x, W - 1)$ with $x \in \{v', \perp\}$. Since $v' \neq v$, $x \neq v$. Among these $2f + 1$ `attest` messages, at least $f + 1$ are from honest parties. Hence, some honest party must have previously sent $(\texttt{attest}, x, W - 1)$, contradicting the choice of m.

Second, suppose m is the event that a party in $\mathcal{Q}$ sends $(\texttt{timeout}, v', W)$. If $W = V$, this is impossible because every party in $\mathcal{Q}$ has already voted in view V and will not have sent `timeout` in view V. If $W > V$, then the party sends $(\texttt{timeout}, \texttt{key}, W)$, so it must have $\texttt{key} = v'$ at that time. Since parties in $\mathcal{Q}$ set `key` to v when voting for v in view V, and `key` is updated only when the party votes, this requires that the party previously voted for v' in some view W' such that $W > W' > V$, again contradicting the choice of m.

It remains to consider the case where m is the event when an honest party sends $(\texttt{attest}, v', W)$ in some view $W \geq V$. There are three ways this can happen.

First, the party may send $(\texttt{attest}, \perp, W)$ after receiving $n - f$ messages, each of which is either of the form $(\texttt{vote}, *, W)$ or $(\texttt{timeout}, *, W)$, with no value appearing at least $2f + 1$ times. Consider the senders of these $n - f$ messages. Since $|\mathcal{Q}| \geq 3f + 1$, this set intersects $\mathcal{Q}$ in at least $2f + 1$ honest parties. By the choice of m, no party in $\mathcal{Q}$ has previously voted for v' in a view greater than V, and no party in $\mathcal{Q}$ has previously sent a timeout for v'. Moreover, in view V, every party in $\mathcal{Q}$ voted for v. Therefore, all messages in this intersection are for value v. Hence, v appears at least $2f + 1$ times among the received $n - f$ messages, so the condition for sending $(\texttt{attest}, \perp, W)$ is not satisfied.

Second, the party may send $(\texttt{attest}, v', W)$ after receiving $2f + 1$ messages, each of which is either $(\texttt{vote}, v', W)$ or $(\texttt{timeout}, v', W)$. Since there are only $2f$ parties outside $\mathcal{Q}$, these $2f + 1$ messages must include a message from some party in $\mathcal{Q}$. If that message is a vote for v', then either $W = V$, contradicting that parties in $\mathcal{Q}$ voted for v in view V, or $W > V$, it contradicts the choice of m. If that message is a timeout for v', it contradicts the choice of m.

Third, the party may send $(\texttt{attest}, v', W)$ by `attest` amplification after receiving $f + 1$ messages of the form $(\texttt{attest}, v', W)$. Since at most f parties are Byzantine, at least one of these `attest` messages must have been sent earlier by an honest party. This contradicts the choice of m.

All possible cases lead to a contradiction. Therefore, no honest party votes for any value $v' \neq v$ in any view $W > V$. □

Theorem 5 (Safety). *If two honest parties P_i and P_j commit v and v' respectively, then $v = v'$.*

Proof. Suppose that parties P_i and P_j commit values v and v', respectively, in views V and V'. Without loss of generality, assume that $V' \geq V$. We show that $v = v'$.

If $V' = V$, then Claim 1 implies that no two distinct values can be committed in the same view. Hence, $v = v'$.

Otherwise, suppose $V' > V$. By Lemma 3, no honest party votes for any value different from v in view V'. Therefore, any value $v' \neq v$ can receive votes only from Byzantine parties, of which there are at most f. Since committing a value requires $n - f$ votes, v' cannot be committed. Hence, $v' = v$. □

Claim 2. *If a party receives $2f + 1$ $(\texttt{attest}, x, V)$ at time t, then all honest parties receive $\geq n - f$ $(\texttt{attest}, x, V)$ messages by time $\max(t, GST) + 2\Delta$.*

Proof. Suppose a party receives $2f+1$ $(\texttt{attest}, x, V)$ messages at time t. Among these messages, at least $f+1$ must have been sent by honest parties. Since honest parties send their `attest` messages to all parties, every honest party receives at least $f + 1$ $(\texttt{attest}, x, V)$ messages by time $\max(t, \text{GST}) + \Delta$.

Upon receiving these $f+1$ `attest` messages, an honest party sends $(\texttt{attest}, x, V)$ to all parties. Consequently, all honest parties send their `attest` messages by time $\max(t, \text{GST}) + \Delta$. Since there are at least $n - f$ honest parties, every honest party receives at least $n - f$ `attest` messages by time $\max(t, \text{GST}) + 2\Delta$. □

Claim 3. *If a party receives $3f+1$ $(\texttt{vote}, x, V)$ at time t, then all honest parties receive at least $n - f$ $(\texttt{attest}, x, V)$ messages by time $\max(t, GST) + 2\Delta$.*

Proof. Suppose a party receives $3f + 1$ $(\texttt{vote}, x, V)$ messages at time t. Among these messages, at least $2f+1$ must have been sent by honest parties. Since honest parties send their `vote` messages to all parties, every honest party receives at least $2f + 1$ $(\texttt{vote}, x, V)$ messages by time $\max(t, \text{GST}) + \Delta$.

Upon receiving these $2f+1$ `vote` messages, an honest party sends $(\texttt{attest}, x, V)$ to all parties. Consequently, all honest parties send their `attest` messages by time $\max(t, \text{GST}) + \Delta$. Since there are at least $n - f$ honest parties, every honest party receives at least $n - f$ `attest` messages by time $\max(t, \text{GST}) + 2\Delta$. □

Claim 4. *If an honest party receives $3f + 1$ $(\texttt{attest}, x, V)$ messages at time t, then the leader l_{V+1} receives at least $2f + 1$ $(\texttt{attest}, x, V)$ messages by time $\max(t, GST) + \Delta$.*

Proof. Suppose an honest party received $3f + 1$ $(\texttt{attest}, x, V)$ messages at time t. Among them at least $2f + 1$ messages are from honest parties who send $(\texttt{attest}, x, V)$ to all parties. Consequently, the leader l_{V+1} receives at least $2f+1$ $(\texttt{attest}, x, V)$ messages by time $\max(t, \text{GST}) + \Delta$. □

The proof of the following claim (Claim 5) is identical to Claim 4.

Claim 5. *If an honest party receives* $4f + 1$ $(\texttt{vote}, x, V)$ *messages at time* t*, then the leader* l_{V+1} *receives at least* $3f + 1$ $(\texttt{vote}, x, V)$ *messages by time* $\max(t, GST) + \Delta$.

Due to space constraints, we present remaining security analysis in Appendix A.

References

1. Abraham, I., Das, S., Efron, Y., Komatovic, J.: Forget-IT: Optimal good-case latency for information-theoretic BFT. Cryptology ePrint Archive, Paper 2026/355 (2026), https://eprint.iacr.org/2026/355
2. Abraham, I., Das, S., Efron, Y., Komatovic, J., Stern, G.: Information-Theoretic Kuplex (2026), https://decentralizedthoughts.github.io/2026-06-05-IT-Kuplex/
3. Abraham, I., Efron, Y., Komatovic, J., Ranchal-Pedrosa, A.: Floor-IT: Information-theoretic BFT in partial synchrony with two round good case latency and optimal resilience. Cryptology ePrint Archive, Paper 2026/1496 (2026), https://eprint.iacr.org/2026/1496
4. Abraham, I., Nayak, K., Ren, L., Xiang, Z.: Good-case latency of byzantine broadcast: A complete categorization. In: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing. pp. 331–341 (2021)
5. Abraham, I., Nayak, K., Shrestha, N.: Optimal good-case latency for rotating leader synchronous bft. In: 25th International Conference on Principles of Distributed Systems (OPODIS 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2022)
6. Abraham, I., Stern, G.: Information theoretic hotstuff. In: Bramas, Q., Oshman, R., Romano, P. (eds.) 24th International Conference on Principles of Distributed Systems, OPODIS 2020, Strasbourg, France (Virtual Conference), December 14-16, 2020. LIPIcs, vol. 184, pp. 11:1–11:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPICS.OPODIS.2020.11
7. Bracha, G.: Asynchronous Byzantine Agreement Protocols. Inf. Comput. **75**(2), 130–143 (1987). https://doi.org/10.1016/0890-5401(87)90054-X
8. Buchman, E., Kwon, J., Milosevic, Z.: The latest gossip on BFT consensus. CoRR **abs/1807.04938** (2018), http://arxiv.org/abs/1807.04938
9. Buterin, V., Hernandez, D., Kamphefner, T., Pham, K., Qiao, Z., Ryan, D., Sin, J., Wang, Y., Zhang, Y.X.: Combining GHOST and Casper. CoRR **abs/2003.03052** (2020), https://arxiv.org/abs/2003.03052
10. de Castro, M.O.T.: Practical Byzantine fault tolerance. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (2000), https://hdl.handle.net/1721.1/86581
11. Chan, B.Y., Pass, R.: Simplex consensus: A simple and fast consensus protocol. Cryptology ePrint Archive, Paper 2023/463 (2023), https://eprint.iacr.org/2023/463
12. Chou, B.K., Lewis-Pye, A., O'Grady, P.: Minimmit: Fast finality with even faster blocks (2026), https://arxiv.org/abs/2508.10862
13. Dutta, P., Guerraoui, R., Vukolic, M.: Best-case complexity of asynchronous Byzantine consensus. Tech. rep., Technical Report EPFL/IC/200499, EPFL (2005)

14. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of the ACM (JACM) **35**(2), 288–323 (1988)
15. Giridharan, N., Suri-Payer, F., Ding, M., Howard, H., Abraham, I., Crooks, N.: Beegees: Stayin'alive in chained bft. In: Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing. pp. 233–243 (2023)
16. Gueta, G.G., Abraham, I., Grossman, S., Malkhi, D., Pinkas, B., Reiter, M., Seredinschi, D.A., Tamir, O., Tomescu, A.: SBFT: A scalable and decentralized trust infrastructure. In: 2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN). pp. 568–580. IEEE (2019)
17. Jalalzai, M.M., Babel, K., Komatovic, J., Klenze, T., Das, S., Elsheimy, F., Setrin, M., Bergschneider, J., Gilkalaye, B.: MonadBFT: Fast, Responsive, Fork-Resistant Streamlined Consensus (2026), https://arxiv.org/abs/2502.20692
18. Kniep, Q., Sliwinski, J., Wattenhofer, R.: Solana alpenglow consensus, https://drive.google.com/file/d/1Rlr3PdHsBmPahOInP6-Pl0bMzdayltdV
19. Malkhi, D., Nayak, K.: Extended abstract: HotStuff-2: Optimal two-phase responsive BFT. Cryptology ePrint Archive, Paper 2023/397 (2023), https://eprint.iacr.org/2023/397
20. Martin, J.P., Alvisi, L.: Fast byzantine consensus. IEEE Transactions on Dependable and Secure Computing **3**(3), 202–215 (2006)
21. Pass, R., Shi, E.: Hybrid consensus: Efficient consensus in the permissionless model. Cryptology ePrint Archive, Paper 2016/917 (2016), https://eprint.iacr.org/2016/917
22. Shoup, V.: Sing a Song of Simplex. In: Alistarh, D. (ed.) 38th International Symposium on Distributed Computing (DISC 2024). Leibniz International Proceedings in Informatics (LIPIcs), vol. 319, pp. 37:1–37:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2024). https://doi.org/10.4230/LIPIcs.DISC.2024.37
23. Shrestha, N., Kate, A., Nayak, K.: Hydrangea: Optimistic two-round partial synchrony with improved fault resilience. In: 35th USENIX Security Symposium (2026), https://www.usenix.org/conference/usenixsecurity26/presentation/shrestha
24. Shrestha, N., Yu, Q., Kate, A., Losa, G., Nayak, K., Wang, X.: Optimistic, signature-free reliable broadcast and its applications. In: Huang, C., Chen, J., Shieh, S., Lie, D., Cortier, V. (eds.) Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025. pp. 3780–3794. ACM (2025). https://doi.org/10.1145/3719027.3765220
25. Stern, G., Abraham, I.: Information Theoretic HotStuff (IT-HS): Part One (2026), https://decentralizedthoughts.github.io/2021-09-20-information-theoretic-hotstuff-it-hs-part-one/
26. Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: HotStuff: Bft consensus with linearity and responsiveness. In: Proceedings of the 2019 ACM symposium on principles of distributed computing. pp. 347–356 (2019)
27. Yu, Q., Losa, G., Wang, X.: TetraBFT: Reducing latency of unauthenticated, responsive BFT consensus. In: Gelles, R., Olivetti, D., Kuznetsov, P. (eds.) Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024. pp. 257–267. ACM (2024). https://doi.org/10.1145/3662158.3662783
28. Yu, Q., Villacis, J., Losa, G., Xiang, Z., Wang, X.: Simple-IT: Practical low-latency signature-free bft consensus (2026), https://arxiv.org/abs/2606.14404

A Remaining Security Analysis of Quintus-Responsive

Lemma 4. *If the first honest party to enter view V does so at time t, then the leader l_V enters view V or higher by time* $\max(t, GST) + \Delta$ *and other honest parties enter view V or higher by time* $\max(t, GST) + 2\Delta$.

Proof. Suppose an honest party P_i enters view V at time t. If $P_i = l_V$, then P_i must have received either $2f + 1$ `attest` messages for the same value, or $3f + 1$ `vote` messages for the same value, in view $V - 1$ by time t.

In the former case, by Claim 2, every honest party receives $n - f$ `attest` messages for that value in view $V - 1$ by time $\max(t, \text{GST}) + 2\Delta$. In the latter case, by Claim 3, every honest party likewise receives $n - f$ `attest` messages for that value in view $V - 1$ by time $\max(t, \text{GST}) + 2\Delta$.

On the other hand, suppose $P_i \neq l_V$. Then, by time t, party P_i must have received either $3f+1$ `attest` messages for the same value or $4f+1$ `vote` messages for the same value in view $V - 1$.

In the former case, by Claim 2, every honest party receives $n - f$ `attest` messages for that value in view $V - 1$ by time $\max(t, \text{GST}) + 2\Delta$. Moreover, by Claim 4, the leader l_V receives at least $2f + 1$ `attest` messages for the same value in view $V - 1$ by time $\max(t, \text{GST}) + \Delta$.

In the latter case, by Claim 3, every honest party receives $n - f$ `attest` messages for that value by time $\max(t, \text{GST}) + 2\Delta$. Similarly, by Claim 5, the leader l_V receives at least $3f+1$ votes for the same value by time $\max(t, \text{GST}) + \Delta$. Therefore, the claim follows. □

Lemma 5. *All honest parties keep entering increasing views.*

Proof. Suppose, for contradiction, that some honest party, P_i, becomes stuck in view V. Let V' be the highest view reached by any honest party at any time. If $V' > V$, then by Lemma 4, party P_i must eventually enter view V' or a higher view, contradicting the assumption that P_i remains stuck in view V.

It remains to consider the case $V' = V$. In this case, no honest party ever enters a view higher than V. Since Lemma 4 guarantees that all honest parties eventually enter view V, it follows that all honest parties eventually become stuck in view V.

Consider any honest party in view V. If it votes in view V, then it multicasts a $(\texttt{vote}, x, V)$ message for some value x. Otherwise, its timer for view V eventually expires; by the timeout rule, it multicasts $(\texttt{timeout}, \texttt{key}, V)$. Thus, every honest party eventually multicasts at least one message for view V of the form $(\texttt{vote}, x, V)$ or $(\texttt{timeout}, x, V)$ for some value x.

Fix a set H of exactly $n - f$ honest parties, which exists because there are at most f Byzantine parties. Let M contain one such message from each member of H. All quorum counts are over distinct senders. By eventual delivery, every honest party eventually receives all messages in the same set M.

For a message collection S, let $c_S(x)$ denote the number of messages in S carrying value x. We distinguish two cases.

Case 1: Some value x satisfies $c_M(x) \geq 2f+1$. Every honest party eventually receives these $2f + 1$ messages and therefore multicasts $(\texttt{attest}, x, V)$, unless it has already done so.

Case 2: Every value x satisfies $c_M(x) \leq 2f$. Fix an honest party P_i that has received all of M. Let $R_i \supseteq M$ be the `vote`/`timeout` messages for V that P_i has received, and let $k_i = |R_i \setminus M|$, so $|R_i| = n-f+k_i$. Each message outside M adds at most one to any value's count. For every value x, $c_{R_i}(x) \leq c_M(x)+|R_i \setminus M| \leq 2f + k_i < 2f + 1 + k_i$. Thus, the fallback attest rule applies to R_i with $k = k_i$, and P_i multicasts $(\texttt{attest}, \perp, V)$. This argument applies to every honest party, even though their additional messages and values of k_i may differ.

Thus, in either case, there exists a value x such that every honest party eventually receives $n - f$ messages of the form $(\texttt{attest}, x, V)$. This is sufficient for every honest party in view V to enter view $V + 1$. □

Claim 6. *If an honest party enters view V then at least one honest party must have already entered $V - 1$.*

Proof. An honest party enters view V only after receiving at least $2f+1$ `attest` messages or at least $3f + 1$ `vote` messages for view $V - 1$. In either case, the corresponding set contains messages from at least $f + 1$ honest parties.

We first consider the case where an honest party sends a `vote` message for view $V-1$. If the vote is sent by the direct-voting rule, then the party must have already entered view $V-1$. Otherwise, the vote is sent by the vote-amplification rule, in which case the party must have received $f + 1$ `vote` messages for the same value in view $V - 1$. Since at most f parties are Byzantine, at least one of these `vote` messages was sent earlier by an honest party which must have sent via the direct vote rule. Therefore, at least one honest party must enter $V - 1$.

Next, consider the case where an honest party sends an `attest` message for view $V - 1$. An honest party sends an `attest` message in one of the following ways. First, it may receive $2f + 1$ messages for the same value, each of which is either a `vote` or a `timeout` message for view $V - 1$. Second, it may receive $n-f+k$ messages, each of which is either a `vote` or a `timeout` message for view $V - 1$, with no value appearing at least $2f + k + 1$ times. Third, it may amplify after receiving $f + 1$ `attest` messages for view $V - 1$.

In the first two cases, at least one honest party must have sent `vote` or `timeout` message for view $V-1$. If it is a `vote` message, by argument above, some honest party must already have entered view $V - 1$. If it is a `timeout` message, then the sender of that `timeout` message must have had $\texttt{curview} = V - 1$ when its timer expired; hence, that honest party must already have entered view $V-1$.

In the third case, where the `attest` message is sent by amplification, the party received $f+1$ `attest` messages for view $V-1$. Since at most f parties are Byzantine, at least one of these `attest` messages was sent earlier by an honest party. Therefore, it must have been sent by one of the first two `attest`-generation rules which implies that some honest party had already entered view $V - 1$.

Therefore, the claim follows. □

Lemma 6. *If l_V is honest and proposes value v in view V at time $t \geq GST$, then no honest party sends $(\texttt{attest}, v', W)$ or $(\texttt{vote}, v', W)$ or $(\texttt{timeout}, v', W)$ for any value $v' \neq v$, including $\perp$, in any view $W \geq V$.*

Proof. We prove this by contradiction. Let $W \geq V$ be the first view in which some honest party sends a conflicting message, either $(\texttt{vote}, v', W)$ or $(\texttt{attest}, v', W)$ or $(\texttt{timeout}, v', W)$ for some value $v' \neq v$. Let m be the first conflicting message by an honest party in the causal order.

We consider all possible ways in which such a message could be sent.

Consider a message $m = (\texttt{vote}, v', W)$. If $W = V$, then this is impossible through the direct voting rule because the leader l_V is honest and proposes only v. Thus, an honest party can directly vote only for v in view V. Moreover, vote amplification cannot trigger an honest party to send $(\texttt{vote}, v', V)$: to send $(\texttt{vote}, v', V)$ by amplification, an honest party must receive $f + 1$ $(\texttt{vote}, v', V)$ messages. Since at most f parties are Byzantine, at least one of these votes must have been sent by an honest party by direct voting rule, contradicting the choice of m.

Now suppose $W > V$. If an honest party directly votes for v' in view W, then it must have received a proposal $(\texttt{propose}, v', W)$ and $2f+1$ $(\texttt{attest}, x, W-1)$ messages for some $x \in \{v', \perp\}$. Since $v' \neq v$, we have $x \neq v$. Among these $2f + 1$ `attest` messages, at least $f + 1$ are sent by honest parties. Hence, some honest party must have previously sent $(\texttt{attest}, x, W - 1)$ with $x \neq v$, contradicting the minimality of W as the first view in which an honest party sends a conflicting message. Similarly, vote amplification cannot produce $(\texttt{vote}, v', W)$, because receiving $f+1$ votes for v' requires at least one prior honest vote for v', contradicting the choice of m.

Second, suppose $m = (\texttt{timeout}, v', W)$. By the timeout rule, the sender has not voted in view W and sends its current `key`. Thus, at the time of sending, the party has $\texttt{key} = v'$.

For $W = V$, this is impossible since l_V is honest and every honest party receives the honest leader's proposal and corresponding view $V-1$ `attest` message before its timer expires and therefore votes for v before it can send a timeout in view V.

For $W > V$, observe that `key` is updated only when an honest party votes. Since the party's `key` is set to v' at the time it sends $(\texttt{timeout}, v', W)$, either it never voted for v after the honest proposal in view V, or it previously voted for v'. The second case contradicts the choice of m.

The first case is impossible. Every honest party receives the honest leader's proposal and corresponding view $V - 1$ `attest` message before its timer expires and votes for v and therefore votes for v, or it advances before voting in view V. This advancement can only be caused by votes for v which triggers vote amplification for v.

We now consider the case where $m = (\texttt{attest}, v', W)$ was sent. An honest party can send such an `attest` message in one of three ways.

First, the party may receive $n - f$ messages for view W, each of which is either a `vote` or a `timeout` message, such that no value appears at least $2f + 1$

times. Since m is the first conflicting message, every honest `vote` or `timeout` message for view W sent before m is for v. Among the $n - f$ received messages, at most f are Byzantine, so at least $3f+1$ are honest messages for v. Therefore, v appears at least $3f+1$, and in particular at least $2f+1$, times among the received messages. Hence, the condition for sending $(\texttt{attest}, \perp, W)$ is not satisfied.

Second, the party may receive $2f + 1$ messages for the same value v', each of which is either a $(\texttt{vote}, v', W)$ or a $(\texttt{timeout}, v', W)$ message. Since at most f parties are Byzantine, at least $f + 1$ of these messages are from honest parties. Therefore, some honest party must have previously sent either $(\texttt{vote}, v', W)$ or $(\texttt{timeout}, v', W)$, contradicting the choice of m.

Third, the party may send $(\texttt{attest}, v', W)$ by `attest` amplification after receiving $f+1$ messages of the form $(\texttt{attest}, v', W)$. Since at most f parties are Byzantine, at least one of these `attest` messages must have been sent earlier by an honest party, contradicting the choice of m.

Thus every possible way of producing the first conflicting honest message leads to a contradiction. Therefore, no honest party sends $(\texttt{vote}, v', W)$ or $(\texttt{attest}, v', W)$ or $(\texttt{timeout}, v', W)$ for any $v' \neq v$ and any view $W \geq V$. □

Lemma 7. *If the first honest party to enter view V does so at time $t \geq GST$ and l_V is honest, then all honest parties commit value x proposed by l_V.*

Proof. By Lemma 4, the leader l_V enters view V or a higher view by time $t+\Delta$, and every honest party enters view V or a higher view by time $t+2\Delta$. Moreover, by Claim 6, no honest party enters any view $V' > V$ before time t. Hence, before time $t + 2\Delta$, no honest party can send a `timeout` message for any view $V' \geq V$ due to view-timer expiration.

We first consider the case where some honest party enters a view $V' > V$ before time $t+2\Delta$. By Claim 6, some honest party must first enter view $V+1$. To do so, it must receive sufficient view-V evidence, namely either enough `attest` messages or enough `vote` messages for the same value in view V.

Since no honest party sends `timeout` for view V due to timeout before time $t + 2\Delta$, any such view-V evidence must ultimately be triggered by votes in view V. In particular, at least $f + 1$ honest parties must have sent `vote` messages for some value in view V. By Lemma 6, after the honest leader l_V proposes x, no honest party sends a vote or an `attest` or a `timeout` message for any conflicting value in any view $V' \geq V$. Therefore, these $f + 1$ honest votes must be for x.

Since honest parties multicast their votes, every honest party eventually receives $f + 1$ $(\texttt{vote}, x, V)$ messages. By the vote-amplification rule, every honest party that has not already voted sends $(\texttt{vote}, x, V)$. Hence, every honest party eventually receives $n - f$ $(\texttt{vote}, x, V)$ messages and commits x.

It remains to consider the case where no honest party enters a view higher than V before time $t + 2\Delta$. Then the honest leader l_V enters view V by time $t + \Delta$ and proposes value x. Since $t \geq$ GST, this proposal is delivered to every honest party by time $t + 2\Delta$. In addition, every honest party has also entered view V by that time. Thus, every honest party votes for the leader's proposal x in view V. Consequently, all honest parties eventually receive $n - f$ $(\texttt{vote}, x, V)$ messages and commit x.

Therefore, in all cases, all honest parties eventually commit the value x proposed by the honest leader l_V. □

Theorem 6 (Liveness). *All honest parties will commit a value in the first view V after GST such that l_V is honest.*

Proof. By Lemma 5, all honest parties continue to make progress by entering higher views. As the leader rotates across views, honest leaders will eventually be elected. Then, by Lemma 7, a value x proposed by an honest leader after GST will be committed by all honest parties. □