Beyond Leaderless: Randomized Consensus Against Mobile Crashes

Ittai Abraham[1], Kaya Alpturer[2], Pranav Garimidi[1], Joachim Neu[3], Tim Roughgarden[1,4], and Ertem Nusret Tas[1]

[1] a16z frontier, {iabraham,pgarimidi,ntas}@a16z.com
[2] Princeton University, kalpturer@princeton.edu
[3] Technion, joachimneu@gmail.com
[4] Institute for Advanced Study, tim.roughgarden@gmail.com

Abstract. Many consensus protocols fail to terminate under attack from an adversary that has the power to move between replicas and repeatedly crash one of them, even in synchrony. We present Multi-Leader Fixed-View Simplex (ML-FVS) to defend against such an adversary while keeping the simple structure and good-case latency of Simplex consensus. Our protocol extends fixed-view Simplex with multiple leaders, each of which may broadcast a proposal. At the voting deadline, each replica votes on the set of proposals it has seen so far. A hidden threshold beacon revealed after the votes gives a priority order on the multiple leaders, resolving each vote to its highest-priority proposal. Our model combines permanent adaptive Byzantine faults with mobile crashes. For a constant crash budget and Δ-delayed Byzantine corruption, we obtain $O(\Delta)$ expected latency over repeated views whose leaders start honest. With immediate Byzantine corruptions, the protocol still achieves liveness. We also evaluate the protocol under random network delays and stochastic faults, finding that multiple leaders reduce view failures even when rational leaders delay their proposals to increase rewards.

1 Introduction

A well-known lower bound is that in a synchronous network, consensus tolerating t crashes requires at least $t+1$ communication rounds in the worst case [6,26]. An even stronger setting is one in which a *mobile* adversary is able to observe the execution and change which replica crashes. Santoro and Widmayer [5, 29] showed that even one mobile crash can lead to an infinite execution of a deterministic consensus protocol under synchrony.[5] Randomized protocols, on the other hand, can avoid this issue by making sure non-terminating executions have probability zero. The key problem to avoid is that, if liveness depends on a single, publicly known leader, the adversary can attack that leader and crash it, moving on to its replacement in the following view. Simply rotating leaders does not resolve this problem when the adversary can follow the rotation.

[5] These classical lower bounds assume agreement and strong unanimity/validity. The first bound assumes $n > t + 1$.

Several protocols in the literature reduce reliance on a single leader, but their guarantees against mobile faults differ (see Appendix F for further comparisons):

In *Leaderless Consensus*, Antoniadis, Desjardins, Gramoli, Guerraoui, and Zablotchi [9] study resilience to *mobile suspension*[6]: in their message-passing model, the suspended replica stops listening and replying for a single round of two message delays while the adversary may select a different replica in the next round. Their BFT-Archipelago protocol tolerates $f - 1$ Byzantine replicas and one suspended replica among $n = 3f+1$ replicas, with an $O(n)$-round worst-case latency bound, too high for practical consensus.

Cadence [12] shows that the difference between mobile suspension and mobile crashes matters for protocols for multiple proposers since, unlike a suspension, a crash can partially deliver messages. One way to interpret their protocol is that it implements a virtual leader via forming a quorum over the success or failure of each individual proposer. While a suspended proposer can be excluded consistently in the views of replicas using a quorum certificate, a crash during the proposal broadcast can split positive and negative votes such that neither inclusion nor exclusion receives a quorum. Then, the view fails and Cadence must start a new agreement instance for the same slot to recover.

Mysticeti [11] orders blocks through a public sequence of leaders. Leaders earlier in this sequence must be committed or skipped before later blocks can be output. While a quorum can skip a suspended proposer, as in the attack on Cadence, a crash during the proposal broadcast can leave only some replicas with the proposal. The proposal may then receive too few votes to be committed and too few negative votes to be skipped. Later proposals can then have all their commit certificates and still wait for a decision on an earlier leader.

An alternative approach is VABA [4], which runs n parallel leader-based protocols and then selects one using shared randomness. This prevents the adversary from knowing in advance which instance will determine the outcome. VABA provides this guarantee in asynchrony, but its high latency even in the good case makes it impractical. Our aim is to use this delayed selection with fewer leaders and lower latency under partial synchrony. We thus ask:

Can we tolerate a small number of adaptive mobile crashes while keeping the simple structure and fast good case of Simplex?

Our protocol, ML-FVS, extends fixed-view Simplex [1, 16]. Each view has k publicly designated leaders, each of which may broadcast a proposal. The protocol keeps the proposal, voting, and finalization steps of Simplex and ranks the leaders at random after voting. We achieve this by making replicas vote on the set of proposals that they have seen and include shares of a $(2f + 1)$-threshold randomness beacon. When votes are combined, the beacon reveals a random priority order, and each vote resolves to its highest-priority proposal.

[6] Note that in a suspension (sometimes also called *atomic crash*), the replica does not send any message in the prescribed period while a (regular, non-atomic) crash can interrupt a broadcast and deliver a subset of messages.

We consider adaptively chosen permanent Byzantine corruptions and temporary crashes. Both count against a total fault budget f, while crashes also count against a separate crash budget c. A Byzantine corruption reveals the replica's state and secrets and once activated never refunds its budget. A crash preserves that state and reveals no secrets. Healing a crash resumes the replica immediately, but the crash counts against both budgets for another Δ. Message delays are bounded by Δ after GST. We consider both immediate Byzantine corruption, which takes effect as soon as its target is chosen (delay $D = 0$), and corruption that takes effect Δ after its target is chosen (delay $D = \Delta$). Mobile crashes are always immediate.

With Δ-delayed Byzantine corruption, we call a view starting at least Δ after GST a *good-case view* if all its leaders are honest at the view's start and no earlier Byzantine attacks are pending against them. In this good case, mobile crashes are still allowed, and the adversary may start new Byzantine attacks during the view.

At the start of a view, the adversary can interrupt proposal broadcasts without knowing which leader the beacon will rank first. In the good case, the view succeeds if that leader completes its broadcast even if the others do not. Note that crashing the highest-priority leader later cannot erase already sent messages. For instance, in a good-case view with two leaders and one mobile crash, at least one leader completes its proposal broadcast. Since the beacon ranks such a leader first with probability at least $1/2$, each such view produces a final certificate within 3Δ with probability at least $1/2$. Repeated good-case views give expected latency 6Δ.

Theorem 1 (Main result, informal). *With $n = 3f + 1$ replicas, $k = c + 1$ leaders suffice against c mobile crashes: each good-case view finalizes a block within 3Δ with probability at least $1/(c+1)$, even with $f - c$ Byzantine replicas. Repeated good-case views give expected latency at most $3\Delta(c + 1)$.*

In particular, two leaders tolerate one mobile crash, requiring at most two views on average.

Moreover, with randomly sampled leaders, the protocol still achieves liveness even when some leaders are Byzantine, under both immediate and delayed Byzantine corruption (Section 4).

1.1 Main Contributions

- **Safety.** We prove that honest replicas running ML-FVS never finalize conflicting blocks under mobile crashes and adaptive Byzantine corruption using quorum intersection (Section 3).
- **Liveness under adaptive faults.** With uniformly randomly sampled $c+1$ leaders, final certificates are produced in infinitely many views under both immediate and delayed Byzantine corruption (Section 4). In addition to the good-case result above, with Δ-delayed Byzantine corruption, we also prove constant expected latency for constant c mobile crashes and $c + 1$ leaders if leaders are announced at the start of the view.

- **View failures under stochastic faults.** We evaluate how multiple leaders reduce view failures under random network delays and stochastic leader stalls. We measure the probability that a view fails to finalize as we vary the number of leaders.
- **Rational proposal timing.** We analyze leaders that delay proposals to collect more valuable transactions [31]. We characterize equilibrium proposal delays and evaluate whether multiple leaders still reduce view failures at equilibrium. In our evaluation, adding a second leader reduces the computed view-failure rate from about 1% to about 0.01% at equilibrium (Section 5).

2 Model

We study our protocol ML-FVS in a partially synchronous network against an adaptive adversary with permanent Byzantine faults and mobile crashes.

Setup. Let $n = 3f+1$ be the number of replicas, where f is the total fault budget defined below. Our protocol will use a quorum size of $Q = n - f = 2f + 1$. Every message is signed by its sender. We assume a public external-validity function that checks a block and its proof [13]. A block is *externally valid* if this function accepts. We assume that an honest leader with the required certificates can construct an externally valid extension of the block chain, and its proof.

Replicas have synchronized clocks and $\Delta > 0$ is known. The parameter Δ determines the fixed view schedule in Section 3. Messages delivered by a scheduled time are processed before the protocol actions, and messages generated by those actions are processed after.

Leader selection. For $1 \leq k \leq n$, each view v has a uniformly sampled set of k distinct leaders. Leader sets are sampled independently across views, using randomness independent of the threshold beacon that later determines their priority order. The selected replicas are indexed by $L_v[j]$, $j \in [k]$. The schedule is public by each view's start and may be announced in advance.

2.1 Adaptive Byzantine Corruptions and Mobile Crashes

A replica is *honest* until a permanent Byzantine corruption takes effect. After corruption, the adversary controls its behavior and may read, copy, or modify its local state, including signing keys, beacon shares, and protocol records. An *always honest* replica is one that never becomes Byzantine. An honest replica is *active* unless currently crashed. A crashed replica neither sends nor processes messages, but preserves its local state and secrets without revealing them to the adversary. Messages arriving while it is crashed are buffered and immediately delivered and processed upon healing. Initially all replicas are honest and active.

Definition 1 (Adaptive mobile adversary). *Fix a total fault budget f, a mobile-crash budget $0 \leq c \leq f$, a permanent corruption delay $D \in \{0, \Delta\}$, and*

a budget refund delay $\tau = \Delta$.[7] *The remaining budgets are initially $F = f$ and $C = c$. The adversary has the following operations.*[8]

1. ***Permanent Byzantine corruption.*** *If $F > 0$, the adversary may start a corruption of an active honest replica at time t. The target is fixed then, and F decreases immediately by one and C is unchanged. The target becomes Byzantine at $t + D$. The replica remains Byzantine forever, cannot be healed, and the budget consumed by this corruption is never refunded.*
2. ***Mobile crash.*** *If $F > 0$ and $C > 0$, the adversary may crash an active honest replica. The crash takes effect immediately and decreases both F and C by one.*
3. ***Healing.*** *The adversary may heal a currently crashed replica at time t. The replica immediately resumes honest execution with its preserved state. Neither budget is restored at healing. At $t + \tau$, both F and C increase by one.*

At most f distinct replicas ever become Byzantine, and at most c crashes are awaiting a budget refund at a time. These crashes, together with all started permanent corruptions, total at most f. The adversary observes the public execution and Byzantine replicas' states. If a crash interrupts a broadcast, it can choose which messages were sent before the interruption.

Timing. We assume that messages sent between distinct replicas take positive time. After the unknown global stabilization time GST, messages sent by active honest replicas arrive within Δ. Before GST, delays have no known bound, but messages in transit at GST arrive by GST + Δ. An honest recipient receives or buffers the message regardless of whether it is crashed or active when the message arrives. A message sent by an active replica cannot be erased by later faults. Healing resumes interrupted broadcasts to the remaining recipients, with delays measured from each send.

2.2 Threshold Priority Beacon

Each view uses a threshold beacon, revealed during voting, to order its public leaders.

Definition 2 (Threshold beacon). *Setup supplies public parameters pk, including share verification keys, and one private beacon share sk_i for each replica. For a tag T, the operations are:*

- $\mathsf{Share}(\mathsf{sk}_i, T)$ *produces replica i's share $\sigma_{i,T}$.*
- $\mathsf{VerifyShare}(\mathsf{pk}, i, T, \sigma_{i,T})$ *checks a share using the corresponding share verification key.*

[7] We treat c as constant and independent of n.

[8] A corruption or crash can target only an active honest replica with no pending permanent corruption. A healed replica may be targeted again before a prior crash refund, but the new operation uses more budget. Each healing schedules its refunds exactly once, and later operations do not change those refunds.

- $\mathsf{Combine}(\mathsf{pk}, T, S)$ *accepts* $Q = 2f + 1$ *valid shares from distinct replicas for that tag, and produces a permutation* π_T *of replica identities with proof* η_T.
- $\mathsf{VerifyOutput}(\mathsf{pk}, T, \pi_T, \eta_T)$ *checks the output and proof.*

We require correctness and uniqueness: every verified output for a fixed tag is the same. Before enough shares are available to reconstruct, an output must remain pseudorandom given the public execution and the adaptively revealed states of at most f *permanently corrupted replicas over the whole execution. In the ideal analysis the outputs are fresh independent uniform permutations. A cryptographic implementation gives computational security with an error bound.*

View v uses the tag $T_v = (v, \texttt{priority})$[9] and reconstructs the priority permutation π_v with proof η_v. The permutation's restriction to the k leaders is uniform. In particular, the priority order remains uniform given the execution up to the proposal step and the keys revealed by Byzantine corruption before the voting deadline.

Release and reconstruction. Let s_v denote the start time of view v. At $s_v + \Delta$, every active honest replica broadcasts its beacon share. A replica uses $\mathsf{Combine}$ on Q verified shares from distinct replicas to reconstruct the priority permutation π_v and its proof η_v. Permutations and proofs received from other replicas are checked using $\mathsf{VerifyOutput}$.

When Q honest replicas complete their share broadcasts at the voting deadline after GST, a recipient that remains active and honest throughout $(s_v + \Delta, s_v + 2\Delta]$ can reconstruct by $s_v + 2\Delta$. The adversary may reconstruct the output as soon as shares are released at $s_v + \Delta$. Section 4 proves that enough honest replicas complete these broadcasts despite mobile crashes.

We give a reference construction in Appendix A.

3 Protocol and Safety

ML-FVS extends fixed-view Simplex to multiple leaders. Views are numbered from 1. Each view v starts at $s_v = 3v\Delta$ and ends at $s_v + 3\Delta$. Its k distinct leaders are publicly known by the view start. Each replica votes on a set of proposals it received by $s_v + \Delta$. Replicas verify and broadcast every certificate when they first form or receive it. At $s_v + 3\Delta$, they enter view $v + 1$, sign no further messages in view v, and continue processing and forwarding certificates from earlier views. Figure 1 illustrates the view schedule.

Proposal rule. Pseudocode 1 gives the rule for each leader. A valid proposal $\langle \texttt{Propose}, v, j, b, w \rangle$ from leader $L_v[j]$ includes an externally valid block b of consensus payload, the view number v, and a reference w, which is either an earlier view number or a special value denoting genesis. A vote certificate $\mathsf{VC}_w(b')$

[9] We consider a single protocol execution. If keys are reused across executions, messages and beacon tags must also include a session identifier.

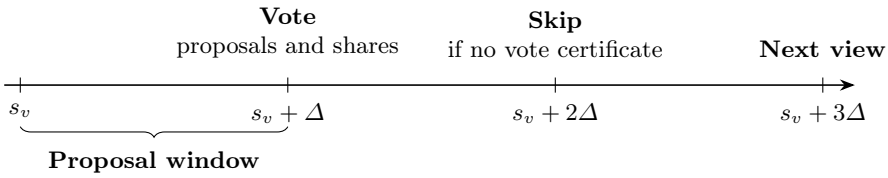

Fig. 1. One view of ML-FVS. After GST, replicas that remain active and honest reconstruct the beacon by $s_v + 2\Delta$. The next view starts at $s_v + 3\Delta$.

Pseudocode 1: Proposal rule for leader j in view v.

```
1 when the local clock reaches s_v:
2     if some w is justified then
3         choose the highest justified w
4         broadcast a valid proposal ⟨Propose, v, j, b, w⟩
5     else
6         send nothing
```

records quorum support for block b' in view w, and a skip certificate SC_i records quorum votes to skip view i. Their formation rules are given below. For an earlier view $w < v$, the proposal includes $\mathsf{VC}_w(b')$ for a block b' that b extends, and SC_i for every view $w < i < v$. For genesis, b extends genesis and the proposal includes SC_i for every prior view $i < v$. At s_v, each leader j proposes using the highest w for which it has the required certificates, treating genesis as lower than every view. We call such a reference justified. It sends nothing if no such reference exists.

Voting rule and beacon reveal. Pseudocode 2 gives the voting rule. At time $s_v + \Delta$, each active honest replica i sends a vote $\langle \texttt{Vote}, v, R_i, \sigma_{i,v} \rangle$. Here R_i is its set of recorded valid proposals for view v, and $\sigma_{i,v} = \mathsf{Share}(\mathsf{sk}_i, T_v)$ is its beacon share. After verifying shares for tag $T_v = (v, \texttt{priority})$ and reconstructing the beacon order from Q valid shares as in Section 2.2, each valid vote resolves to the (leader, block) pair of its highest-priority proposal, or to $\perp$ if the set is empty. A vote certificate consists of Q valid votes from distinct replicas that resolve to the same leader and block, together with the beacon output and its proof. Figure 2 illustrates vote resolution under a priority order.

Beacon shares are verified with $\mathsf{VerifyShare}$ and counted once per replica toward reconstruction.[10] A received beacon output is checked against its proof using $\mathsf{VerifyOutput}$. At the voting deadline, each active honest replica fixes the set of proposals included in its vote before processing any shares released at that deadline. The set cannot subsequently change.

[10] A valid share can still be used even if the vote contains an invalid proposal or more than one proposal from the same leader.

Pseudocode 2: Voting rule for proposals in view v.

Record:
  when *a valid proposal for view v from leader j is received by $s_v + \Delta$, before your vote set is fixed*:
    record it only if it is the first for leader j
Vote:
  when *the local clock reaches $s_v + \Delta$*:
    fix R as the set of recorded proposals (possibly $\emptyset$)
    broadcast $\langle \texttt{Vote}, v, R, \mathsf{Share}(\mathsf{sk}_i, T_v) \rangle$
Resolve:
  when *Q shares from distinct replicas passing* $\mathsf{VerifyShare}$ *for T_v have been collected*:
    $(\pi_v, \eta_v) \leftarrow \mathsf{Combine}(\mathsf{pk}, T_v, S)$, for the collected shares S
    resolve each valid `Vote`, including later arrivals, to the highest-priority (leader, block) pair in its signed set R, or to $\perp$ if $R = \emptyset$
Vote certificate:
  when *Q valid* `Vote` *messages from distinct replicas have been collected and resolve to the same pair with block b*:
    form $\mathsf{VC}_v(b)$

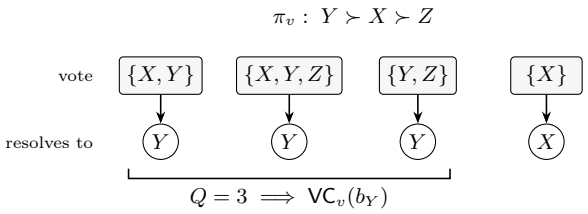

Fig. 2. Example vote resolution for $Q = 3$. Under $Y \succ X \succ Z$, three votes resolve to Y and form $\mathsf{VC}_v(b_Y)$.

A valid vote is correctly signed and contains only valid proposals for its view, with at most one per leader. An honest replica signs at most one vote per view, with its single beacon share.

Finalization and skip rules. Pseudocode 3 gives the finalization and skip rules. An active honest replica that obtains $\mathsf{VC}_v(b)$ signs and broadcasts a finalization vote $\langle \texttt{Final}, v, b \rangle$ if it has signed neither a finalization vote nor a skip vote in view v. At $s_v + 2\Delta$, it signs and broadcasts a skip vote $\langle \texttt{Skip}, v \rangle$ if it has no vote certificate and has signed neither a finalization vote nor a skip vote. A final certificate $\mathsf{FC}_v(b)$ consists of Q finalization votes for b; a skip certificate SC_v consists of Q skip votes. Each certificate requires signatures from distinct replicas for the same view. Replicas form certificates whenever the required signatures arrive. A vote certificate and a skip certificate may both exist. When an honest replica obtains $\mathsf{FC}_v(b)$, it finalizes b and its ancestors. It appends the newly finalized blocks to its output log, with each block following its parent.

Communication. Apart from beacon setup and certificate forwarding, the scheduled protocol steps use at most k proposal broadcasts, n vote broadcasts, and n

Pseudocode 3: Voting rule for finalization and skip votes in view v.

```
1  Final:
2  |  when VC_v(b) is obtained:
3  |  |  if you signed neither Final nor Skip in view v then
4  |  |  |  sign and broadcast ⟨Final, v, b⟩
5  Skip:
6  |  when the local clock reaches s_v + 2Δ, after you have processed delivered
   |   messages:
7  |  |  if no VC_v(·) is known and you signed neither Final nor Skip in view v then
8  |  |  |  sign and broadcast ⟨Skip, v⟩
9  Decide:
10 |  when FC_v(b) is obtained:
11 |  |  decide b
```

finalization or skip broadcasts, for $O(kn + n^2)$ point-to-point messages in a view where replicas follow these steps.

Each vote contains up to k proposals. If a complete proposal, including its validity proof and justifications, takes at most P bytes, votes use $O(kn^2P)$ bytes in total. A vote certificate contains Q votes, so broadcasting one full vote certificate from every replica can add $O(kn^3P)$ bytes.

3.1 Safety

We prove safety regardless of message delays. Any two quorums intersect in at least $f + 1$ replicas. Since at most f replicas ever become Byzantine, the intersection contains an always honest signer, even if later corruptions reveal keys for past views. We defer proofs to Appendix B.

Every vote certificate contains an honest vote, which resolves only to an externally valid block.

Lemma 1 (External validity). *Every block certified by a vote certificate is externally valid.*

Observe that two vote certificates in the same view always share an honest signer, whose vote must resolve to the same leader and block pair.

Lemma 2 (Unique vote certificate). *In any view, vote certificates can certify at most one* (*leader*, *block*) *pair.*

Similarly, a final certificate and a skip certificate in the same view would share an honest signer, who cannot sign both.

Lemma 3 (Final and skip certificates). *In any view, a final certificate and a skip certificate cannot both exist.*

A finalized view cannot be skipped, so later valid proposals must extend its finalized block.

Lemma 4 (Agreement). *At most one block is finalized in each view, and every block finalized in a later view extends every block finalized in an earlier view.*

4 Liveness

We first establish quorum availability and certificate catch-up under the full fault model. We then analyze liveness with fixed Byzantine faults, followed by mobile crashes and adaptive Byzantine corruption. The probability bounds below use the ideal beacon with independent uniform outputs in Definition 2. We defer proofs to Appendix C.

The delayed refund means that a crash continues to count against the fault budget after healing. This leaves a quorum that stays active and honest throughout each interval of length Δ.

Lemma 5 (Active honest quorums). *For every time t, at least $Q = n - f$ replicas remain active and honest throughout $(t, t + \Delta]$, including the actions at its end.*

Between voting and skipping, this quorum either forwards a vote certificate or produces enough skip votes to form a skip certificate. Since messages are buffered for crashed replicas, they are also able to obtain these once they are healed. The following lemma extends the certificate catch-up argument of [1].

Lemma 6 (Certificate catch-up). *For every view w, any honest replica active at time $t \geq \max\{\mathrm{GST}, s_w + 2\Delta\} + \Delta$ has either $\mathsf{VC}_w(b)$ for some block b, or SC_w.*

Hence, at each view start $s_v \geq \mathrm{GST} + \Delta$, active honest leaders have the certificates needed to make justified proposals.

4.1 Liveness with Fixed Byzantine Faults

We now consider a set of at most f replicas that are Byzantine from the start, with no further corruptions or crashes. We establish liveness when the highest-priority leader is honest.

Lemma 7 (Liveness with fixed Byzantine faults). *With a fixed set of at most f Byzantine replicas and no crashes, every view v with $s_v \geq \mathrm{GST} + \Delta$ whose highest-priority leader is honest, produces a final certificate for its block at every honest replica by $s_v + 3\Delta$.*

Infinitely many views with an honest highest-priority leader give infinitely many final certificates. If honest leaders propose externally valid proper extensions, agreement then gives an unbounded finalized chain.

4.2 Liveness with Mobile Crashes

We now study liveness under mobile crashes, using hidden priorities to derive good-case success probabilities and expected latency bounds.

Lemma 8 (A completed highest-priority proposal). *Suppose a post-*GST *view's highest-priority leader completes a valid proposal broadcast at s_v and remains honest at every time strictly before $s_v + \Delta$. Then an active honest replica obtains a final certificate for that proposal by $s_v + 3\Delta$.*

Lemma 9 (Liveness when leaders remain honest). *Suppose a view v with $s_v \geq \mathrm{GST} + \Delta$ has $k > c$ honest leaders, and no leader may become Byzantine during that view, regardless of the priority outcome. Then an active honest replica obtains a final certificate by $s_v + 3\Delta$ with probability at least $(k - c)/k$. This holds for both $D = 0$ and $D = \Delta$.*

Good case. For $D = \Delta$, a view v with $s_v \geq \mathrm{GST} + \Delta$ is a *good-case view* if all its leaders are honest at s_v and no permanent attack started before s_v is pending against any of them. The adversary may now start new Byzantine attacks during the view, but these cannot take effect before the voting deadline.

Lemma 10 (Good-case liveness). *Let $D = \Delta$. In a good-case view with $k > c$ leaders, an active honest replica obtains a final certificate by $s_v + 3\Delta$ with probability at least $(k - c)/k$.*

With $c+1$ leaders that remain honest until voting, the crash budget leaves at least one leader able to complete its proposal broadcast. Because the priority order is hidden until after proposals, the adversary can only guess which leader will rank first. This gives the two-leader guarantee highlighted in the introduction.

Corollary 1 (Small leader sets). *Under Lemma 9 or Lemma 10, $k = c + 1$ gives success probability at least $1/(c + 1)$. In particular, two leaders give probability at least $1/2$ with $c = 1$, even with $f - 1$ Byzantine replicas outside the set.*

When the same conditions hold in successive views, each independently sampled priority order gives another chance to finalize, even after earlier failures. The probability of a long run of failures therefore decreases geometrically, and each additional attempt costs 3Δ.

Corollary 2 (Expected latency). *Let v_0 be a post-*GST *view chosen using only the history available at its start. Suppose the conditions of either Lemma 9 or Lemma 10 hold in each view from v_0 until a final certificate is obtained, with the same $k > c$.*

Let N count the views from v_0 up to and including the first view that produces a final certificate. Then

$$\Pr[N > m] \leq (c/k)^m, \qquad \mathbb{E}[N] \leq \frac{k}{k - c}.$$

The expected time from s_{v_0} is at most $3\Delta k/(k-c)$. For $k = c+1$ this is $3\Delta(c+1)$, hence $O(\Delta)$ for constant c.

4.3 Adaptive Byzantine Corruption

We now consider views whose leaders may already be targeted for Byzantine corruption. The sampling argument counts leaders outside these targets that complete their proposal broadcasts at s_v despite crashes. Since the priority order is uniform and remains hidden during proposals, each of these leaders has probability $1/k$ of ranking first. If no new Byzantine corruption starts, these leaders remain honest, so the view finalizes whenever one of them ranks first. The supporting lemmas and bounds for general $k > c$ are in Appendix C.

Lemma 11 (Sampling bound). *Consider a view v with $s_v \geq \mathrm{GST} + \Delta$ and $k = c + 1$ leaders.*[11] *For either corruption delay ($D \in \{0, \Delta\}$), with probability at least $(2/3)^k/k$, by the view's end, an active honest replica obtains a final certificate or the adversary spends a Byzantine corruption.*

When leaders are drawn at the view's start, delayed corruptions started at or after the draw cannot take effect before the voting deadline. The same counting argument then gives probability at least $(2/3)^k/k$ of finalization in each view, even if new Byzantine corruptions are started. Repeated attempts give the following expected latency.

Theorem 2 (Expected latency with delayed Byzantine corruption). *Let $D = \Delta$ and $k = c + 1$. Suppose each view v with $s_v \geq \mathrm{GST} + \Delta$ draws its public k-element leader set exactly at s_v. Then the expected time from the start of any such view until an active honest replica obtains a final certificate is at most $3\Delta k(3/2)^k$, which is $O(\Delta)$ for constant c.*

The leader draw in Theorem 2 requires randomness separate from the priority beacon, which is released only at the voting deadline. Announcing leaders before the view starts would give the adversary time to begin corrupting them before s_v.

The adversary can start at most f Byzantine corruptions, so eventually no new ones occur. For every fixed set of f corruption targets, independent uniform sampling produces infinitely many leader sets with more than c replicas outside that set, with probability one. Hidden priorities then give repeated chances to finalize, even when corruptions take effect immediately and the leader schedule is public in advance.

Theorem 3 (Liveness). *Let $k > c$. Active honest replicas obtain final certificates in infinitely many views both for $D = 0$ and $D = \Delta$.*

4.4 Single Leaders and Known Priorities

We illustrate why the protocol uses multiple leaders and keeps their priority order hidden until voting.

[11] The leader set may be announced in advance.

Lemma 12 (A single leader). *For $f \geq 1$, a mobile adversary with $c = 1$ can prevent the $k = 1$ protocol from producing any final certificate, without starting any Byzantine corruption.*

Even with multiple leaders, an adversary that knows their priorities can split the votes and prevent finalization in a view.

Example 1 (Known priorities). Let $n = 4$, $f = c = 1$, and $k = 2$, with no Byzantine replicas. If the priority order is known before proposals, crash the highest-priority leader i during its broadcast, after the proposal has been sent to itself and one other replica. Heal it before the voting deadline and delay the messages to the remaining recipients by Δ. Let the proposal from the other leader j reach every replica after i heals and before the voting deadline. Two vote sets then contain both proposals, and two contain only j's proposal. Leader i gets two resolved votes, as does j, so neither reaches the quorum of three. If the order was instead hidden and uniform while this delivery pattern was fixed, j would rank first with probability $1/2$; all four votes would then resolve to j and the view would finalize.

5 Reliability and Proposal Timing

We next study whether multiple leaders improve reliability when leaders can increase their rewards by delaying proposals. For our evaluation, we use a probabilistic network model with empirical link-delay distributions.[12] While this network model captures a realistic simulation setting as the latencies are derived from real-world measurements, we also add stochastic faults that model random software and hardware failures that can delay proposals [23].

Moreover, observe that in an implementation of ML-FVS with k leaders, the timing of the leaders' actions directly affects the outcome of each view. We therefore model each leader as rational, as timing games have been studied theoretically [25,31] and observed in practice [34]. We argue that a realistic common-case evaluation requires taking them into account. We do not model adaptive attacks or recovery from mobile crashes. We use two placements: a global placement uniform across cities around the world, and a regional placement concentrated in North America and Europe (Fig. 3).

We consider one view with k public leaders and an available priority beacon. Proposals are valid and justified at view start, and replicas follow the voting rules. Each directed link has a known delay distribution, from which message delays are sampled independently. Here, Δ is the voting deadline measured from view start and need not bound message delays. Formally, each leader independently stalls with probability p. A stall adds a delay of Δ^{stall} to its proposal to all recipients. We define the resulting timing game below, then compare view-failure rates at equilibria.

[12] For the empirical distributions, we used round-trip time (RTT) quantiles from CloudPing, accessed on 11 August 2026.

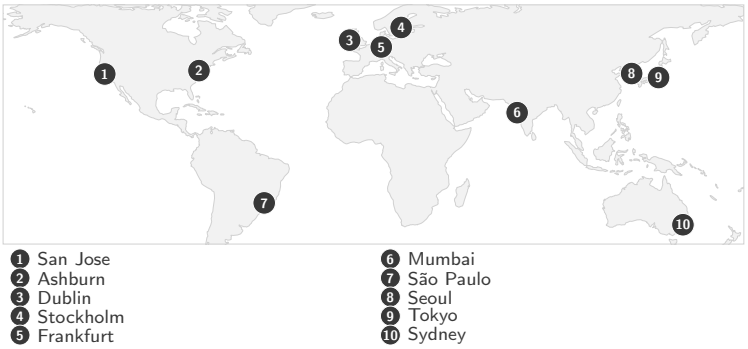

Fig. 3. Locations of 100 replicas. The global placement assigns ten to each city. The regional placement assigns 40 to North America, 40 to Europe, and 20 elsewhere.

Value of delay. In many blockchains, delaying a proposal can increase its value because more transaction fees and maximal extractable value (MEV) can be collected. Following [28,31], we use an increasing reward function $V(s) = B + M(s)$, where B is the base reward and $M(s)$ is the value accumulated by proposal time s. Leaders trade off this increase in value against the risk that their proposals arrive too late to be finalized.

Definition 3 (Timing game).

1. *Each leader i chooses a proposal time $s_i \in [0, \Delta]$, where Δ is the voting deadline.*
2. *Leaders know one another's locations, but choose before network delays, stalls, and the priority order are revealed.*
3. *Leader i maximizes its expected reward, receiving $V(s_i)$ if its block is finalized and zero otherwise.*

The formal equilibrium definition and an analysis with equal link delays appear in Appendix D.

5.1 Evaluation Method

Setup. We use the two placements and delay model above, with $n = 100$, $Q = 67$, stall probability $p = 1\%$, stall delay $\Delta^{\text{stall}} = 500\,\text{ms}$, and $V(s) = 0.041 + 0.0065s$, with s in seconds and rewards in ETH. The initial value and growth rate follow the Ethereum parameters in [25]. Unless stated otherwise, $\Delta = 4$ seconds. The k leaders are sampled uniformly without replacement.[13]

[13] Our simulation code can be accessed here: `https://github.com/kalpturer/beyond-leaderless`.

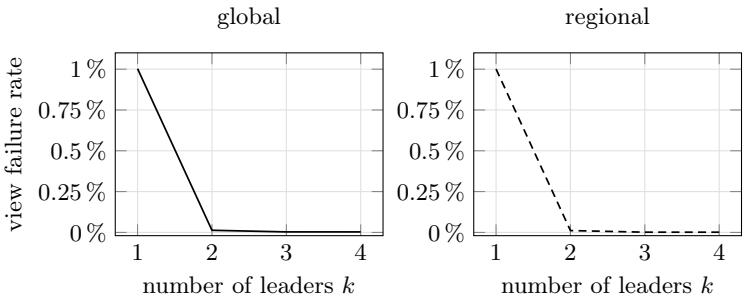

Fig. 4. View failure rates for one to four leaders at $\Delta = 4$ seconds.

View failures. The analysis assumes that, whenever the proposal votes support a vote certificate, at least $Q = 67$ replicas obtain it by 2Δ, and their finalization votes reach all replicas by 3Δ. Under this assumption, a vote certificate leads to a final certificate and prevents a skip certificate. We therefore measure view failure as the event that no vote certificate forms from the proposals received by the voting deadline.

Equilibria. Starting from prompt proposals, we update each leader's proposal time in turn to a best response on a 0.25 ms grid, holding the others' times fixed. We check that no leader can improve its expected reward by choosing another grid point. We compute the approximate equilibria on this grid. We vary Δ from 200 ms to 4 s for one and two leaders, then vary k from one to four at $\Delta = 4$ seconds. In both cases, we enumerate all leader-location configurations, weighting each by its probability under uniform sampling.

5.2 Single and Multiple Leaders

At $\Delta = 4$ seconds, Fig. 4 compares the computed view-failure rates for one to four leaders in both placements. Adding a second leader reduces the failure rate from about 1% to about 0.01%, even though leaders continue to delay their proposals at equilibrium.

More precisely, adding a second leader reduces the computed view-failure rate from 1.0029% to 0.01296% in the global placement. In the regional placement, it falls from 1.0012% to 0.01125%.

For $k = 3, 4$, computed failure rates are 0.003063% and 0.002964% in the global placement, and 0.001350% and 0.001251% in the regional placement. To understand the proposal times, consider a single leader. Its expected reward initially increases with block value, decreases when stalled proposals no longer arrive in time, and drops near the voting deadline as unstalled proposals also fail to obtain a quorum, as seen in Fig. 5 in Appendix E. The computed equilibria use the single-leader proposal time at each leader's location for all four leader counts. On average, leaders leave 107.15 ms in the global placement and 79.55 ms

in the regional placement between proposing and the voting deadline, well below the stall duration. The model reports no failures for a prompt single leader at this voting deadline (Fig. 6 in Appendix E). The remaining failures come from later proposal times. In this setting, multiple leaders therefore improve reliability even when rational leaders leave too little time to absorb a stall.

Acknowledgments. This work was done while the second author was interning at a16z frontier research.

AI disclosure. Generative AI tools, such as GPT-6 Astra, were used during the research process and preparation of this manuscript. The authors take full responsibility for its contents.

References

1. Abraham, I.: Simplex on a fixed view schedule. Blog post (May 2026), https://decentralizedthoughts.github.io/2026-05-14-simplex-FVS/
2. Abraham, I., Dolev, D., Gonen, R., Halpern, J.Y.: Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In: Proc. 25th ACM Symposium on Principles of Distributed Computing (PODC). pp. 53–62 (2006). https://doi.org/10.1145/1146381.1146393
3. Abraham, I., Dolev, D., Halpern, J.Y.: Distributed protocols for leader election: A game-theoretic perspective. In: Proc. 27th International Symposium on Distributed Computing (DISC). pp. 61–75 (2013). https://doi.org/10.1007/978-3-642-41527-2_5
4. Abraham, I., Malkhi, D., Spiegelman, A.: Asymptotically optimal validated asynchronous Byzantine agreement. In: Proc. 38th ACM Symposium on Principles of Distributed Computing (PODC). pp. 337–346 (2019). https://doi.org/10.1145/3293611.3331612
5. Abraham, I., Stern, G.: Consensus with one mobile crash in synchrony or one crash in asynchrony has infinite executions. Decentralized Thoughts (2024), https://decentralizedthoughts.github.io/2024-03-07-mobile-is-FLP/, march 7, 2024
6. Aguilera, M.K., Toueg, S.: A simple bivalency proof that t-resilient consensus requires $t + 1$ rounds. Information Processing Letters **71**(3–4), 155–158 (1999). https://doi.org/10.1016/S0020-0190(99)00100-3
7. Aiyer, A.S., Alvisi, L., Clement, A., Dahlin, M., Martin, J.P., Porth, C.: BAR fault tolerance for cooperative services. In: Proc. 20th ACM Symposium on Operating Systems Principles (SOSP). pp. 45–58 (2005). https://doi.org/10.1145/1095810.1095816
8. Alpturer, K., Babel, K., Saraf, A.: Timing games in responsive consensus protocols. arXiv:2510.25144 (2025)
9. Antoniadis, K., Desjardins, A., Gramoli, V., Guerraoui, R., Zablotchi, I.: Leaderless consensus. In: 41st IEEE International Conference on Distributed Computing Systems (ICDCS). pp. 392–402 (2021). https://doi.org/10.1109/ICDCS51616.2021.00045, Author version

10. Arun, B., Li, Z., Suri-Payer, F., Das, S., Spiegelman, A.: Shoal++: High throughput DAG BFT can be fast and robust! In: 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI). pp. 813–826 (2025), https://www.usenix.org/conference/nsdi25/presentation/arun
11. Babel, K., Chursin, A., Danezis, G., Kichidis, A., Kokoris-Kogias, L., Koshy, A., Sonnino, A., Tian, M.: Mysticeti: Reaching the limits of latency with uncertified DAGs. arXiv:2310.14821v6 (2025), https://arxiv.org/abs/2310.14821v6
12. Babel, K., Elsheimy, F., Heimbach, L., Jalalzai, M.M., Klenze, T., Komatovic, J., Milionis, J., Setrin, M., Shoup, V.: Cadence: Extreme pipelining with multiple concurrent proposers. arXiv:2607.02275v2 (2026), https://arxiv.org/abs/2607.02275v2
13. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. Cryptology ePrint Archive, Paper 2001/006 (2001), https://eprint.iacr.org/2001/006
14. Camenisch, J., Drijvers, M., Hanke, T., Pignolet, Y.A., Shoup, V., Williams, D.: Internet Computer consensus: Extended abstract. In: Proc. 41st ACM Symposium on Principles of Distributed Computing (PODC). pp. 81–91 (2022). https://doi.org/10.1145/3519270.3538430
15. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of Bitcoin without the block reward. In: Proc. 23rd ACM Conference on Computer and Communications Security (CCS). pp. 154–167 (2016). https://doi.org/10.1145/2976749.2978408
16. Chan, B.Y., Pass, R.: Simplex consensus: A simple and fast consensus protocol. IACR ePrint 2023/463 (2023)
17. Danezis, G., De Angeli, Z., Vitanov, A.I., Jovanovic, P., Kokoris-Kogias, L., Sonnino, A., Tennage, P., Zablotchi, I.: Barnacle: Adaptive multi-leader scheduling for DAG-based consensus. arXiv:2609.03978v2 (2026), https://arxiv.org/abs/2609.03978v2
18. drand contributors: drand protocol specification. Project documentation, https://docs.drand.love/docs/specification/, accessed September 22, 2026
19. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Financial Cryptography and Data Security. pp. 436–454. Springer (2014). https://doi.org/10.1007/978-3-662-45472-5_28, arXiv:1311.0243
20. Gafni, E., Losa, G.: Time is not a healer, but it sure makes hindsight 20:20. arXiv:2305.02295v2 (2024), https://arxiv.org/abs/2305.02295v2
21. Garimidi, P., Neu, J., Resnick, M.: Multiple concurrent proposers: Why and how. arXiv:2509.23984 (2025), https://arxiv.org/abs/2509.23984
22. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling Byzantine agreements for cryptocurrencies. In: Proc. 26th ACM Symposium on Operating Systems Principles (SOSP) (2017). https://doi.org/10.1145/3132747.3132757
23. Giridharan, N., Mishra, S., Alvisi, L., Crooks, N., Marsh, B., Meling, H., Nayak, K., Prusak, G.: Ambulance: Saving BFT through racing. In: 20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26). pp. 1325–1347. USENIX Association (Jul 2026), https://www.usenix.org/conference/osdi26/presentation/giridharan
24. Halpern, J.Y., Vilaça, X.: Rational consensus: Extended abstract. In: Proc. 35th ACM Symposium on Principles of Distributed Computing (PODC). pp. 137–146 (2016). https://doi.org/10.1145/2933057.2933088, arXiv:2005.10141

25. Kelkar, M., Tas, E.N., Bahrani, M., Roughgarden, T.: Designing incentives for responsive consensus protocols. Cryptology ePrint Archive, Paper 2026/1234 (2026), https://eprint.iacr.org/2026/1234
26. Lamport, L., Fischer, M.: Byzantine generals and transaction commit protocols. Tech. Rep. 62, Computer Science Laboratory, SRI International (1982), https://lamport.azurewebsites.net/pubs/trans.pdf, minor revision April 25, 1984
27. Li, P., Wang, G., Chen, X., Long, F., Xu, W.: Gosig: A scalable and high-performance byzantine consensus for consortium blockchains. In: ACM Symposium on Cloud Computing (SoCC) (2020). https://doi.org/10.1145/3419111.3421272, https://www.cs.toronto.edu/~fanl/papers/gosig-socc20.pdf
28. Öz, B., Kraner, B., Vallarano, N., Kruger, B.S., Matthes, F., Tessone, C.J.: Time moves faster when there is nothing you anticipate: The role of time in MEV rewards. arXiv:2307.05814 (2023)
29. Santoro, N., Widmayer, P.: Time is not a healer. In: STACS 89. Lecture Notes in Computer Science, vol. 349, pp. 304–313. Springer (1989). https://doi.org/10.1007/BFb0028994
30. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in Bitcoin. In: Financial Cryptography and Data Security (FC 2016). pp. 515–532. Springer (2017). https://doi.org/10.1007/978-3-662-54970-4_30, arXiv:1507.06183
31. Schwarz-Schilling, C., Saleh, F., Thiery, T., Pan, J., Shah, N., Monnot, B.: Time is money: Strategic timing games in Proof-of-Stake protocols. In: Proc. 5th Conference on Advances in Financial Technologies (AFT). pp. 30:1–30:17 (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.30, arXiv:2305.09032
32. Stathakopoulou, C., David, T., Pavlovic, M., Vukolić, M.: Mir-BFT: High-throughput robust BFT for decentralized networks. arXiv:1906.05552v3 (2021), https://arxiv.org/abs/1906.05552v3
33. Tennage, P., Basescu, C., Kokoris-Kogias, L., Syta, E., Jovanovic, P., Estrada-Galiñanes, V., Ford, B.: QuePaxa: Escaping the tyranny of timeouts in consensus. In: Proc. 29th ACM Symposium on Operating Systems Principles (SOSP). pp. 281–297 (2023). https://doi.org/10.1145/3600006.3613150
34. Wahrstätter, A.: On attestations, block propagation, and timing games. Ethereum Research (Aug 2024), https://ethresear.ch/t/on-attestations-block-propagation-and-timing-games/20272, august 14, 2024
35. Wong, E.L., Levy, I., Alvisi, L., Clement, A., Dahlin, M.: Regret freedom isn't free. In: Proc. 15th International Conference on Principles of Distributed Systems (OPODIS). pp. 80–95 (2011). https://doi.org/10.1007/978-3-642-25873-2_7
36. Xiang, Z., Tonkikh, A., Spiegelman, A.: Prefix consensus for censorship resistant BFT. arXiv:2602.02892v2 (2026), https://arxiv.org/abs/2602.02892v2

A Threshold Beacon Construction

Consider threshold BLS with hashed output [18]. Let $\mathsf{e} : G_1 \times G_2 \to G_T$ be a nondegenerate pairing of prime-order groups of order $q > n$, with generator $g_2 \in G_2$. Trusted setup, or a separately analyzed distributed key generation (DKG) protocol, supplies shares $x_i = p(\alpha_i)$ of a random polynomial over $\mathbb{F}_q$ of degree at most $2f$, at distinct nonzero public points α_i, conditioned on $x = p(0) \neq 0$. Replica i's private beacon share is $\mathsf{sk}_i = x_i$. Publish $Y = g_2^x$ and $V_i = g_2^{x_i}$, included in the public parameters pk. Use separate hashes H_1 to G_1 and H_2 to bit strings. Replica i produces $\sigma_i = H_1(T)^{x_i}$; its share check is

$$\mathsf{e}(\sigma_i, g_2) = \mathsf{e}(H_1(T), V_i).$$

For Q verified shares indexed by I, interpolate at zero:

$$\lambda_i = \prod_{j \in I \setminus \{i\}} \frac{-\alpha_j}{\alpha_i - \alpha_j}, \qquad \sigma = \prod_{i \in I} \sigma_i^{\lambda_i} = H_1(T)^x.$$

A fixed public permutation sampler uses $H_2(T, \sigma)$ as its seed to produce π_T. Return this permutation with proof $\eta_T = \sigma$. To verify a received permutation, check that it matches this seed and that $\mathsf{e}(\sigma, g_2) = \mathsf{e}(H_1(T), Y)$. Validate group membership and canonical encodings before these checks. Every set of Q verified shares for the same tag yields the same σ. Cryptographic unpredictability relies on appropriate BLS hardness and random-oracle assumptions and the setup's security.

B Proofs for Section 3 (Protocol and Safety)

Lemma 1 (External validity). *Every block certified by a vote certificate is externally valid.*

Proof. A vote certificate $\mathsf{VC}_v(b)$ contains an always honest replica's `Vote` resolving to b. Such `Vote` messages contain only valid proposals, so b is externally valid. □

Lemma 2 (Unique vote certificate). *In any view, vote certificates can certify at most one* (*leader*, *block*) *pair.*

Proof. Suppose vote certificates exist for pairs (j, b) and (j', b') in view v. Their signer sets must intersect in an always honest replica. An always honest replica signs at most one `Vote` in the view, and the common beacon order resolves that signed `Vote` to at most one pair. Hence $(j, b) = (j', b')$. □

Lemma 3 (Final and skip certificates). *In any view, a final certificate and a skip certificate cannot both exist.*

Proof. Suppose for a contradiction that both $\mathsf{FC}_v(b)$ and SC_v exist. Each certificate has Q distinct signers, so an always honest replica has signed both $\langle \texttt{Final}, v, b\rangle$ and $\langle \texttt{Skip}, v\rangle$. This contradicts the rule that an always honest replica signs `Final` or `Skip` only if it has signed neither in that view. □

Lemma 4 (Agreement). *At most one block is finalized in each view, and every block finalized in a later view extends every block finalized in an earlier view.*

Proof. Every final certificate contains an always honest `Final` signer and therefore has a vote certificate. By Lemma 2, all final certificates in a view certify the same block.

Suppose b is finalized in view v. Then $\mathsf{VC}_v(b)$ exists, and Lemma 3 rules out SC_v. We show by induction on $u \geq v$ that every vote certificate in view u certifies b or a descendant of b. The case $u = v$ follows from Lemma 2.

For $u > v$, an always honest signer of a vote certificate has checked the proposal's justification. A genesis justification, or a justification from a view below v, would require SC_v, which does not exist. The justification must therefore contain $\mathsf{VC}_w(b')$ for some $v \leq w < u$. By induction, b' is b or a descendant, and the certified proposal extends b'. Every final certificate in a later view thus certifies a descendant of b. □

C Proofs for Section 4 (Liveness)

Lemma 5 (Active honest quorums). *For every time t, at least $Q = n - f$ replicas remain active and honest throughout $(t, t + \Delta]$, including the actions at its end.*

Proof. If a replica is crashed at some time $u \in (t, t + \Delta]$, either it remains crashed at the end of the interval or it heals at some $h \geq u > t$. If it heals, its charge remains until $h + \Delta > t + \Delta$. Thus a crash charge for every such replica is outstanding at the end of the interval. Add the permanent charges of replicas Byzantine by that time. These charges total at most f, so at most f replicas have these charges. Every other replica is active and honest throughout the stated interval. A replica that heals at t may be in this quorum: its charge can be refunded at $t + \Delta$, but it is active throughout $(t, t + \Delta]$. □

Lemma 6 (Certificate catch-up). *For every view w, any honest replica active at time $t \geq \max\{\mathrm{GST}, s_w + 2\Delta\} + \Delta$ has either $\mathsf{VC}_w(b)$ for some block b, or SC_w.*

Proof. Fix a view w. By Lemma 5, a quorum H stays active and honest throughout $(s_w + \Delta, s_w + 2\Delta]$. Either one member forwards a vote certificate, or all members broadcast skip votes.

Assume $f \geq 1$, as the case $f = 0$ is immediate. Reconstructing the beacon requires another honest replica's share, so no honest replica obtains a vote certificate at or before $s_w + \Delta$. If a member of H obtains one by $s_w + 2\Delta$, it therefore

completes its broadcast during this interval. Otherwise, no member has signed a finalization vote, so all broadcast skip votes at $s_w + 2\Delta$.

The certificate or Q skip votes arrive by $\max\{\mathrm{GST}, s_w + 2\Delta\} + \Delta$ and are processed on arrival or upon healing. Hence every honest replica active at or after this time has a vote certificate or a skip certificate. □

Lemma 7 (Liveness with fixed Byzantine faults). *With a fixed set of at most f Byzantine replicas and no crashes, every view v with $s_v \geq \mathrm{GST} + \Delta$ whose highest-priority leader is honest, produces a final certificate for its block at every honest replica by $s_v + 3\Delta$.*

Proof. For every earlier view $u < v$, $s_u + 2\Delta \leq s_v - \Delta$, so $\max\{\mathrm{GST}, s_u + 2\Delta\} + \Delta \leq s_v$. By Lemma 6, every honest leader has a vote certificate or a skip certificate for every earlier view before its proposal step and can therefore make a justified proposal.

Let j be the highest-priority leader. Every honest replica receives its valid proposal by the voting deadline and includes it in its signed set. The votes and beacon shares of at least Q honest replicas reach every honest replica by $s_v + 2\Delta$. They reconstruct the same priority output and obtain a vote certificate for j's proposal before the skip deadline. By uniqueness of the vote certificate, any finalization vote already sent in this view is for the same block. These replicas all broadcast a finalization vote, and their Q messages arrive by $s_v + 3\Delta$, forming a final certificate. □

Lemma 8 (A completed highest-priority proposal). *Suppose a post-*GST *view's highest-priority leader completes a valid proposal broadcast at s_v and remains honest at every time strictly before $s_v + \Delta$. Then an active honest replica obtains a final certificate for that proposal by $s_v + 3\Delta$.*

Proof. We follow the proposal through three successive quorums: the first votes, the second broadcasts finalization votes, and the third receives them.

If $f = 0$, the single replica finalizes at the voting deadline. Assume $f \geq 1$. For $i \in \{0, 1, 2\}$, let H_i be the replicas active and honest throughout $(s_v + i\Delta, s_v + (i+1)\Delta]$, including the actions at its end. By Lemma 5, $|H_i| \geq Q$.

Every replica in H_0 receives or retains the proposal by the voting deadline. The leader cannot equivocate before the voting deadline. Even if it becomes Byzantine exactly at the voting deadline, a conflicting proposal sent then cannot reach another replica before its vote set is fixed. Each replica in H_0 therefore includes the same proposal, and its vote resolves to that proposal.

Later faults cannot erase the sent messages or change the signed vote sets.

Every replica in H_1 receives the Q matching votes and beacon shares from H_0 by $s_v + 2\Delta$, reconstructs the priority order, and obtains the matching vote certificate before the skip deadline. An honest replica cannot sign a finalization vote at or before the voting deadline: reconstruction requires another honest replica's share, first released at the voting deadline and received after positive transmission time. Nor can it sign a skip vote before the scheduled skip step. Crashes preserve this signing state. Any earlier finalization vote is for the same

block by Lemma 2. Thus all replicas in H_1 broadcast a finalization vote at times in $(s_v + \Delta, s_v + 2\Delta]$.

Every replica in $H_1 \cap H_2$ is active and honest throughout $(s_v + \Delta, s_v + 3\Delta]$ and receives these finalization votes by $s_v + 3\Delta$. Since $|H_1 \cap H_2| \geq 2Q - n = f + 1$, an active honest replica obtains a final certificate by the view's end. □

Lemma 9 (Liveness when leaders remain honest). *Suppose a view v with $s_v \geq \mathrm{GST} + \Delta$ has $k > c$ honest leaders, and no leader may become Byzantine during that view, regardless of the priority outcome. Then an active honest replica obtains a final certificate by $s_v + 3\Delta$ with probability at least $(k - c)/k$. This holds for both $D = 0$ and $D = \Delta$.*

Proof. After refunds due at s_v, any crash charge outstanding at or added during the proposal step remains until after that step. These charges cover at most c leaders. Every other leader is active, has the certificates needed to propose by Lemma 6, and completes its valid broadcast at s_v. There are at least $k - c$ such leaders. These leaders are fixed before the priority beacon is revealed, so one ranks first with probability at least $(k - c)/k$. On this event, Lemma 8 gives the final certificate. □

Lemma 10 (Good-case liveness). *Let $D = \Delta$. In a good-case view with $k > c$ leaders, an active honest replica obtains a final certificate by $s_v + 3\Delta$ with probability at least $(k - c)/k$.*

Proof. By the good-case condition and $D = \Delta$, every leader remains honest strictly before the voting deadline. As in Lemma 9, at most c leaders have outstanding crash charges or are crashed during the proposal step. At least $k - c$ leaders therefore complete their broadcasts, and Lemma 8 applies when one of these ranks first. □

Corollary 2 (Expected latency). *Let v_0 be a post-*GST *view chosen using only the history available at its start. Suppose the conditions of either Lemma 9 or Lemma 10 hold in each view from v_0 until a final certificate is obtained, with the same $k > c$.*

Let N count the views from v_0 up to and including the first view that produces a final certificate. Then

$$\Pr[N > m] \leq (c/k)^m, \qquad \mathbb{E}[N] \leq \frac{k}{k - c}.$$

The expected time from s_{v_0} is at most $3\Delta k/(k-c)$. For $k = c+1$ this is $3\Delta(c+1)$, hence $O(\Delta)$ for constant c.

Proof. Condition on the history at s_{v_0}. The conditional failure probability in every view is at most c/k. Multiplying these bounds gives the tail bound; summing it for $m \geq 0$ gives the expectation. □

Let B_v contain the targets of all permanent attacks started by s_v, including attacks still pending. Write $b = |B_v|$ and $h = |L_v \setminus B_v|$, and let

$$d_b = \min\{c, f - b\}.$$

At most d_b crash charges can be outstanding if no new permanent attack starts.

Lemma C.1 (Finalization or a new permanent attack). *Consider a view v with $s_v \geq \mathrm{GST} + \Delta$. For either corruption delay ($D \in \{0, \Delta\}$), with probability at least $\max\{h - d_b, 0\}/k$, by the view's end, an active honest replica obtains a final certificate or the adversary starts a new permanent attack during the view.*

Proof. First consider an adversary that starts no new permanent attacks in the view. At least $\max\{h - d_b, 0\}$ leaders outside B_v are free of outstanding crash charges at the proposal step and complete their justified broadcasts by Lemma 6. These leaders remain honest. The priority order is uniform conditional on their completed broadcasts, so Lemma 8 gives success probability at least $\max\{h - d_b, 0\}/k$.

For an adversary that may start new attacks, consider a second execution with the same randomness. It follows the original execution until the first new permanent attack in the view, omits that attack, and then continues with any allowed behavior that starts no new permanent attacks. The earlier bound applies to the second execution. The executions differ only if the original adversary starts a new permanent attack. Otherwise, a final certificate obtained in the second execution is also obtained in the original. Thus success in the second execution implies one of the two outcomes in the lemma. □

Lemma C.1 uses the number h of leaders outside the targets. When the targets are fixed before the leader set is drawn, we can average over the draw.

Lemma C.2 (Leader and priority sampling). *Consider a view v with $s_v \geq \mathrm{GST} + \Delta$. Let B contain the targets of all permanent attacks started before L_v is drawn, and let $b = |B|$. For either corruption delay ($D \in \{0, \Delta\}$), with probability at least*

$$q(b) = \frac{1}{k\binom{n}{k}} \sum_{h=d_b+1}^{k} (h - d_b) \binom{n-b}{h} \binom{b}{k-h},$$

by the view's end, an active honest replica obtains a final certificate or the adversary starts a new permanent attack after the draw.

Proof. First consider an execution with no new permanent attacks. The set B is fixed before selection. For $H = |L_v \setminus B|$, we have

$$\Pr[H = h] = \frac{\binom{n-b}{h}\binom{b}{k-h}}{\binom{n}{k}}.$$

The adversary may adapt its crashes to the public set, but at most d_b leaders can have outstanding crash charges at the proposal step. At least $\max\{H - d_b, 0\}$

leaders complete valid broadcasts. Their identities are fixed before the priority output is known, giving probability at least $\max\{H - d_b, 0\}/k$ that one ranks first. Taking expectation gives $q(b)$. The comparison of executions in Lemma C.1, now omitting the first new permanent attack after the set draw, proves the stated bound when the adversary may start new attacks. □

If the schedule is announced in advance, the adversary may target leaders after seeing L_v; any such attack starts after the draw and so counts as the second outcome in Lemma C.2. The bound $q(b)$ still depends on the number of targets b, which the adversary chooses. We next show that $q(b)$ is smallest at $b = f - c$, which gives a bound that holds for every choice of targets.

Lemma 11 (Sampling bound). *Consider a view v with $s_v \geq \mathrm{GST} + \Delta$ and $k = c + 1$ leaders.*[14] *For either corruption delay ($D \in \{0, \Delta\}$), with probability at least $(2/3)^k/k$, by the view's end, an active honest replica obtains a final certificate or the adversary spends a Byzantine corruption.*

Proof. We first bound $q(b)$ for every $k > c$. With B and b as in Lemma C.2 and $H = |L_v \setminus B|$, the bound in Lemma C.2 is

$$q(b) = \frac{1}{k}\mathbb{E}\big[\max\{H - d_b, 0\}\big].$$

We show that $q(b) \geq q(f - c) > 0$, with $q(f - c) \geq (2/3)^k/k$ when $k = c + 1$. For $b \leq f - c$, $d_b = c$, and increasing b can only decrease the number of sampled leaders outside B. For $b \geq f - c$, use the same leader set for two target sets, one obtained by adding a target to the other. Adding one target reduces H by at most one, while reducing $d_b = f - b$ by exactly one. Hence $\max\{H - d_b, 0\}$ cannot decrease. This proves the minimum at $f - c$. In particular,

$$q(f - c) = \frac{1}{k\binom{n}{k}} \sum_{h=c+1}^{k} (h - c)\binom{n - f + c}{h}\binom{f - c}{k - h} > 0.$$

Positivity follows because $k > c$ and $n - f + c > c$, so a uniform leader set has positive probability of containing more than c replicas outside a fixed set of $f - c$ targets. When $k = c + 1$, the positive term requires all k leaders to avoid B. Thus

$$q(f - c) = \frac{\binom{n-f+c}{k}}{k\binom{n}{k}}.$$

For $0 \leq j \leq c$ and $n = 3f + 1$, $(n - f + c - j)/(n - j) \geq 2/3$, which gives the last inequality. Combining this bound with Lemma C.2 proves the lemma. □

Theorem 2 (Expected latency with delayed Byzantine corruption). *Let $D = \Delta$ and $k = c + 1$. Suppose each view v with $s_v \geq \mathrm{GST} + \Delta$ draws its public k-element leader set exactly at s_v. Then the expected time from the start of any such view until an active honest replica obtains a final certificate is at most $3\Delta k(3/2)^k$, which is $O(\Delta)$ for constant c.*

[14] The leader set may be announced in advance.

Proof. Corruptions started at or after the leader draw cannot take effect before voting. We use this to bound the probability of finalization in each view, then the expected number of views.

Fix $k > c$. Let B contain the replicas targeted for Byzantine corruption before this view's draw, including pending corruptions, and write $b = |B|$. The set B is fixed before selection. Hence the distribution of $H = |L_v \setminus B|$ is the one given in Lemma C.2. At most $d_b = \min\{c, f - b\}$ leaders can be excluded by outstanding crash charges or by crashes during the initial proposal step. At least $\max\{H - d_b, 0\}$ leaders outside B therefore complete justified broadcasts at s_v.

These leaders remain honest strictly before the voting deadline. The adversary does not know the priority order when their broadcasts complete. With probability at least $\max\{H - d_b, 0\}/k$, one of them ranks first and Lemma 8 gives a final certificate. Taking expectation over H gives $q(b)$ for a final certificate.

The proof of Lemma 11 gives $q(b) \geq q(f-c) > 0$. Set $q_\star = q(f-c)$, and let N count this view and each subsequent view through the next final certificate. The success probability is at least $q_\star$ conditional on every earlier history. Multiplying the conditional failure bounds and summing the tail gives

$$\Pr[N > m] \leq (1 - q_\star)^m, \qquad \mathbb{E}[N] \leq \frac{1}{q_\star}.$$

Thus the expected time is at most $3\Delta/q_\star$. For $k = c+1$, the proof of Lemma 11 gives $q_\star \geq (2/3)^k/k$, yielding the stated bound. □

Theorem 3 (Liveness). *Let $k > c$. Active honest replicas obtain final certificates in infinitely many views both for $D = 0$ and $D = \Delta$.*

Proof. Infinitely many views give a probability of at least $1/k$ of either finalization or a new Byzantine corruption. Only finitely many corruptions can occur, so infinitely many of these views must finalize.

For any fixed set B of at most f replicas, an independently sampled k-element set has positive probability of containing more than c replicas outside B, since $n - f > c$ and $k > c$. Independence across views makes this occur infinitely often with probability one. There are only finitely many such sets B, so with probability one this property holds for every one of them.

Let B^* contain all Byzantine corruption targets, so $|B^*| \leq f$. The property above applies to B^* even when it depends on the public schedule. For each view v, let $h_v = |L_v \setminus B_v|$, using the actual target set at its start. Since $B_v \subseteq B^*$, every set with more than c replicas outside B^* has $h_v > c$. Thus infinitely many views v with $s_v \geq \mathrm{GST} + \Delta$ satisfy $h_v > c$. This condition is known before the view's priority reveal and does not use the future set B^*.

At each such view, Lemma C.1 gives conditional probability at least $1/k$ that a final certificate is obtained or a new permanent attack is started. The probability of neither outcome in the next m such views is at most $(1 - 1/k)^m$. The same tail argument after every fixed view shows that one of these two outcomes occurs infinitely often with probability one. New permanent attacks

can be started in only finitely many views, so infinitely many must produce a final certificate. □

Lemma 12 (A single leader). *For $f \geq 1$, a mobile adversary with $c = 1$ can prevent the $k = 1$ protocol from producing any final certificate, without starting any Byzantine corruption.*

Proof. At each view start, crash the public leader before it sends its proposal. Heal it at $s_v + \varepsilon$ for $0 < \varepsilon < \Delta$. Even if it immediately resumes its broadcast, delay the messages to other replicas by Δ, past the voting deadline. At most its own vote supports the proposal, fewer than Q, so neither a vote certificate nor a final certificate can form. The crash charge is refunded against both budgets at $s_v + \Delta + \varepsilon$, before the next view. All replicas are active at the skip deadline and can receive a skip certificate by the next view start. This attack can repeat indefinitely. □

D Timing Equilibria and Equal Link Delays

D.1 Leader Timing

A single leader's optimal proposal time satisfies

$$s^* \in \arg\max_{0 \leq s \leq \Delta} V(s) \Pr[\text{the leader's block is finalized} \mid s].$$

With multiple leaders, let $U_i(s_i, s_{-i})$ be leader i's expected reward when the other leaders use proposal times s_{-i}.

Definition D.1 (Timing equilibrium). *Proposal times $s^* = (s_1^*, \ldots, s_k^*)$ form a* pure timing equilibrium *if no leader can increase its expected reward by changing its proposal time alone:*

$$s_i^* \in \arg\max_{a \in [0,\Delta]} U_i(a, s_{-i}^*) \qquad \textit{for every leader } i.$$

D.2 Equal Link Delays

Suppose every link between distinct replicas has the same fixed delay $\ell > 0$, while self-delivery is immediate. Each leader stalls independently with probability $p \in (0, 1)$. A stall delays its proposal at every recipient, including its sender, by $\Delta^{\text{stall}} > 0$.

Assume $f \geq 1$, $1 \leq k \leq f + 1$, and $\Delta \geq \ell + \Delta^{\text{stall}}$. The latest proposal times that allow delivery to every replica by the voting deadline, with and without a stall, are respectively

$$s_{\text{safe}} = \Delta - \ell - \Delta^{\text{stall}}, \qquad s_{\text{late}} = \Delta - \ell.$$

For sufficiently small p, the increase in V gives $(1 - p)V(s_{\text{late}}) > V(s_{\text{safe}})$. Otherwise, proposing at s_{safe} is optimal, and every view finalizes when all leaders choose this time.

Theorem D.1 (Equilibrium timing and reliability). *Suppose V is positive and strictly increasing on $[0, \Delta]$, and $(1-p)V(s_{\text{late}}) > V(s_{\text{safe}})$. With equal link delays, the unique timing equilibrium has every leader propose at s_{late}, and the probability of no final certificate by view end is p^k.*

Proof. Call a proposal timely if it reaches every replica by the voting deadline. We first show that a view finalizes exactly when at least one proposal is timely, then compare the expected rewards from s_{safe} and s_{late}.

The highest-priority timely proposal receives at least $n - (k-1) \geq Q$ votes, since only senders of higher-priority proposals that are not timely can resolve their votes differently. A proposal that is not timely receives at most its sender's vote and cannot form a certificate. Every replica obtains the resulting vote certificate by $\Delta + \ell < 2\Delta$ and the final certificate by $\Delta + 2\ell < 3\Delta$.

Fix the other leaders' proposal times. Conditional on being timely, a leader's probability of winning is positive and independent of its own proposal time: stalls are independent, and the beacon selects uniformly among timely proposals. Since V is strictly increasing, the only possibilities for an optimal proposal time are s_{safe}, which is always timely, and s_{late}, which is timely with probability $1-p$. Later proposals cannot win. The two choices give expected rewards proportional to $V(s_{\text{safe}})$ and $(1-p)V(s_{\text{late}})$, respectively. The inequality makes s_{late} the unique best response, regardless of the other leaders' choices. At this equilibrium, a view fails exactly when all k leaders stall. Independence gives failure probability p^k. □

Thus, a second leader reduces the failure probability from p to p^2 without changing proposal times.

E Additional Evaluation Figures

Figure 5 shows the single-leader reward curves. Figure 6 compares view-failure rates across voting deadlines.

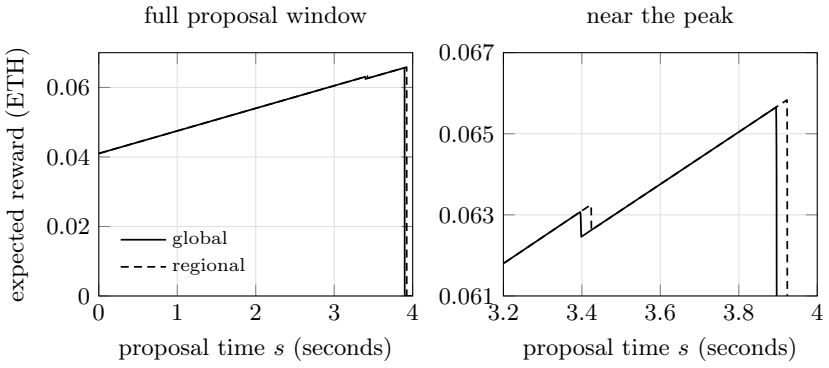

Fig. 5. Expected reward for a single leader in Frankfurt, with voting deadline $\Delta = 4$ seconds. The right plot shows the curves near their peaks.

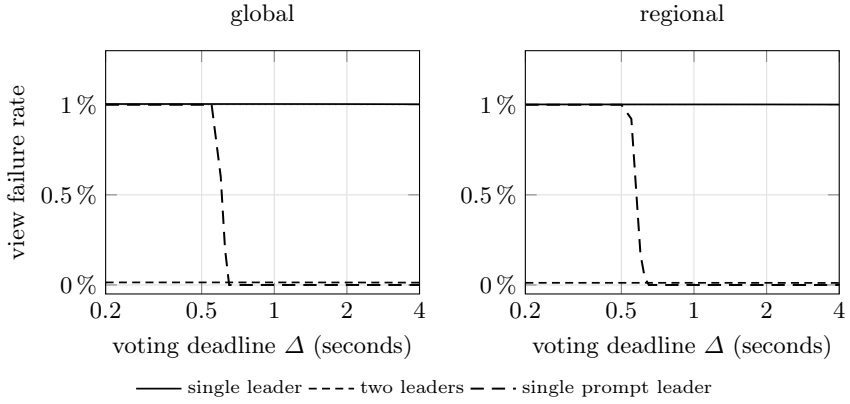

Fig. 6. View failure rates in the global (left) and regional (right) placements using the computed proposal times for one and two leaders, compared with a prompt single leader. Proposals independently stall for 500 ms with probability 1% (dotted line).

F Related Work

Consensus with multiple proposers. Many consensus protocols use multiple proposers [11, 12, 21].

Mir-BFT [32] distributes proposal work across parallel leaders. MCP [21] combines proposals under one leader for censorship resistance. Ambulance [23] uses a rigged race in which the leader has a distinct advantage. QuePaxa's asynchronous core [33] uses private random proposal priorities. Its crash-fault model differs from our setting with permanent adaptive Byzantine corruptions. Gosig [27] also selects one proposal from a small random leader set, using privately computed scores revealed with proposals.

A certified outcome for every proposer. Cadence [12] implements a virtual leader whose success in a slot depends on obtaining a certified outcome for every proposer. When each proposer either completes its broadcast at slot start or remains silent, an honest quorum can certify inclusion or exclusion and finalize in three message delays. A crash during the broadcast can instead divide the votes so that neither outcome has a quorum. One such proposer prevents the slot from completing through the fast path, even if every other proposer succeeds. Appendix G.1 gives an execution with one mobile crash that forces this outcome after GST. Cadence then starts fallback agreement for the same slot. In our good case, a completed broadcast from the highest-priority leader suffices for finalization.

Secret leader selection. Algorand [22] uses cryptographic sortition to let participants privately learn their selection and reveal proof when they send a message. Our leaders are public, but their priority order remains unpredictable, including to Byzantine leaders, until honest beacon shares are released at the voting deadline.

Publicly ranked proposals. The multi-slot protocol built from Prefix Consensus [36] uses a public proposal ranking. An earlier proposer that equivocates or sends to only some replicas can prevent later proposals from being committed in that slot. The protocol identifies that proposer and gives it lower priority in later slots.

Ordered leaders in DAG protocols. Mysticeti [11] orders leader slots and stops output at the first unresolved slot. Partial dissemination can leave a slot without the evidence required either to commit or to skip it, delaying later blocks even when their leaders have enough support. A later leader can resolve the earlier slot through the indirect decision rule. Shoal++ [10] uses different rules: within each DAG it evaluates one consensus instance at a time and chooses later leaders as that instance resolves. If a later leader causes the initial leader to be skipped, intermediate tentative leaders can also be skipped. Recent work on Barnacle [17] adapts the number of leader slots to balance lower latency against delays caused by slow leaders. These designs order multiple proposals, whereas our view forms

one certificate for a leader whose priority is revealed after the adversary has chosen which broadcasts to interrupt.

Leaderless and adaptive designs. ICC publicly ranks backup leaders [14] and delays their proposals according to rank. VABA runs n parallel protocol instances and randomly selects one after they have run [4]. Antoniadis, Desjardins, Gramoli, Guerraoui, and Zablotchi [9] define a consensus protocol to be *leaderless* if every correct process decides in every eventually synchronous execution allowing one process to be suspended per round. In their message-passing model, a suspended process sends and receives nothing throughout a request–reply round of two message delays. Their BFT-Archipelago protocol tolerates one such suspension together with $f-1$ Byzantine processes, and terminates deterministically within $O(n)$ rounds in the worst case.

The classical mobile-fault lower bound rules out deterministic consensus with one faulty sender per round, even under synchrony [5, 29]. Gafni and Losa [20] study equivalent formulations. A crash can interrupt a broadcast after partial dissemination, which whole-round suspension excludes.

Incentives in consensus. Strategic block withholding in Bitcoin shows how participants can benefit from deviating from a consensus protocol [15, 19, 30]. In timing games, leaders can delay proposals to collect more valuable transactions, at the risk of missing the voting deadline. This tradeoff has been modeled for Ethereum [31]. Other work studies how to encourage earlier proposals [8, 25]. Our evaluation finds fewer failed views with multiple leaders, even though leaders still benefit from delaying. Other work studies rational behavior in distributed systems outside blockchains [2, 3, 7, 24, 35].

G Mobile Crashes and an Interrupted Proposer

Classical lower bound. The classical lower bound states that any deterministic consensus protocol with agreement and unanimity validity has a nonterminating execution under send omissions affecting one sender per step, even in synchrony [29]. A crash can interrupt a broadcast after messages have been sent to some recipients, and the adversary can later move to another replica. Mobile delays and crashes give related formulations and proofs [5, 20].

A mobile crash that delays Mysticeti. Mysticeti orders leader slots, each identifying a proposer and a round. It directly skips a slot when $Q = 2f + 1$ blocks in the next round have no parent from that proposer. Direct commitment instead requires Q certificates for the proposal. Its output follows the fixed slot order and stops at the first unresolved slot [11, Sections II.C and III.B, Algorithms 1–3].

Consider four replicas A, B, C, D, with $f = c = 1$ and no Byzantine corruptions. In round r, let the proposer slots be ordered D, A, B, with D the primary proposer. Replica A sends its proposal only to B before crashing. Keep A crashed until the other replicas have constructed their blocks for rounds $r + 1$ and $r + 2$.

Arrange delivery so that B knows A's proposal when it creates its round-$r+1$ block, whereas C and D create their round-$r+1$ blocks before learning that proposal. Deliver B's proposal to all three active replicas before their voting blocks are created. The resulting references are

round-$(r+1)$ block	supports A	supports B
B_{r+1}	yes	yes
C_{r+1}	no	yes
D_{r+1}	no	yes.

These blocks are valid: each can reference the three round-r blocks of B, C, D, and B_{r+1} also references A's block. Deliver the three voting blocks and their dependencies, and let each active replica's round-$r+2$ block reference all three. Every such block certifies B, while none certifies A. Thus B has a direct-commit certificate, but A has only one supporting voting block and two that do not support it. Neither count reaches three, so A is neither directly committed nor directly skipped.

Mysticeti waits for the primary proposer, here D. Let B, C, D enter the proposal round at time 0 after GST, and let A enter at $\Delta/2$. Deliver A's proposal to B at $3\Delta/4$. Replicas C, D create their voting blocks at $\Delta/4$, after receiving the proposals of B, C, D. Replica B receives C's proposal at $\Delta/4$ and D's at Δ, so it votes after receiving A's proposal. Delay C, D's voting blocks to B until $5\Delta/4$ so they do not reveal D's proposal earlier. Later delivery provides the dependencies needed to validate the voting blocks.

Mysticeti's indirect rule considers only slots in rounds strictly greater than the unresolved slot's decision round, which is $r+2$ for its default three-round configuration. Hence a later slot in round r cannot resolve it. Algorithm 3 stops ordered output at A, even though B is already marked for commitment. A resolving proposal must come from round $r+3$ or later.

G.1 A Mobile Crash That Forces Cadence to Use Fallback

Cadence requires a quorum of matching inclusion or exclusion votes for each proposer before its fast path can commit the slot [12, Section 4.2 and Algorithms 2–4]. The execution below uses one immediate crash, a budget refund delay $\tau = \Delta$, and no Byzantine corruption.

Consider four replicas P, A, B, C, with $f = c = 1$, and two honest proposers P, A. The slot starts at time 0 after GST, and its proposal deadline is Δ. Fix $0 < \varepsilon < \Delta/4$.

1. P sends its assigned chunk to itself and the chunk assigned to A, then crashes before sending to B, C.
2. Heal P at time ε. It resumes sending to B, C. Deliver these messages at $\Delta + \varepsilon$, a delay of exactly Δ. Deliver A's chunk from P at $\Delta - \varepsilon$, and the other proposer's chunks to their recipients, by $\Delta - \varepsilon$, with P's chunk from A arriving at that time.

3. By the deadline, P, A vote to include P's proposal, while B, C vote to exclude it. Any chunks forwarded after reception at $\Delta - \varepsilon$ can arrive after the deadline while respecting the network delay bound. This also covers early voting. All four replicas vote to include A's proposal.
4. Neither set of two votes for P reaches the quorum of three. Chunks received later do not change the signed votes. Thus, no certified outcome for P can be formed by this protocol, and the slot requires fallback.

Cadence's fallback starts a separate multivalued Byzantine agreement protocol for the same slot [12, Section 4.3].