TECHNICAL REPORT
YL-2010-0007

DISSECTING ZAB

Flavio Junqueira, Benjamin Reed, and Marco Serafini
Yahoo! Labs
701 First Ave
Sunnyvale, CA 94089
{fpj,breed,serafini@yahoo-inc.com}

Bangalore • Barcelona • Haifa • Montreal • New York
Santiago • Silicon Valley

Yahoo! Labs Technical Report No. YL-2010-0007

Yahoo! Labs Technical Report No. YL-2010-0007

DISSECTING ZAB

Flavio Junqueira, Benjamin Reed, and Marco Serafini
Yahoo! Labs
701 First Ave
Sunnyvale, CA 94089
{fpj,breed,serafini@yahoo-inc.com}

1. Introduction

In this document, we present Zab, *the ZooKeeper atomic broadcast protocol*. ZooKeeper uses Zab to propagate state updates generated by a primary process, and we refer to these state updates as *transactions*. The primary produces a sequence of non-commutative transactions and broadcast them in order. To guarantee a consistent application state, ZooKeeper requires that every replica process delivers a prefix of the sequence of transactions. This property is necessary to guarantee that the state used to generate the state update in a transaction corresponds to the one the transaction is applied against.

Zab satisfies a different set of properties compared to previous atomic broadcast protocols in the literature, in particular with respect to causality. We still refer to it as an atomic broadcast protocol because, at a high level, we require that processes deliver the same set of transactions and in the same order. To distinguish from previous protocols in the literature, we call the broadcast problem we solve *primary-order atomic broadcast*, or simply *PO atomic broadcast*.

Here is a quick list of highlights of the protocol:

1. The protocol proceeds in epochs, and each epoch can have at most one process broadcasting messages;
2. Once a process executes the first phase of the protocol, it promises to not accept proposals from previous epochs. This property is critical to guarantee that during recovery no chosen proposal is missing from the selected initial history of transactions;
3. For a given epoch, all processes that participate in the epoch must have the initial history of transactions as a prefix of its own history;
4. To guarantee that transactions are generated using a consistent state, we need to make sure that a process has completed recovery before it broadcasts new transactions.

This documents has the following structure:

- Section 2 defines sequences and related operations. We use sequences for many parts, so it is important to make clear our assumptions for sequences;
- Section 3 presents our system model. It discusses processes, channels, and calls available;
- Section 4 presents Zab and its invariants;
- Section 5 proves that Zab satisfy five core properties. It also discusses the relation to causal order;
- Section 6 proves auxiliary lemmas that we use to show the core set of theorems;
- Section 7 discusses claims that we naively found ourselves trying to prove every so often.

2. Background: Sequences

We make extensive use of sequences in this proof, and the definitions we use follow closely the ones in the work of Lamport on TLA+ [3]. A sequence $S = s_1 \cdot s_2 \cdots s_k$ of length k is an ordered list of elements of some domain D. We use $\langle\rangle$ to denote the empty sequence. A sequence can be represented as a function with domain $1 \ldots k$ and range D: $S : [i \in 1 \ldots k \rightarrow s[i] \in D]$.

If S and S' are sequences, we use $S \circ S'$ to denote the concatenation of the two sequences. More formally:

$$S \circ S' = [i \in 1 \ldots |S| + |S'| \rightarrow \text{If } i \leq |S| \text{ then } S[i] \text{ else } S'[i]] \tag{2.1}$$

We also define the set of elements $E(S)$ of the sequence S as:

$$E(S) = \{e \in D : \exists i \in 1 \ldots |S| : S[i] = e\} \tag{2.2}$$

and the precedence relation $\prec_S$ between elements in the sequence S:

$d \prec_S d'$ iff there are $S[i] = d$ and $S[j] = d'$ such that $i < j$, $d, d' \in D$.

3. System model

A system comprises a set $\Pi = \{p_1, p_2, \ldots, p_n\}$ of processes that can crash and recover. If a process is not crashed, then we say it is *up*; otherwise we say that it is *down*. Processes communicate by exchanging messages through channels.

3.1. Processes

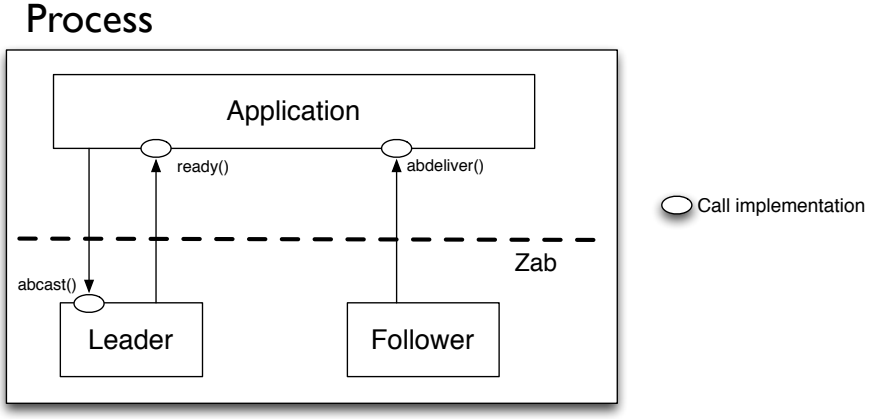

Figure 1: Process model.

Each process implements three roles: application, leader, and follower. As illustrated in Figure 1, the application role executes application steps, whereas the leader and the follower roles correspond to roles of Zab. When up, a process always performs the application and the follower roles. A process might also perform the leader role if the leader oracle outputs its process identifier. If a process is a leader, then it also executes the primary steps of the application.

Algorithms 1 and 2 implement skeletons for the application and Zab, respectively. The application algorithm uses three variables:

- `isReady` indicates whether a process is ready to broadcast;
- `instance` indicates the primary instance and the epoch of transaction identifiers;
- `loracle` is an oracle variable: a given process reads the value of `loracle` to determine which process is the current leader. Note that this oracle variable is used in both algorithms.

The Zab algorithm has two upon clauses that refer to steps enabled. These steps correspond to the ones described in Section 4, and they map to actions enabled upon preconditions becoming true. An example of a precondition is receiving a message of a particular type.

Algorithm 1 Application for process p

```
1: isReady← false
2: instance← 0
3: txns ← ∅
4: upon a leader change according to loracle
5:     ready ← false
6: end upon
7: upon a call to ready(e) // Primary
8:     isReady← true
9:     instance ← e
10:    z ← ⟨e, 1⟩
11: end upon
12: upon a new value v to broadcast // Primary
13:    if (loracle = p) ∧ isReady then
14:        abcast(⟨v, z⟩);
15:        increment z;
16:    end if
17: end upon
18: upon a call to abdeliver(⟨v, z⟩)
19:    transactions ← transactions ∪ ⟨v, z⟩
20: end upon
```

Algorithm 2 Zab for process p

```
1: upon a follower step enabled
2:     Execute follower step
3: end upon
4: upon a leader step enabled and loracle = p
5:     Execute leader step
6: end upon
7: upon a leader change according to loracle
8:     Discard the content of all input and output buffers
9:     Start new iteration of the Zab protocol and proceed to Phase 1
10: end upon
```

Zab proceeds in epochs. In each epoch, only a primary process in Π broadcasts. Such a primary broadcasts transactions $\langle v, z\rangle$, where value $v \in V$, V is the set of broadcast values, $z \in \mathcal{Z}$, and $\mathcal{Z}$ is a totally ordered set of transaction identifiers. Each transaction identifier $z = \langle e, c\rangle$ has two components: an epoch $e \in \mathbb{N}$ and a counter $c \in \mathbb{N}$. We use *epoch*(z) to denote the

epoch of a transaction identifier and *counter*(z) to denote the counter value of z. Incrementing a transaction identifier z corresponds to incrementing the counter c. A transaction identifier z precedes an identifier z', $z \prec_z z'$, if either *epoch*$(z) <$ *epoch*(z') or *epoch*$(z) =$ *epoch*(z') and *counter*$(z) <$ *counter*(z'). We also use $z \preceq_z z'$ to denote that either $z \prec_z z'$ or $z = z'$

To broadcast a transaction, a primary invokes *abcast*$(\langle v, z\rangle)$, and processes deliver a transaction by invoking *abdeliver*$(\langle v, z\rangle)$. When processing *abdeliver*$(\langle v, z\rangle)$, we simply introduce it to the set `txns`. We assume that the application uses this set to generate and maintain its state, and it is out of the scope of this work to describe how this procedure is executed. Because `txns` is a set, by definition multiple deliveries of a transaction $\langle v, z\rangle$ are not reflected to the application state. The exact semantics of broadcasting and delivering are given by five core properties, which we present and prove below: *broadcast integrity, total order, local primary order, global primary order, primary integrity*. We additionally require primary uniqueness for epochs and show two causal order properties that follow from the core properties.

The application algorithm implements and exposes the *abdeliver*$(\langle v, z\rangle)$ call, which is called by the follower code in the same process. For the order of broadcast and deliver events in a process, we use $\epsilon \prec_{p_i} \epsilon'$ to denote that process p_i has executed ϵ before ϵ', where ϵ and ϵ' are either an *abcast*$(\langle v, z\rangle)$ or an *abdeliver*$(\langle v, z\rangle)$ event.

Before broadcasting any transaction in a given epoch e, a primary must wait until the follower in the same process proceeds to the broadcast phase (Phase 3, see below) and its state must reflect all transactions delivered by its corresponding follower by the end of Phase 2. For this purpose, the application exposes a call *ready*(e) that a follower uses to enable the application to broadcast transactions. The call to *ready*(e) also serves the purpose of setting the value of the primary instance (variable `instance`). We show below that by construction this value is unique to a primary.

3.2. Channels

For each channel c_{ij} between a pair of processes p_i and p_j, we assume that processes p_i and p_j have each an output buffer and an input buffer, similarly to the model of [1]. A call to send a message m to process p_j is represented by an event *send*(m, p_j), which inserts m into the output buffer of p_i for c_{ij}. Messages are transmitted following the order of send events, and they are inserted into the input buffer. A call to receive the next message m in the input buffer is represented by an event *recv*(m, p_i).

Recall from Algorithm 2 that the Zab protocol proceeds in iterations (Line 9). To state the channel properties, we refer to iterations, and we assume that iterations are identified uniquely. Note that processes are not necessarily aware of such an identification scheme, since we use it only for formalization purposes.

Let $\sigma^{i,j}_{k,k'}$ be the sequence of messages process p_i sends to process p_j while iteration of p_i is k and iteration of p_j is k'. Every valid sequence of *send* and *recv* events must satisfy the following:

Integrity: If there is an event *recv*(m, p_i) at process p_j that returns m, then there must be a previous event *send*(m, p_j) at process p_i;

Prefix: If there is an event *recv*(m, p_i) in process p_j, $m \in \sigma^{i,j}_{k,k'}$, and there is m' such that $m' \prec m$ in $\sigma^{i,j}_{k,k'}$, then there must be an event *recv*(m', p_i) in process p_j that precedes *recv*(m, p_i);

Single Iteration: The input buffer of a process p_j for channel c_{ij} contains messages from at most one iteration of each process.

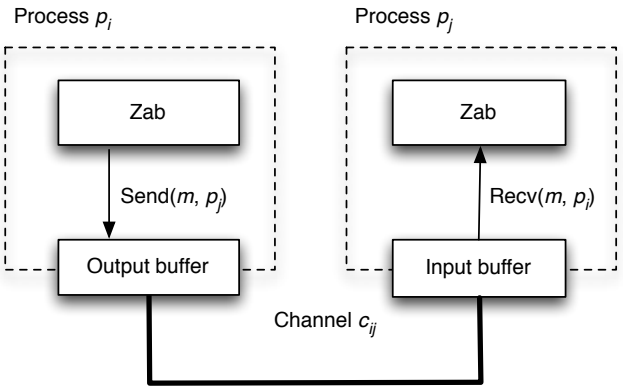

Figure 2: Channel model.

4. Zab: definitions, description, and invariants

According to the system model, each process is equipped with a leader oracle `loracle` that outputs a process identifier of a prospective leader. If the oracle of process $p_i \in \Pi$ says that p_i is the leader, then p_i executes the leader steps along with the follower steps. Otherwise, it only executes the follower steps.

An execution of Zab proceeds in epochs, and for each epoch, a process executes up to three phases. Recall that we expose three calls between Zab and the application:

- *abcast*$(\langle v, z\rangle)$;
- *abdeliver*$(\langle v, z\rangle)$;
- *ready*(e).

We first present some terminology that we use in the following sections, and next a description of the protocol and protocol invariants.

4.1. Terminology

Definition 1. (R phase) R phase is the recovery phase (Recovery phase: Phase 1 + Phase 2);

Definition 2. (B phase) B phase is the broadcast phase (Broadcast phase: Phase 3);

Definition 3. (Transaction) A *transaction* is a pair $\langle v \in V, z \in \mathcal{Z}\rangle$, where V is the set of broadcast values and $\mathcal{Z}$ is the set of transaction ids. The set $\mathcal{Z}$ is totally ordered and we use $\prec_z$ to denote a strict total order relation over the elements of $\mathcal{Z}$;

Definition 4. (Proposal) A *proposal* is a pair: current epoch, transaction.

Definition 5. (Broadcast transactions) β_e is the sequence of transactions broadcast by a primary of epoch e using *abcast*$(\langle v, z\rangle)$.

Definition 6. (Primary mapping) The primary mapping *primary* is a function from epoch numbers to processes: *primary* : $\mathbb{N} \rightarrow 2^{\Pi} \cup \{\perp\}$. Process $p_i \in$ *primary*(e) if there is a call to *ready*(e) in p_i; $\perp \in$ *primary*(e) *iff* no process has called *ready*(e).

Definition 7. (History) Each follower f has a history h_f of accepted transactions. A history is a sequence. We also use h_f^e to refer to the sequence of transactions follower f accepted while $f.p = f.a = e$. If follower f has not joined e, then $h_f^e = \perp$.

Definition 8. (Learning) A follower learns a transaction $\langle v, z \rangle$ by learning that a quorum Q of followers has accepted a proposal $\langle e, \langle v, z \rangle \rangle$, for some e.

Definition 9. (Established leader) A leader ℓ_e is established for epoch e if the *NEWLEADER*(e, I_e) proposal of ℓ_e is accepted by a quorum Q of followers.

Definition 10. (Established epoch) An epoch e is established if there is an established leader for e.

Definition 11. (Initial history) The initial history of an epoch e, I_e, is the history of a prospective leader of e at the end of phase 1 of epoch e. The initial history is defined only if the prospective leader of e completes Phase 1.

Definition 12. (Chosen transaction) A transaction $\langle v, z \rangle$ is chosen when a quorum of followers accept a proposal $\langle e, \langle v, z \rangle \rangle$ for some e.

Definition 13. (Sequence of chosen transactions) C_e is the sequence of chosen transactions in epoch e. A transaction $\langle v, z \rangle$ is chosen in epoch e iff there exists a quorum of followers Q such that each $f \in Q$ has accepted $\langle e, \langle v, z \rangle \rangle$.

Definition 14. (Sequence of chosen proposals broadcast in the B phase) CB_e is the sequence of proposals chosen during the B phase of epoch e;

Definition 15. (Follower joins epoch) A follower *joins* epoch e if it accepts a *NEWLEADER*(e, I_e) proposal in Phase 2, and *participates* in epoch e if its current epoch $f.a$ is e;

Definition 16. (Earlier and later epochs) We say that an epoch e is earlier than an epoch e' to denote that $e < e'$. Similarly, we say that an epoch e is later that e';

Definition 17. (Sequence of transactions delivered) Δ_f is the sequence of transactions follower f uniquely delivered, which is the sequence induced by the identifiers of the elements in `txns`.

Definition 18. (Sequence of transactions delivered in the B phase) D_f is the sequence of transactions follower f delivered while in the B phase of epoch $f.a$.

Definition 19. (Last committed epoch of a follower) Given a follower f, we use $f.e$ to denote the last epoch e such that f has learned that e has been committed.

4.2. Zab description

Each process executes one iteration of this protocol at a time, and a process at any step may decide to drop the current iteration and start a new one in Phase 1. In the case the process is a leader, it executes both the leader and the follower steps of the algorithm concurrently. The three phases of the protocol are as follows:

Phase 1 (Discovery): Follower f and leader ℓ execute the following steps:

CEPOCH$(f.p)$	Last promise of follower e
NEWEPOCH(e)	Leader proposed epoch
ACK-E$(f.a, h_f)$	Acknowledgement of new epoch e
NEWLEADER(e, I_e)	New leader proposal
ACK-LD(e)	Acknowledgment of new leader proposal
COMMIT-LD(e)	Commit message for epoch e
$\langle e, \langle v, z\rangle\rangle$	Proposal message for transaction $\langle v, z\rangle$ in epoch e
ACK$(e, \langle v, z\rangle)$	Acknowledgement for proposal $\langle e, \langle v, z\rangle\rangle$
COMMIT$(e, \langle v, z\rangle)$	Commit for proposal $\langle e, \langle v, z\rangle\rangle$

Table 1: Summary of messages

$f.p$	Last new epoch proposal acknowledged by follower f, initially $\perp$
$f.a$	Last new leader proposal acknowledged by follower f, initially $\perp$
h_f	History of follower f, initially $\langle\rangle$
$f.zxid$	zxid of the last accepted transaction in h_f, initially $\perp$

Table 2: Summary of follower persistent variables

Step $f.1.1$ A follower sends to the prospective leader ℓ its last promise in a *CEPOCH*$(f.p)$ message.

Step $\ell.1.1$ Upon receiving *CEPOCH*(e) messages from a quorum Q of followers, the prospective leader ℓ proposes *NEWEPOCH*(e') to the followers in Q. Epoch number e' is such that it is later than any e received in a *CEPOCH*(e) message.

Step $f.1.2$ Once it receives a *NEWEPOCH*(e') from the prospective leader ℓ, if $f.p < e'$, then make $f.p \leftarrow e'$ and acknowledge the new epoch proposal *NEWEPOCH*(e'). The acknowledgment *ACK-E*$(f.a, h_f)$ contains the current epoch $f.a$ of the follower and its history. Follower completes Phase 1.

Step $\ell.1.2$ Once it receives a confirmation from each follower in Q, it selects the history of one follower f in Q to be the initial history $I_{e'}$. Follower f is such that for every follower f' in Q, $f'.a < f.a$ or $(f'.a = f.a) \wedge (f'.zxid \preceq_z f.zxid)$. Prospective leader completes Phase 1.

Phase 2 (Synchronization): Follower f and leader ℓ execute the following steps:

Step $\ell.2.1$ The prospective leader ℓ proposes *NEWLEADER*$(e', I_{e'})$ to all followers in Q.

Step $f.2.1$ Upon receiving the *NEWLEADER*(e', T) message from ℓ, the follower starts a new iteration if $f.p \neq e'$. If $f.p = e'$, then it executes the following actions atomically:

1. It sets $f.a$ to e';
2. For each $\langle v, z\rangle \in E(I_{e'})$, it accepts $\langle e', \langle v, z\rangle\rangle$, and make $h_f = T$.

Finally, it acknowledges the *NEWLEADER*$(e', I_{e'})$ proposal to the leader, thus accepting the transactions in T.

Step $\ell.2.2$ Upon receiving acknowledgements to the *NEWLEADER*$(e', I_{e'})$ from a quorum of followers, the leader sends a commit message to all followers and completes Phase 2.

Step $f.2.2$ Upon receiving a commit message from the leader, it delivers all transactions in the initial history $I_{e'}$ by invoking *abdeliver*$(\langle v, z\rangle)$ for each transaction $\langle v, z\rangle$ in $I_{e'}$, following the order of $I_{e'}$, and completes Phase 2.

Phase 3 (Broadcast): Follower f and leader ℓ execute the following steps:

Step $\ell.3.1$**:** Leader ℓ proposes to all followers in Q in increasing order of zxid, such that for each proposal $\langle e', \langle v, z\rangle\rangle$, *epoch*$(z) = e'$, and z succeeds all zxid values previously broadcast in e'.

Step $\ell.3.2$**:** Upon receiving acknowledgments from a quorum of followers to a given proposal $\langle e', \langle v, z\rangle\rangle$, the leader sends a commit *COMMIT*$(e', \langle v, z\rangle)$ to all followers.

Step $f.3.1$**:** Follower f initially invokes *ready*(e') if it is leading.

Step $f.3.2$**:** Follower f accepts proposals from ℓ following reception order and appends them to h_f.

Step $f.3.3$**:** Follower f commits a transaction $\langle v, z\rangle$ by invoking *abdeliver*$(\langle v, z\rangle)$ once it receives *COMMIT*$(e', \langle v, z\rangle)$ and it has committed all transactions $\langle v', z'\rangle$ such that $\langle v', z'\rangle \in E(h_f)$, $z' \prec_z z$.

Step $\ell.3.3$**:** Upon receiving a *CEPOCH*(e) message from follower f while in Phase 3, leader ℓ proposes back *NEWEPOCH*(e') and *NEWLEADER*$(e', I_{e'} \circ \beta_{e'})$.

Step $\ell.3.4$**:** Upon receiving an acknowledgement from f of the *NEWLEADER*$(e', I_{e'} \circ \beta_{e'})$ proposal, it sends a commit message to f. Leader ℓ also makes $Q \leftarrow Q \cup \{f\}$.

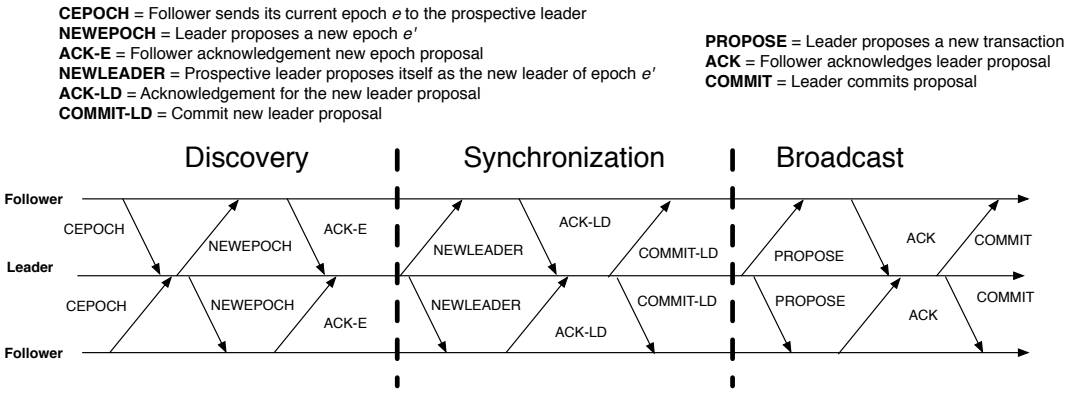

Figure 3: Zab protocol summary

4.3. Some insight

Here we give examples showing how Zab handles each one of them. These examples illustrate how the mechanisms Zab implements ensure that our safety properties hold.

In the two example we show in Figures 4 and 5, we have three processes, and for each process we show the follower history. In Example 1, we have that all three processes participate in moving to a new epoch and send their current history along with their latest epoch. Note that we omit the state of the leader in the figure, since the leader simply selects the initial history out of a set of histories received. The initial history chosen is the one of the follower with highest zxid, which is follower 1, and this history includes all transactions chosen so far.

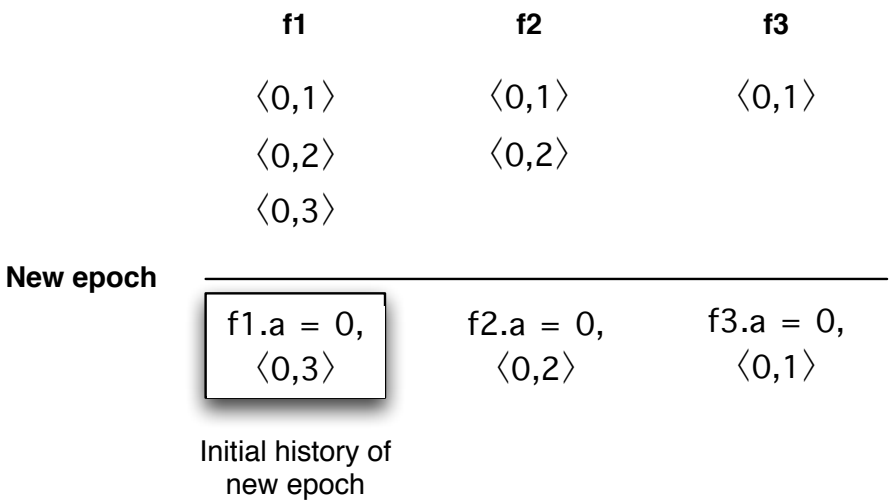

Figure 4: Example 1

Example 2 shows a scenario that leads to violation of safety. By the protocol, however, this scenario is impossible with Zab. From the example, if $f_3.a = 3$, then there must have been a quorum of followers promising not to accept transactions from an epoch earlier than 3 in Phase 1 of epoch 3. If such a promise is made before epochs 1 and 2 become established, then the leaders of epochs 1 and 2 would not be able to obtain a quorum in Phase 1 to establish these epochs. If the promise comes after epochs 1 or 2 are established, then the initial history of epoch 3 must include at least transaction $\langle 1, 1\rangle$. Consequently, this scenario is impossible with Zab.

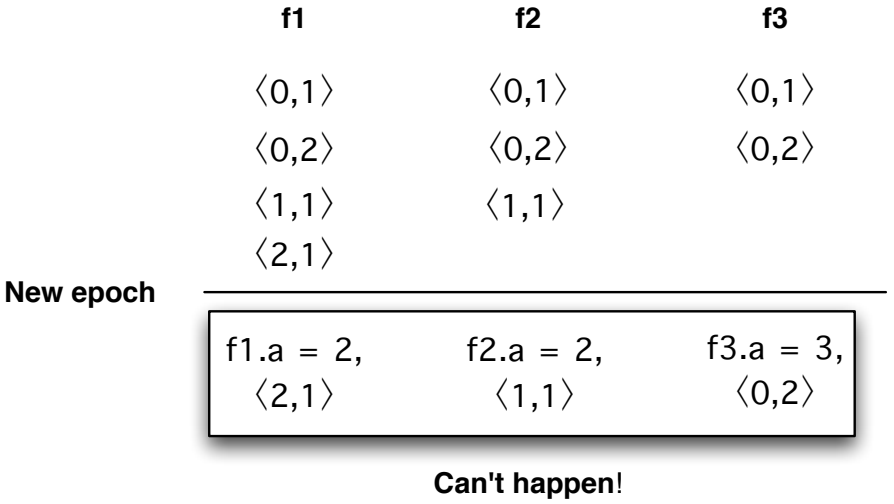

Figure 5: Example 2

4.4. Protocol invariants

The following are invariants guaranteed by the protocol.

Invariant 20. A process executes Phase 1, 2, and 3 in this order for any epoch e. A process can abandon its current epoch e at any time and start executing the steps of Phase 1 for a new epoch.

Invariant 21. Followers receive proposals during the B phase of an epoch e in zxid order.

Invariant 22. A follower f accepts a proposal $\langle e, \langle v, z\rangle\rangle$ only if its current epoch $f.a = e$.

Invariant 23. During the B phase of epoch e, a follower f such that $f.a = e$ accepts proposals and delivers transactions following the zxid order.

Invariant 24. A follower f sets its current epoch $f.a$ and accepts the leader history in Phase 2 atomically.

Invariant 25. In Phase 1, a follower f promises, before it provides its history h_f as the initial history of epoch e, not to accept proposals from the leader of any epoch $e' < e$.

Invariant 26. The initial history I_e of an epoch e is the history of some follower. Messages *ACK-E*$(f.a, h_f)$ (Phase 1) and *NEWLEADER*(e, I_e) (Phase 2) do not alter, reorder, or lose transactions in h_f and I_e, respectively.

Invariant 27. A follower delivers $\langle v, z\rangle \Rightarrow \exists Q \in \mathcal{Q} : \forall f \in Q : f$ has accepted $\langle e, \langle v, z\rangle\rangle$, for some epoch e.

Invariant 28. Let f be a follower. $D_f \sqsubseteq \beta_{f.e}$.

The integrity property of channels guarantees that messages are not corrupted or generated spontaneously. Corruption and spontaneous generation of a message might lead to incorrect behavior as the message content does not necessarily reflect the state of the sender. The prefix and single iteration properties enable the protocol to accept proposals and deliver transactions following the order of proposal and commit messages received, without further verification. In fact, Invariants 23 and 28 are trivially satisfied with these channel properties.

5. Properties

5.1. Primaries are unique

With the following claim, we show that for any epoch e there can be at most one primary.

Claim 29. Zab satisfies primary uniqueness: For every epoch number e, there is at most one process that calls *ready*(e).

FORMULA: $\forall e \in \mathbb{N} : |\mathit{primary}(e)| = 1$

Proof:
By the protocol, a process calls *ready*(e) once it becomes an established leader and it delivers the transactions in the initial history. By Lemma 46, there is at most one established leader for a given epoch e, and consequently the claim follows.
□

In the following sections, we use ρ_e to denote the unique primary of an established epoch e and $\rho_e \prec \rho_{e'}$ to denote that $e < e'$.

5.2. Core properties

Claim 30. Zab satisfies broadcast integrity: If some follower f delivers $\langle v, z\rangle$, then there is some epoch e such that the primary ρ_e has broadcast $\langle v, z\rangle$.

FORMULA: $\exists f \in \Pi, v \in V, z \in \mathcal{Z} : \langle v, z\rangle \in \Delta_f \Rightarrow \exists e : \langle v, z\rangle \in \beta_{\rho_e}$

Proof:
By the protocol, only transactions broadcast by primaries are delivered. By the integrity property of channels, only messages sent are received.
□

Claim 31. Zab satisfies agreement: If some follower f delivers $\langle v, z\rangle$ and some follower process f' delivers $\langle v', z'\rangle$, then f' delivers $\langle v, z\rangle$ or f delivers $\langle v', z'\rangle$.

FORMULA: $\langle v, z\rangle \in E(\Delta_f) \wedge \langle v', z'\rangle \in E(\Delta_{f'}) \Rightarrow \langle v, z\rangle \in E(\Delta_{f'}) \vee \langle v', z'\rangle \in E(\Delta_f)$

Proof:
If $\langle v, z\rangle = \langle v', z'\rangle$ or $f = f'$, then the claim is vacuously true. Assuming that $\langle v, z\rangle \neq \langle v', z'\rangle$ and $f \neq f'$, we have by Lemma 46 and the protocol, no two leaders propose different transactions with the same zxid. Suppose without loss of generality that $z \prec_z z'$. By assumption, we have that $\langle v, z\rangle \in E(\Delta_f)$. By Lemma 56, we have that $\langle v, z\rangle \in E(I_{f.e})$ or $\langle v, z\rangle \in E(D_f)$. There are two cases to consider:

epoch$(z) =$ ***epoch***(z') : By Invariant 23, followers accept $\langle v, z\rangle$ and $\langle v', z'\rangle$ following their zxid order. Assuming that $\langle v', z'\rangle \in E(D_{f'})$, we have also by Invariant 23 that:

$$\langle v, z\rangle \in E(D_{f'}) \tag{5.1}$$

Otherwise, we have that $\langle v', z'\rangle \in I_{f'.e}$. By the assumption that $\langle v, z\rangle$ has been delivered, we have that $\langle v, z\rangle$ has been chosen (Invariant 27). Consequently, by Lemma 55, we have also that $\langle v', z'\rangle \in I_{f'.e}$. By Lemma 56:

$$\langle v, z\rangle, \langle v', z'\rangle \in E(\Delta_{f'}) \tag{5.2}$$

epoch$(z) <$ ***epoch***(z') : By Invariant 22 and the protocol, we have that:

$$\langle v', z'\rangle \in E(\Delta_{f'}) \Rightarrow \textit{epoch}(z') \text{ has been established} \tag{5.3}$$
$$\langle v, z\rangle \in E(\Delta_f) \Rightarrow \exists e' : \langle v, z\rangle \in C_{e'} \tag{5.4}$$

By Lemma 55:

$$\begin{array}{ll} \bigwedge & \text{Eq. 5.3} \\ \bigwedge & \text{Eq. 5.4} \\ \Rightarrow & \langle v, z\rangle \in E(I_{\textit{epoch}(z')}) \end{array} \tag{5.5}$$

By Lemma 52:

$$\text{Eq. 5.5} \Rightarrow \langle v, z\rangle \in E(I_{f'.e}) \tag{5.6}$$

By assumption, we have that $\langle v', z'\rangle \in E(\Delta_{f'})$. By Lemma 56, we have that $\Delta_{f'} = I_{f'.e} \circ D_{f'}$, and we conclude that $\langle v, z\rangle, \langle v', z'\rangle \in E(\Delta_{f'})$

□

Claim 32. Zab satisfies total order: If some follower delivers $\langle v, z\rangle$ before $\langle v', z'\rangle$, then any follower that delivers $\langle v', z'\rangle$ must also deliver $\langle v, z\rangle$ and deliver $\langle v, z\rangle$ before $\langle v', z'\rangle$.

FORMULA: $\langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle \wedge \langle v', z'\rangle \in E(\Delta_{f'}) \Rightarrow \langle v, z\rangle \prec_{\Delta_{f'}} \langle v', z'\rangle$

Proof:
By assumption, we have that $\langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle$ and $\langle v', z'\rangle \in \Delta_{f'}$. By Lemma 57, we have that $\Delta_f \sqsubseteq C_{f.e}$. We then have that:

$$(\Delta_f \sqsubseteq C_{f.e}) \wedge \langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle \Rightarrow \langle v, z\rangle \prec_{C_{f.e}} \langle v', z'\rangle \tag{5.7}$$

and that:

$$(\Delta_{f'} \sqsubseteq C_{f'.e}) \wedge \langle v', z'\rangle \in E(\Delta_{f'}) \Rightarrow \langle v', z'\rangle \in E(C_{f'.e}) \tag{5.8}$$

Case $f'.e < f.e$**:** By Corollary 54 and Lemma 47, we have that:

$$C_{f'.e} \sqsubseteq C_{f.e} \tag{5.9}$$

and that:

$$\text{Eq. 5.7} \wedge \text{Eq. 5.8} \wedge \text{Eq. 5.9} \Rightarrow \langle v, z\rangle \prec_{C_{f'.e}} \langle v', z'\rangle \tag{5.10}$$

Consequently, we have that:

$$\text{Eq. 5.10} \wedge \text{Lemma 57} \wedge \langle v', z'\rangle \in \Delta_{f'} \Rightarrow \langle v, z\rangle \prec_{\Delta_{f'}} \langle v', z'\rangle \tag{5.11}$$

Case $f'.e \geq f.e$**:** By Corollary 54 and Lemma 47, we have that:

$$C_{f.e} \sqsubseteq C_{f'.e} \tag{5.12}$$

and that:

$$\text{Eq. 5.12} \wedge \text{Eq. 5.8} \wedge \text{Eq. 5.7} \Rightarrow \langle v, z\rangle \prec_{C_{f'.e}} \langle v', z'\rangle \tag{5.13}$$

Consequently, we have that:

$$\text{Eq. 5.13} \wedge \text{Lemma 57} \wedge \langle v', z'\rangle \in \Delta_{f'} \Rightarrow \langle v, z\rangle \prec_{\Delta_{f'}} \langle v', z'\rangle \tag{5.14}$$

□

Claim 33. Zab satisfies local primary order: If a primary broadcasts $\langle v, z\rangle$ before it broadcasts $\langle v', z'\rangle$, then a follower f that delivers $\langle v', z'\rangle$ must also deliver $\langle v, z\rangle$ before $\langle v', z'\rangle$.

FORMULA: $\langle v, z\rangle \prec_{\beta_e} \langle v', z'\rangle \wedge \langle v', z'\rangle \in \Delta_f \Rightarrow \langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle$

Proof:
Let f be a follower process. There are two cases to consider:

Case $f.e = e$**:** By Invariant 28, we have that:

$$D_f \sqsubseteq \beta_{f.e} \tag{5.15}$$

and by the same invariant:

$$\langle v, z\rangle \prec_{\beta_e} \langle v', z'\rangle \wedge \langle v', z'\rangle \in \Delta_f \Rightarrow \langle v, z\rangle \prec_{D_f} \langle v', z'\rangle \tag{5.16}$$

Finally we have that by Lemma 56:

$$\langle v, z\rangle \prec_{D_f} \langle v', z'\rangle \Rightarrow \langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle \tag{5.17}$$

Case $f.e > e$**:** By Lemma 56 and Invariant 28, we have that:

$$\langle v, z\rangle \prec_{\beta_e} \langle v', z'\rangle \wedge \langle v', z'\rangle \in \Delta_f \Rightarrow \langle v', z'\rangle \in E(I_{f.e}) \tag{5.18}$$

By Invariants 23 and 26:

$$\langle v, z\rangle \prec_{\beta_e} \langle v', z'\rangle \wedge \text{Eq. 5.18} \Rightarrow \langle v, z\rangle \prec_{I_{f.e}} \langle v', z'\rangle \tag{5.19}$$

Finally we have that by Lemma 56:

$$\langle v, z\rangle \prec_{I_{f.e}} \langle v', z'\rangle \wedge \langle v', z'\rangle \in \Delta_f \Rightarrow \langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle \tag{5.20}$$

□

Claim 34. Zab satisfies global primary order: Let transactions $\langle v, z\rangle$ and $\langle v', z'\rangle$ be as follows:

- A primary ρ_e broadcasts $\langle v, z\rangle$
- A primary $\rho_{e'}$, $\rho_e \prec \rho_{e'}$, broadcasts $\langle v', z'\rangle$

If a follower f delivers both $\langle v, z\rangle$ and $\langle v', z'\rangle$, then f must deliver $\langle v, z\rangle$ before $\langle v', z'\rangle$.

FORMULA: $\langle v, z\rangle \in \beta_{\rho_e} \wedge \langle v', z'\rangle \in \beta_{\rho_{e'}} \wedge \rho_e \prec \rho_{e'} \wedge \langle v, z\rangle \in \Delta_f \wedge \langle v', z'\rangle \in \Delta_f \Rightarrow \langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle$

Proof:
By Lemma 57, we have that:

$$\langle v, z\rangle \in E(\Delta_f) \Rightarrow \langle v, z\rangle \in E(C_{f.e}) \tag{5.21}$$
$$\langle v', z'\rangle \in E(\Delta_f) \Rightarrow \langle v', z'\rangle \in E(C_{f.e}) \tag{5.22}$$

Case $f.e = e'$**:** We have by Invariant 28 that $\langle v, z\rangle \notin E(D_f)$. By Lemma 56 we have that $\Delta_f = I_{f.e} \circ D_f$. Consequently, $\langle v, z\rangle \in E(I_{f.e})$ and $\langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle$.

Case $f.e > e'$**:** We have by Invariant 28 that $\langle v, z\rangle, \langle v', z'\rangle \in E(I_{f.e})$. By Invariant 27, we have that $\langle v, z\rangle \in E(\Delta_f)$ implies that $\langle v, z\rangle$ has been chosen. By Lemma 55, $\langle v, z\rangle \in I_{epoch(z')}$, and by Lemma 52, we have that $I_{epoch(z')} \sqsubseteq I_{f.e}$ and that $\langle v, z\rangle \prec_{I_{f.e}} \langle v', z'\rangle$. By Lemma 56, we conclude that $\langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle$.

□

Claim 35. Zab satisfies primary integrity: If ρ_e broadcasts $\langle v, z\rangle$ and some follower f delivers $\langle v', z'\rangle$ such that $\langle v', z'\rangle$ has been broadcast by $\rho_{e'}$, $e' < e$, then ρ_e must deliver $\langle v', z'\rangle$ before it broadcasts $\langle v, z\rangle$.

FORMULA:

$$\bigwedge \langle v, z\rangle \in E(\beta_e)$$
$$\bigwedge \exists f, e'' : f.e = e'' \wedge \langle v', z'\rangle \in E(\Delta_f)$$
$$\bigwedge \mathit{epoch}(z') < e$$
$$\Rightarrow \mathit{abdeliver}(\langle v', z'\rangle) \prec_{\rho_e} \mathit{abcast}(\langle v, z\rangle)$$

Proof:
If some follower has delivered $\langle v', z'\rangle$, then $\langle v', z'\rangle$ has been chosen, and by Lemma 55 $\langle v', z'\rangle \in E(I_{\mathit{epoch}(z)})$. Suppose by way of contradiction that ρ_e broadcasts $\langle v, z\rangle$ before it delivers $\langle v', z'\rangle$. There are two cases to consider:

- Process p_i invokes $\mathit{abcast}(\langle v, z\rangle)$ before it delivers the initial history of epoch e. This is not possible by Algorithm 1 and the Zab protocol: a primary only broadcasts a transaction if `isReady` evaluates to true and a follower only calls $\mathit{ready}(e)$ once it finishes delivering the transactions in the initial history;
- Process p_i delivers $\langle v', z'\rangle$ while in the B phase of epoch e. This action violates Invariant 23.

□

5.3. Relation to causal order

The definition of causal order is based on the *precedence* (or happens before) relation of Lamport [2], which is defined as follows:

Definition 36. (Precedence) Let ϵ and ϵ' be two events in a distributed system. The transitive relation $\epsilon \rightarrow \epsilon'$ holds if any one of the following conditions holds:

1. ϵ and ϵ' are two events on the same process and ϵ comes before ϵ';
2. ϵ is the sending of a message m by one process, and ϵ' is the receipt of m by another process;
3. There exists a third event ϵ'' such that $\epsilon \rightarrow \epsilon''$ and $\epsilon'' \rightarrow \epsilon'$.

For broadcast protocols, the events are either broadcast or deliver events. We use $\langle v, z\rangle \prec_c \langle v', z'\rangle$ to denote that $\mathit{abcast}(\langle v, z\rangle) \rightarrow \mathit{abcast}(\langle v', z'\rangle)$. The Causal Order property for atomic broadcast protocols is typically defined as:

Definition 37. (Causal order) If $\langle v, z\rangle \prec_c \langle v', z'\rangle$ and a process p delivers $\langle v', z'\rangle$, then process p must also deliver $\langle v, z\rangle$ and deliver $\langle v, z\rangle$ before $\langle v', z'\rangle$.

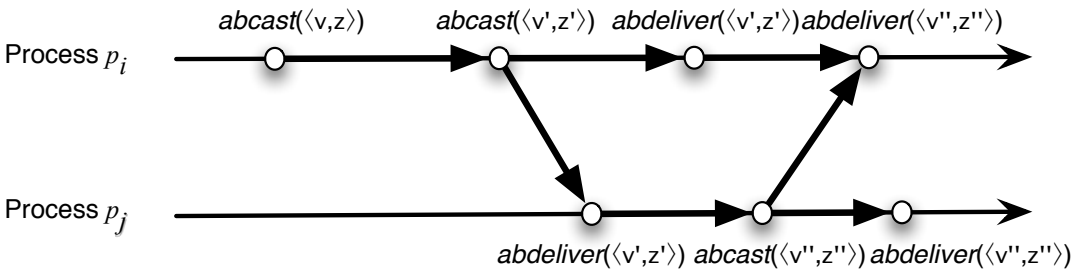

Figure 6: Counterexample of causal order.

This property, however, cannot be satisfied by Zab. Figure 6 gives an example in which two transactions are causally related, $\langle v, z\rangle$ and $\langle v'', z''\rangle$, but transaction $\langle v, z\rangle$ is not delivered. To simplify the discussion, we present only events for two processes.

PO atomic broadcast implements a causal ordering based on a weaker precedence relation.

Definition 38. (PO precedence) Let ϵ and ϵ' be two events in a distributed system. Events are either $abcast(\langle v, z\rangle)$ or $abdeliver(\langle v, z\rangle)$ events. The transitive relation $\epsilon \rightarrow_{po} \epsilon'$ holds if any one of the following conditions holds:

1. ϵ and ϵ' are local to the same process, ϵ occurs before ϵ', and at least one of the following holds: $\epsilon \neq abcast(\langle v, z\rangle)$, $\epsilon' \neq abcast(\langle v', z'\rangle)$, or $epoch(z) = epoch(z')$;
2. $\epsilon = abcast(\langle v, z\rangle)$ and $\epsilon' = abdeliver(\langle v, z\rangle)$;
3. There is a third event ϵ'' such that $\epsilon \rightarrow_{po} \epsilon''$ and $\epsilon'' \rightarrow_{po} \epsilon'$.

This weaker form of the precedence relation defines a partial order of transactions $\prec_{po}$ that is strictly weaker than $\prec_c$. We use $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$ to denote that $abcast(\langle v, z\rangle) \rightarrow_{po} abcast(\langle v', z'\rangle)$. The *PO causal order* property enforced by PO atomic broadcast is the same as causal order but considers the $\prec_{po}$ relation.

Definition 39. (PO causal order) If $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$ and a process p delivers $\langle v', z'\rangle$, then process p must also deliver $\langle v, z\rangle$ and deliver $\langle v, z\rangle$ before $\langle v', z'\rangle$.

The example of Figure 6 does not violate this variant of causal order.. PO atomic broadcast additionally enforces a property that we call *conflict freedom*.

Definition 40. (Strict causality) If a process p delivers $\langle v, z\rangle$ and $\langle v', z'\rangle$, then either $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$ or $\langle v', z'\rangle \prec_{po} \langle v, z\rangle$.

Figure 7 shows that an execution satisfying causal order atomic broadcast that may violate conflict-freedom order, thus showing that PO atomic broadcast is neither weaker nor stronger than causal order atomic broadcast.

Claim 41. *PO atomic broadcast implements PO causal order:* If $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$ and a follower f delivers $\langle v', z'\rangle$, then follower f must also deliver $\langle v, z\rangle$ and deliver $\langle v, z\rangle$ before $\langle v', z'\rangle$.

FORMULA: $\langle v, z\rangle \prec_{po} \langle v', z'\rangle \wedge \langle v', z'\rangle \in E(\Delta_f) \Rightarrow \langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle$

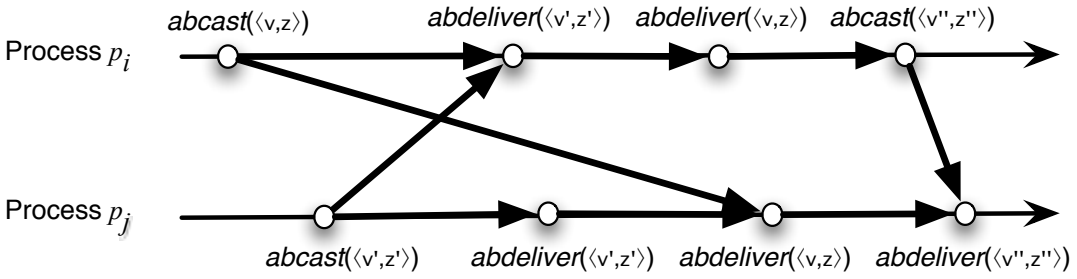

Figure 7: Example of an execution that satisfies causal order (and PO Causal Order), but not strict causality, $epoch(z) < epochz' < epoch(z'')$.

Proof:
According to PO precedence, there are three cases when $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$ holds. We show the claim for the first two base cases. The third case follows by a simple induction on the sequence of transitive causal relations $\prec_{po}$, starting from $\langle v', z'\rangle$ and proceeding backward until $\langle v, z\rangle$.

By local primary order (Claim 33), transactions broadcast by the same primary ρ_e must be delivered according to the broadcast order, so the claim holds if $epoch(z) = epoch(z')$.

We now consider the case where $epoch(z) \neq epoch(z')$. Let p_i and p_j be the processes broadcasting $\langle v, z\rangle$ and $\langle v', z'\rangle$, respectively. By the definition of $\prec_{po}$, $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$ implies that p_j delivers $\langle v, z\rangle$ before broadcasting $\langle v', z'\rangle$. Note that if $p_i = p_j$, then the last observation holds due to the relaxation of the first case of the precedence relation. By agreement (Claim 31), either f or p_j delivers both $\langle v, z\rangle$ and $\langle v', z'\rangle$.

If $epoch(z) > epoch(z')$, then we reach a contradiction. By the global primary order property (Claim 34), either f or p_j delivers $\langle v', z'\rangle$ before $\langle v, z\rangle$, a contradiction with the total order property (Claim 32) since p_j delivers $\langle v, z\rangle$ before broadcasting $\langle v', z'\rangle$ by assumption.

Consequently, we must have that $epoch(z) < epoch(z')$. By the assumption $epoch(z) < epoch(z')$ and the global primary order property, $\langle v, z\rangle$ is delivered before $\langle v', z'\rangle$ by some process. The claim follows by the total order property.
□

Claim 42. *PO atomic broadcast implements strict causality:* If a follower f delivers $\langle v, z\rangle$ and $\langle v', z'\rangle$, then $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$ or $\langle v', z'\rangle \prec_{po} \langle v, z\rangle$.

FORMULA: $\exists f : \langle v, z\rangle, \langle v', z'\rangle \in E(\Delta_f) \Rightarrow \langle v, z\rangle \prec_{po} \langle v', z'\rangle \vee \langle v', z'\rangle \prec_{po} \langle v, z\rangle$

Proof:
If $epoch(z) = epoch(z')$, then we have by the uniqueness of primaries that either $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$ if $z \prec_z z'$ or $\langle v', z'\rangle \prec_{po} \langle v, z\rangle$ if $z' \prec_z z$.

If $epoch(z) \neq epoch(z')$, then, since $\langle v', z'\rangle \in E(\Delta_f)$, we have that either $\langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle$ or $\langle v', z'\rangle \prec_{\Delta_f} \langle v, z\rangle$. Suppose without loss of generality that $\langle v, z\rangle \prec_{\Delta_f} \langle v', z'\rangle$. There are two other cases to consider:

epoch$(z) <$ ***epoch***(z')**:** By primary integrity, before broadcasting $\langle v', z'\rangle$ the primary of epoch $epoch(z')$ delivers every transaction from previous epochs that is ever delivered by any process. Consequently, $\langle v, z\rangle \prec_{po} \langle v', z'\rangle$;

epoch$(z) >$ ***epoch***(z')**:** This case cannot happen by global primary order.

□

6. Lemmata

Lemma 43. *For every follower f and epoch e, if f participates in e, then $I_e \sqsubseteq h_f$.*

> **FORMULA:** $f.a = e \Rightarrow I_e \sqsubseteq h_f$

Proof:
In Phase 2, a follower f receives a sequence T of transactions from a prospective leader of epoch e executing Phase 2 or from the established leader of e executing Phase 3. In the former case, $h_f = I_e$ once the follower processes the new leader proposal, before it completes Phase 2. In the latter case, T might contain transactions broadcast in the B phase of epoch e, and they constitute a suffix of the sequence (Invariant 20 and behavior of the leader in Phase 3). Consequently $I_e \sqsubseteq h_f$ at the end of Phase 2.

Once f proceeds to the B phase, it accepts proposals in order of zxid and adds them to its history (Invariant 23). Consequently, if $f.a = e$, then $I_e \sqsubseteq h_f$.
□

Lemma 44. *For every two followers f and f' that participate in epoch e, either $h_f \sqsubseteq h_{f'}$ or $h_{f'} \sqsubseteq h_f$.*

> **FORMULA:** $\forall f, f' : f.a = f'.a = e \Rightarrow h_f \sqsubseteq h_{f'} \vee h_{f'} \sqsubseteq h_f$

Proof:
For a follower f participating in e, $I_e \sqsubseteq h_f$ (Lemma 43). Followers accept proposals for an epoch following the order of the leader of e (Invariants 21, 23, and 26). Consequently, one must be a prefix of the other.
□

Lemma 45. *Let e, e' be epochs such that $e' < e$. If a quorum Q' of followers accept a proposal $\langle e', \langle v, z\rangle\rangle$ once a quorum Q of followers completes Phase 1 of epoch e, then at least one follower in Q is such that $\langle v, z\rangle \in E(h_f)$ at the end of Phase 1.*

> **FORMULA:**
>
> $$\bigwedge \exists Q' \subseteq \Pi : \forall f \in Q' : f \text{ has accepted } \langle e', \langle v, z\rangle\rangle$$
> $$\bigwedge \exists Q \subseteq \Pi : \forall f \in Q : f \text{ has completed Phase 1 of } e$$
> $$\bigwedge e' < e$$
> $$\Rightarrow \exists f \in Q : \langle v, z\rangle \in h_f^{e'}$$

Proof:
By the protocol, once a follower completes Phase 1 of e, it has promised not to accept proposals from the leader of earlier epochs (Invariant 25). Once a quorum of followers complete Phase 1 for epoch e, then a leader of $e' < e$ is not able to obtain a quorum of followers to accept and choose proposals of epoch e'. Consequently, at least one follower in Q must have accepted $\langle e', \langle v, z \rangle\rangle$ before completing Phase 1 of e.
□

Lemma 46. *If a quorum Q acknowledges the* NEWEPOCH(e) *message of leader ℓ and a quorum Q' acknowledges the* NEWEPOCH(e) *message of leader ℓ', then $\ell = \ell'$.*

FORMULA:

$$\begin{array}{ll} \exists & Q, Q' \subseteq \Pi : \\ & \bigwedge Q \text{ acknowledges the } \mathit{NEWEPOCH}(e) \text{ message of leader } \ell \\ & \bigwedge Q' \text{ acknowledges the } \mathit{NEWEPOCH}(e) \text{ message of leader } \ell' \\ \Rightarrow & \ell = \ell' \end{array}$$

Proof:
The claim is trivial if $Q = Q'$. Assuming that $Q \neq Q'$, suppose by way of contradiction that $\ell \neq \ell'$. A follower f only acknowledges the *NEWEPOCH*(e) from a prospective leader ℓ if $f.p < e$. Once a quorum Q acknowledges the *NEWEPOCH*(e) message of a given leader ℓ, no other quorum Q' is able to acknowledge the *NEWEPOCH*(e) message of a leader $\ell' \neq \ell$ by the intersection of quorums, otherwise some follower has acknowledged a *NEWEPOCH*(e) from two different leaders, which violates the pre-condition to accept a *NEWEPOCH*(e) message.
□

Lemma 47. *If e is an established epoch, then $C_e = I_e \circ \mathrm{CB}_e$*

Proof:

$I_e \circ \boldsymbol{CB}_e \sqsubseteq C_e$**:** No follower accepts a proposal in the B phase of epoch e before accepting the proposals in I_e (Lemma 43). Followers accept proposals of the R phase in the order of I_e (Invariant 26) and of the B phase in zxid order (Invariant 23). Consequently, the sequence of chosen proposals in epoch e must be such that the initial history I_e is a prefix of C_e and that $I_e \circ \boldsymbol{CB}_e \sqsubseteq C_e$.

$C_e \sqsubseteq I_e \circ \boldsymbol{CB}_e$**:** Suppose by way of contradiction that $C_e \not\sqsubseteq I_e \circ \boldsymbol{CB}_e$. In this case, there is at least one position i such that $C_e[i] \neq (I_e \circ \boldsymbol{CB}_e)[i]$. Consequently, a quorum of followers participating in e:

- has skipped proposals or accepted proposals of the initial history out of order (impossible by Invariant 26);
- has accepted proposals in the B phase of e violating the order of zxid (impossible by Invariant 23);
- has accepted a proposal from the leader of a different epoch (impossible by Invariant 22);

- has accepted a proposal from multiple leaders for epoch e (impossible by Lemma 46).

□

Corollary 48. For each epoch e, there is at most one initial history I_e.

FORMULA: $\neg\exists p_i, p_j \in Pi : p_i$ sends *NEWLEADER*$(e, I_e) \wedge p_j$ sends *NEWLEADER*$(e, I'_e) \wedge I_e \neq I'_e$

Proof:
By Lemma 46, if ℓ is able to complete Phase 1 of epoch e, then there is a quorum Q of followers such that for each $f \in Q$, $e \leq f.p$. Consequently, once ℓ completes Phase 1 for epoch e, no quorum of followers acknowledge a new epoch proposal for $e' \leq e$.
□

Lemma 49. $\langle v, z\rangle \in E(C_e) \Rightarrow \forall e' > e : \langle v, z\rangle \in I_{e'}$.

Proof:
The epoch $epoch(z)$ in which $\langle v, z\rangle$ is broadcast is such that $epoch(z) \leq e$. If a proposal $\langle v, z\rangle$ is chosen in epoch e, then a quorum of followers have accepted $\langle v, z\rangle$ in e. To have an operation chosen in epoch e, a quorum of followers must have current epoch e when they accept such a transaction by Invariant 22.

FORMULA:

$$\langle v, z\rangle \in E(C_e) \Rightarrow \exists Q \subseteq \Pi : \bigvee \forall f \in Q : f \text{ has sent } \textit{ACK}(e, \langle v, z\rangle)$$
$$\bigvee \forall f \in Q : p_i \text{ has sent } \textit{ACK-LD}(e)$$

Suppose now that a new prospective leader ℓ attempts to establish a new epoch $e' > e$ (running Phase 1 for e'). There are three cases to consider:

- The latest current epoch ℓ receives in Phase 1 from a quorum Q is e. By Lemma 43, every follower f, $f.a = e$, is such that $I_e \sqsubseteq h_f$, and there is exactly one such a history by Corollary 48. If $\langle v, z\rangle \in E(C_e) \wedge \langle v, z\rangle \in E(I_e)$, then the claim follows by Lemma 43. If $\langle v, z\rangle \in E(CB_e)$, then by Invariant 23 all followers accept proposals in order and by the intersection of quorums, the quorum Q' that chooses an operation in e and Q must have at least one follower f in common. By Invariant 25, a follower that completes Phase 1 of e' does not accept proposals from an earlier epoch, and by Lemma 45, there is no transaction $\langle v, z\rangle$ chosen in epoch e such that $\langle v, z\rangle \notin E(h_f)$, for some follower $f \in Q \cap Q'$. Finally, the initial history $I_{e'}$ is by the protocol the history of a follower f after completing Phase 1 of e' such that $f.a = e$ and it has the highest $f.zxid$ among all followers of Q. We conclude that $\langle v, z\rangle \in E(I_{e'})$

  FORMULA: $\exists f \in Q : f.a = e \wedge I_{e'} = h_f^e \Rightarrow \langle v, z\rangle \in E(h_f)$

  Note that if a follower f participates in establishing a new epoch e'', $e < e'' < e'$, but $f.a = e$, then its history does not change (Invariant 24). If a follower does change its history, then it changes its current epoch to e'' according to the protocol;

FORMULA: (Follower f sends *ACK-E*$(e'', h_f)) \wedge (f.a = e) \Rightarrow h_f = h_f^e$

- The latest current epoch ℓ receives in Phase 1 from a quorum Q is e'', $e' > e'' > e$. We show by induction that a follower participating in epoch e'' must contain $\langle v, z \rangle$ in its history:

  Base case: Let e'' be the first epoch later than e such that some follower f sets current epoch to e''. If f sets $f.a \leftarrow e''$, then a quorum of followers has promised not to accept proposals from a leader of an epoch earlier than e''. Moreover, the initial history of e'' must contain $\langle v, z \rangle$, since:

  - it must contain chosen proposals of e: if some follower f has set $f.a \leftarrow e''$, then a quorum Q' of followers must have promised not to accept proposals from earlier epochs in Phase 1. If $\langle v, z \rangle$ is chosen in e, then there must be some follower f' in Q' such that $\langle v, z \rangle \in E(h_{f'})$ by the intersection of quorums and the assumption that a quorum of followers has acknowledged the *NEWEPOCH*(e'') in Phase 1 of e''. Moreover, since by assumption e'' is the first epoch later than e such that some follower f sets $f.a \leftarrow e''$, then the latest current epoch among the followers of Q must be e, and the follower f in Q with current epoch e and highest zxid must contain $\langle v, z \rangle$ in its history by Invariants 23 and 26;

    FORMULA: $\exists f \in Q : \langle v, z \rangle \in E(h_f)$

  - A follower atomically changes its current epoch and accepts the initial history when executing the R phase for a given epoch (Invariant 24). Consequently, if there has been any attempt to establish an epoch later than e and earlier than e'', but a follower has not changed its current epoch, then it does not change its history.

    FORMULA: (Follower f sends *ACK-E*$(e''', h_f)) \wedge (f.a = e) \Rightarrow h_f = h_f^e$

  Induction hypothesis and step: Suppose the claim holds for $e'' - 1$ and earlier epochs later than e, and show for e''. Let Q be a quorum of followers that send a current epoch message to the prospective leader in Phase 1 of e'' and f be a follower with the highest epoch and highest zxid. By assumption that $\langle v, z \rangle \in E(C_e)$, we have that $f.a \geq e$. By the protocol, the history of h_f is selected as the initial history of $I_{e''}$. By the induction hypothesis and the assumption that no transaction is lost or reordered during recovery (Invariant 26), $\langle v, z \rangle \in E(h_f)$ and consequently $\langle v, z \rangle \in I_{e''}$. By Lemma 43, a follower participating in e'' must contain $\langle v, z \rangle$ in its history.

  Back to the claim, there is by assumption a follower f with current epoch e'' that sent a *CEPOCH*(e'') message during Phase 1 of e'. Also by assumption, e'' is the latest epoch among followers in Q' and $f.zxid$ is the highest among the followers with current epoch e'' in Q. By the previous argument, $\langle v, z \rangle \in E(h_f)$.

- The latest current epoch leader ℓ receives in Phase 1 from a quorum Q is e'', $e'' < e$. By assumption a quorum of followers have accepted a proposal in epoch e. Such followers can only accept proposals in epoch e if their current epoch is e, either by Invariant 24 or by the construction of the protocol (a follower only accepts a proposal in the B phase of e if its current epoch is e). By the protocol, a follower does not go back to a previous epoch, and by the intersection of quorums Q must contain at least one follower with epoch at least e.

□

Corollary 50. In Phase 1 of epoch e, the history h_f of the follower f with the latest $f.a$ and highest $f.zxid$ contains all proposals chosen in epoch e', for all $e' < e$.

Proof sketch:
By Lemma 49.
□

Lemma 51. *If the history of a follower participating in epoch e is such that $\langle v, z\rangle, \langle v', z'\rangle \in E(h_f)$ and $\langle v, z\rangle, \langle v', z'\rangle \in E(C_e)$ and there is no $\langle v'', z''\rangle \in E(h_f)$ such that $\langle v, z\rangle \neq \langle v'', z''\rangle$, $\langle v', z'\rangle \neq \langle v'', z''\rangle$, and $\langle v, z\rangle \prec_{h_f} \langle v'', z''\rangle \prec_{h_f} \langle v', z'\rangle$, then there is no follower participating in epoch $e' > e$ such that $\langle v, z\rangle, \langle v', z'\rangle \in E(h_{f'})$ and there is $\langle v'', z''\rangle \in E(h_{f'})$ such that $\langle v, z\rangle \neq \langle v'', z''\rangle$, $\langle v', z'\rangle \neq \langle v'', z''\rangle$, and $\langle v, z\rangle \prec_{h_{f'}} \langle v'', z''\rangle \prec_{h_{f'}} \langle v', z'\rangle$.*

FORMULA:

$$
\begin{array}{ll}
\exists & f \in \Pi : \\
\bigwedge & f.a = e \\
\bigwedge & \langle v, z\rangle, \langle v', z'\rangle \in E(h_f) \\
\bigwedge & \langle v, z\rangle, \langle v', z'\rangle \in E(C_e) \\
\bigwedge & \neg\exists \langle v'', z''\rangle \in E(h_f) : \langle v, z\rangle \neq \langle v'', z''\rangle \wedge \\
 & \wedge \langle v', z'\rangle \neq \langle v'', z''\rangle \wedge \langle v, z\rangle \prec_{h_f} \langle v'', z''\rangle \prec_{h_f} \langle v', z'\rangle \\
\Rightarrow & \\
\neg\exists & f' \in \Pi : \\
\bigwedge & f'.a = e', e' > e \\
\bigwedge & \langle v, z\rangle, \langle v', z'\rangle \in E(h_{f'}) \\
\bigwedge & \exists \langle v'', z''\rangle \in E(h_{f'}) : \langle v, z\rangle \neq \langle v'', z''\rangle \wedge \\
 & \langle v', z'\rangle \neq \langle v'', z''\rangle \wedge \langle v, z\rangle \prec_{h_{f'}} \langle v'', z''\rangle \prec_{h_{f'}} \langle v', z'\rangle
\end{array}
$$

Proof:
Suppose by way of contradiction that such f', $\langle v'', z''\rangle$, and e' exist. There are two cases to consider:

- $epoch(z) = epoch(z')$. This case can't happen by the protocol: proposals of the same epoch are processed in order (Invariant 23) and there can't be multiple established leaders for the same epoch so that proposals of different leaders interleave (Lemma 46);
- $epoch(z) \neq epoch(z')$. There are two sub-cases to consider:
  - $epoch(z) \leq epoch(z'') < epoch(z')$: If $\langle v', z'\rangle \in E(C_e)$, then e is an established epoch. We prove with a simple induction argument over epoch numbers that $\langle v'', z''\rangle \notin I_{e'}$ for any epoch number $e' > e$ such that $f'.a = e'$ for some follower f'.

    Base case: Let e' be the first epoch greater than $epoch(z')$ such that some follower f' makes $f'.a = e'$. The initial history $I_{e'}$ must be such that the epoch of the highest zxid of $I_{e'}$ is e by the intersection of quorums and the assumption that

there is no established epoch e'' such that $e < e'' < e'$. By Lemma 44, the assumption that $\langle v', z' \rangle \in E(C_e)$, and the assumption that $\langle v'', z'' \rangle \prec_{h_{f'}} \langle v', z' \rangle$, there is no follower f'' that participated in e and $\langle v'', z'' \rangle \in h_{f''}$. Consequently, $\langle v'', z'' \rangle \notin I_{e'}$.

Induction hypothesis: If a follower f' is such that $f'.a = e', e' > e$, then $\langle v'', z'' \rangle \notin I_{e'}$.

Induction step: Let the claim hold for all $e'' > e$. We prove it for $e' > e''$. We have by the algorithm that $I_{e'} = h_{f''}$, for some f'' (Invariant 26). If $f''.a = e$, then it follows from the base case. Otherwise, we have that $\langle v'', z'' \rangle \notin I_{f''.a}$ by the induction hypothesis. By Invariant 22, f'' does not accept proposals from an epoch other than $f''.a$, and by the algorithm, the leader $\ell_{e''}$ in the B phase proposes transactions such that the zxid epoch is e''. Consequently, f'' does not accept $\langle v'', z'' \rangle$ in epoch $f''.a$, and we have that $\langle v'', z'' \rangle \notin I_{e'}$.

– $epoch(z) < epoch(z'') = epoch(z')$: There cannot be such a z'', otherwise some follower has violated Invariant 23.

□

Lemma 52. *Let e, e' be epochs, $e < e'$, and e be an established epoch. $I_e \sqsubseteq I_{e'}$.*

FORMULA:

$$\begin{array}{ll} \bigwedge & e \text{ is an established epoch} \\ \bigwedge & e < e' \\ \Rightarrow & I_e \sqsubseteq I_{e'} \end{array}$$

Proof:
By Lemma 49:

$$\langle v, z \rangle \in E(I_e) \Rightarrow \langle v, z \rangle \in E(I_{e'})$$

By Invariants 23, 26:

$$\begin{array}{ll} & (\langle v, z \rangle, \langle v', z' \rangle \in E(I_e)) \wedge (\langle v, z \rangle \prec_{I_e} \langle v', z' \rangle) \\ \Rightarrow & (\langle v, z \rangle, \langle v', z' \rangle \in E(I_{e'})) \wedge (\langle v, z \rangle \prec_{I_{e'}} \langle v', z' \rangle) \end{array}$$

By Lemma 49 and Invariant 26 we also have that $I_e[1] = I_{e'}[1]$. Finally, by Lemma 51 we have that $I_e \sqsubseteq I_{e'}$.

□

Lemma 53. *Let e, e' be epochs such that $e < e'$, e is established, and there is no established epoch e'', $e < e'' < e'$. $C_e \sqsubseteq I_{e'}$.*

FORMULA:

$$\begin{array}{ll} & e \text{ is an established epoch} \\ \bigwedge & \\ \bigwedge & \forall e'', e < e'' < e' : e'' \text{ is not established} \\ \Rightarrow & C_e \sqsubseteq I_{e'} \end{array}$$

Proof:
By Lemma 47, $C_e = I_e \circ CB_e$. By Lemma 52, $I_e \sqsubseteq I_{e'}$ and by Lemma 49 a transaction chosen in e is in the initial history of later epochs. We prove by induction on the length k that $I_e \cdot \langle v_1, z_1\rangle \cdot \langle v_1, z_2\rangle \cdots \langle v, z_k\rangle \sqsubseteq I_{e'}$, where all proposals $\langle e, \langle v_i, z_i\rangle\rangle$ have been chosen during the B phase of epoch e, $1 \leq i \leq k$, $epoch(z_i) = e$, and $z_i \prec_z z_j$ if $i < j$.

Base case: $k = 1$. If $\langle v_1, z_1\rangle$ has been chosen, then by Lemma 45, there is a follower f in Phase 1 of e' such that $f.p = e'$, $f.a = e$ and $\langle v_1, z_1\rangle \in E(h_f^e)$. By the protocol (choice of initial history of e' and Invariant 23), $I_e \cdot \langle v_1, z_1\rangle \sqsubseteq I_{e'}$.

Induction hypothesis: Claim holds for k.

Induction step: Assume that $I_e \cdot \langle v_1, z_1\rangle \cdot \langle v_2, z_2\rangle \cdots \langle v_k, z_k\rangle \sqsubseteq I_{e'}$ and show for $k+1$. If $\langle v_{k+1}, z_{k+1}\rangle$ has been chosen, then a quorum of followers participating in e has accepted $\langle v_{k+1}, z_{k+1}\rangle$. By assumption, we have that $I_e \cdot \langle v_1, z_1\rangle \cdot \langle v_2, z_2\rangle \cdots \langle v_k, z_k\rangle \sqsubseteq I_{e'}$. By the protocol, a follower that accepts $\langle v_{k+1}, z_{k+1}\rangle$ must also have accepted all $\langle v_i, z_i\rangle$, $i \leq k$. By the protocol (choice of initial history), $I_e \cdot \langle v_1, z_1\rangle \cdot \langle v_2, z_2\rangle \cdots \langle v_{k+1}, z_{k+1}\rangle \sqsubseteq I_{e'}$.

By the definition of initial history, $I_{e'}$ is the history of the prospective leader of e' selects for epoch e' at the end of Phase 1.
□

Corollary 54. Let e, e' be epochs such that $e < e'$, e is established. $C_e \sqsubseteq I_{e'}$.

Proof:
By Lemmas 52 and 53, and a simple induction on the epoch numbers.
□

Lemma 55. *If e is an established epoch and $\langle v, z\rangle \in E(C_{e'})$ for some e', epoch$(z) < e$, then $\langle v, z\rangle \in E(I_e)$.*

FORMULA: e is an established epoch $\wedge \exists e' : \langle v, z\rangle \in E(C_{e'}) \wedge epoch(z) < e \Rightarrow \langle v, z\rangle \in E(I_e)$

Proof:
There are three cases to consider:

$e' < e$: If $\langle v, z\rangle$ has been chosen in an epoch earlier than e, then $\langle v, z\rangle \in E(I_e)$ by Corollary 54;

$e' = e$: We have that $\langle v, z\rangle \in E(I_e)$ by Invariant 22;

$e < e'$: Suppose by way of contradiction that $\langle v, z\rangle \notin E(I_e)$. By the protocol, we have that:

$$\exists f, e'' : h_f^{e''} = I_{e'} \wedge \langle v, z\rangle \in E(h_f^{e''}) \tag{6.1}$$

By Equation 6.1 and Corollary 48, we have that:

$$\exists e'': \quad \bigwedge e' \geq e'' > e \qquad (6.2)$$
$$\bigwedge \langle v, z\rangle \in E(I_{e''})$$
$$\bigwedge \exists f, e''' : I_{e''} = h_f^{e'''} \wedge e''' < e$$

By the protocol, the initial history is provided by a follower with latest current epoch among a quorum of followers. By the intersection of quorums and the assumption that e'' is later than e, we have that Equation 6.2 cannot hold.

□

Lemma 56. *For every follower $f \in \Pi$, $\Delta_f = I_{f.e} \circ D_f$.*

Proof:
When a follower f learns that a new epoch e has been established by receiving a commit message from the established leader of e, it delivers the transactions in the initial history I_e. Once it moves to Phase 3, the B phase, the follower delivers transactions following the order of β_e (Invariant 28). Consequently, $\Delta_f = I_{f.e} \circ D_f$.
□

Lemma 57. *For every follower $f \in \Pi$, $\Delta_f \sqsubseteq C_{f.e}$*

Proof:
By Invariant 28, we have that:

$$D_f \sqsubseteq \beta_{f.e} \qquad (6.3)$$

By Invariant 21, Invariant 23, and the definition of chosen:

$$CB_e \sqsubseteq \beta_e \qquad (6.4)$$

By Lemma 47 and Lemma 56, Δ_f and $C_{f.e}$ have $I_{f.e}$ as a common prefix. It remains to show that $D_f \sqsubseteq CB_{f.e}$. We know by Invariant 28 and by Equation 6.4 that either $D_f \sqsubseteq CB_{f.e}$ or $CB_{f.e} \sqsubseteq D_f$. By Invariant 27, we have that $D_f \sqsubseteq CB_{f.e}$.
□

7. Fallacies

Fallacy 58. A quorum Q of followers during Phase 1 of epoch e is such that there is $f \in Q$ such that for all $f' \in Q$, $h_{f'} \sqsubseteq h_f$.

Counterexample:
We construct an execution as follows:

- A quorum Q of followers does not crash in this execution;
- A follower f participates in epoch e'', and all other followers have participated in epoch e'' and currently participate in epoch $e' > e''$;
- In the B phase of e'', follower f has accepted proposals with zxid up to z, whereas all other followers have accepted up to $z' \prec_z z$;

- All followers in $Q \setminus \{f\}$ accept a proposal in the B phase of e';
- All followers in Q execute the phase 1 of $e > e'$, and there is no follower f' such that $h_{f'}$ is a prefix of all other followers in the phase 1 of e.

□

Fallacy 59. Once a quorum of followers Q completes Phase 1, no transaction can be chosen in previous epochs.

Counterexample:
Messages from a previous leader might still reach a minority of followers. If there are enough followers in Q that have accepted a given proposal, then along with followers in the minority that leader is still able to reach, we have a quorum and chosen proposals in a previous epoch.
□

Fallacy 60. Let e be an epoch. Once a quorum of followers completes Phase 1 for e, no other $e' < e$ becomes established.

Counterexample:
An epoch e becomes established once a quorum of followers acknowledge the *NEWLEADER*(e, I_e) proposal. Say that each follower in a quorum Q' acknowledges the *NEWLEADER*$(e', I_{e'})$. Assuming that $\Pi \setminus \{f\}$, $f \in Q'$, contains a quorum Q, it is straightforward to construct an execution such that f acknowledges the *NEWLEADER*$(e', I_{e'})$ after each follower in $Q' \setminus \{f\}$ has acknowledged *NEWLEADER*$(e', I_{e'})$ and each follower in Q has acknowledged the *NEWLEADER*(e, I_e) proposal.
□

References

[1] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition)*, chapter 7. John Wiley Interscience, March 2004.

[2] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21:558–565, July 1978.

[3] Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.