Zookeeper Atomic Broadcast

Emmanouil (Manos) Giortamis
Chair of Decentralized Systems Engineering (DSE)
Department of Computer Science
https://dse.in.tum.de/

WS 2022/23

ZAB: High-performance broadcast for primary-backup systems

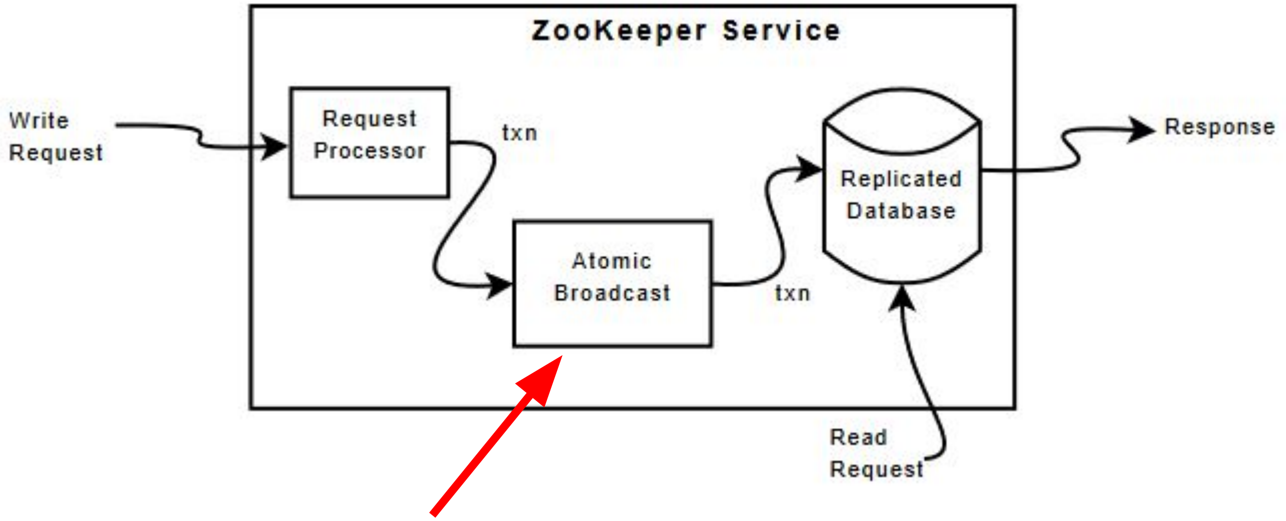
The problem

- Zookeeper uses a primary-backup scheme for replica consistency
- Primary nodes apply *incremental* and *idempotent* changes
- Application order must follow delivery order for state changes
  - “FIFO” order
  - *At-least-once* semantics
- Primary nodes can crash
  - Half-completed broadcasts
  - Fast and consistent recovery needed

Existing solutions did not satisfy the above requirements (atomic, custom ordering, fast recovery)

Solution: ZAB

- A crash-recovery Atomic Broadcast protocol
- Guarantees *at-least-once* semantics
- Implements custom “*Primary order*” ordering



Atomic broadcast

- Useful primitive with many applications in distributed systems
- Atomic: either completes as a whole, or not at all
- Broadcast: transfer of messages to all recipients simultaneously
- Ordering: how messages from different broadcasts are ordered
- Various ordering semantics:
  - Total order
  - FIFO order
  - Causal order
  - …

ZAB model

- Primary-backup replica roles
- Any replica can act as a primary
  - At most one active at a time
- Works with epochs
  - That change with each new primary
- State changes are called transactions (TXs)
  - Invoked only by a primary
- TXs are identified by an epoch and a counter pair (*zxid*)
  - In some epoch a new TX increments the counter
- Ordering as follows:
  - *zxid1* < *zxid2* iff:
    - *zxid1*.epoch < *zxid2*.epoch
    - *zxid1*.epoch = *zxid2*.epoch and *zxid1*.counter < *zxid2*.counter

ZAB model (cntd)

- Uses TCP that preserves FIFO order
- (Safety) Properties:
  - Integrity: a process receives a message iff it was sent by some process
  - Agreement: any two processes deliver the same messages
  - Primary Order (PO):
    - If a primary broadcasts m before m’, then a recipient delivers m before m’
    - If two primaries broadcast two messages in different epochs, a recipient delivers them in the epoch order
  - Primary Integrity: a primary broadcasts iff it has delivered the TXs of previous epochs

Protocol overview

- Two roles: *leader* and *follower*
- ZAB assumes a leader election to happen first
- All replicas know which is the new leader via an *oracle*
- It consists of three phases:
  - Discovery
  - Synchronization
  - Broadcast
- A leader is not “officially” leading until the synchronization phase completes
- A leader executes the steps of the follower as well
- A leader receives all changes from previous leaders (epochs) before broadcasting their own

Discovery phase

- Follows after a leader election algorithm
- *Q*: quorum

Leader *l*	Follower
	Send last epoch number *e* to *l*
Upon receiving messages from *Q*, propose new epoch *e'* s.t. *e'* > *e* for all messages received in *Q*	
	Upon receiving *e'*, if *e'* > *e*, then *e* = *e'*. Reply with *ACK* with the highest *zxid*
Upon receiving *ACK*s from *Q*, select highest *zxid.* Receive missing TXs from followers (s.t. *zxid'* < *zxid*)	

Synchronization phase

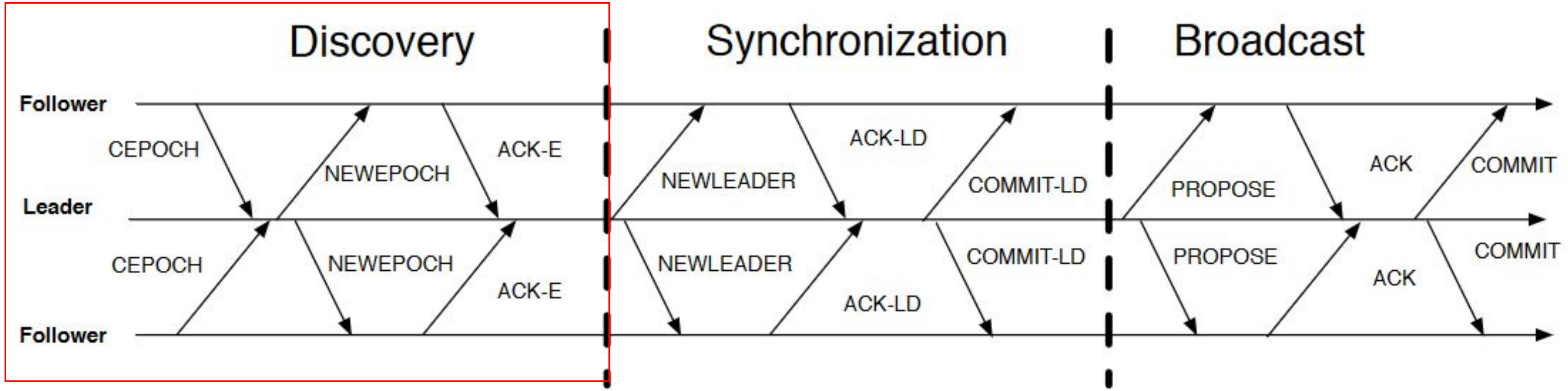
Leader *l*	Follower
Propose *NEWLEADER* and send highest *zxid* selected.	
	Set *l* as the new leader. Accept TXs with zxid’ < *zxid.* Reply with *ACK*
Upon receiving *ACK*s from *Q*, send *COMMIT* message to all followers	
	Upon receiving *COMMIT*, deliver the previously accepted TXs

Broadcast phase

Leader	Follower
Increment counter and propose TXs Now *zxid'* > previous largest *zxid*	
	Accept the TXs and reply with *ACK*
Upon receiving *ACK*s from *Q*, send *COMMIT* to all followers	
	Upon receiving *COMMIT*, deliver the TXs

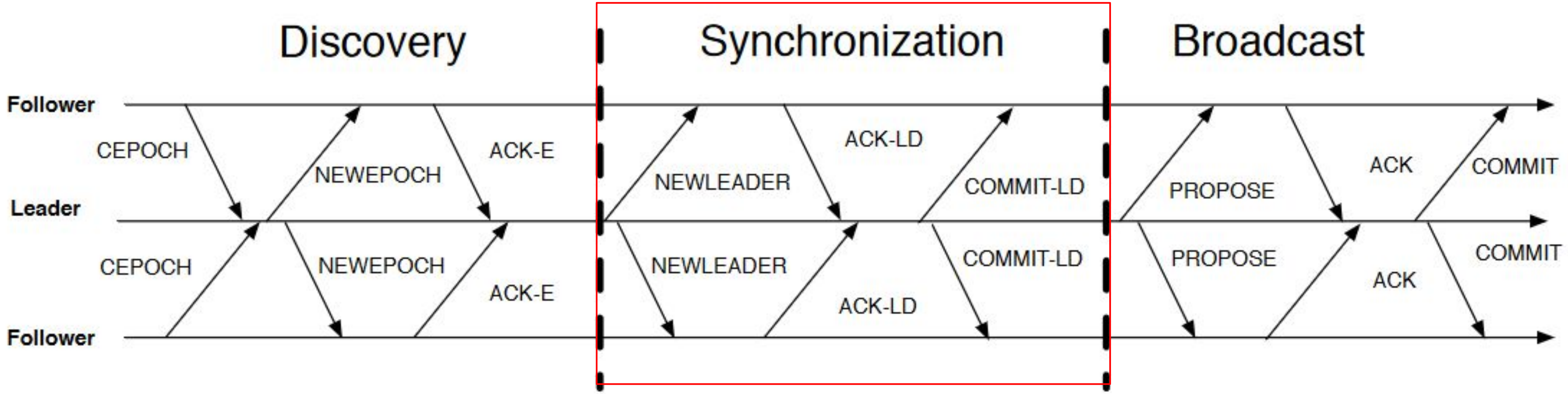
Complete picture

- CEPOCH: last epoch number
- NEWEPOCH: new epoch proposal
- ACK-E: acknowledgement of epoch proposal



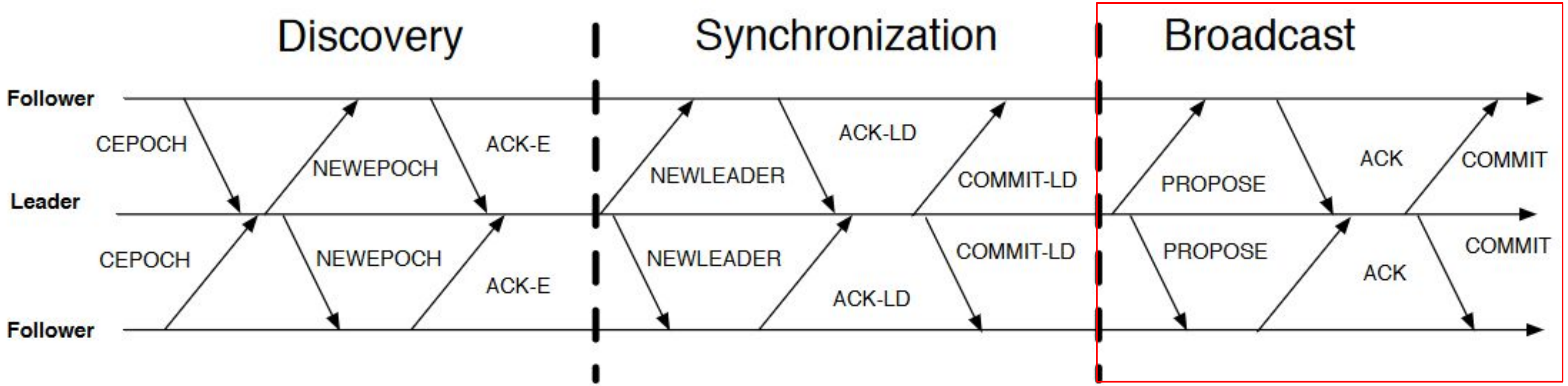
Complete picture

- NEWLEADER: propose self as new leader
- ACK-LD: acknowledgement of new leader proposal
- COMMIT-LD: commit leader proposal



Complete picture

- PROPOSE: propose new TX
- ACK: acknowledgement of leader's proposal
- COMMIT: leader commits proposal



Failure detection

- Leaders and followers use heartbeat messages
- A leader:
  - Receives heartbeats from Q
  - If not enough heartbeats, steps down
  - Triggers leader election
  - Moves to phase 1 (Discovery)
- A follower:
  - Follows a leader as long as heartbeat messages are received
  - If not, it abandons them
  - Triggers leader election
  - Moves to phase 1 (Discovery)

References

- ZAB: https://marcoserafini.github.io/papers/zab.pdf
- Zookeeper: https://www.usenix.org/legacy/event/atc10/tech/full_papers/Hunt.pdf