# ZABViz - A Visualization tool for Zookeeper Atomic Broadcast Protocol

Rhythm Girdhar and Omkar Kulkarni

*Abstract*— **This research introduces ZABViz, a sophisticated visualization tool designed to enhance the understanding and exploration of the Zookeeper Atomic Broadcast Protocol (ZAB) — a critical consensus algorithm widely employed in distributed systems such as Apache ZooKeeper, HDFS, and Kafka. Recognizing the intricate nature of ZAB and the challenges associated with its comprehension, we conducted a comprehensive study, delving into scholarly papers, instructional videos, and engaging in extensive discussions with domain experts.**

**Motivated by the necessity to simplify complexities of ZAB and elevate the learning experience, we conceptualized ZABViz. In contrast to static representations, ZABViz provides a visual journey through ZAB's core functionalities, including node communication, log replication, and leader election. The significance of ZABViz lies in its potential to offer researchers, developers, and educators an intuitive platform for comprehending the nuances of ZAB's processes.**

**This report provides an in-depth exploration of ZABViz, spanning its motivational underpinnings, unique features, and the profound implications it holds for academic understanding and real-world applications in the domain of distributed systems.**

## I. Introduction

In distributed systems, the Zookeeper Atomic Broadcast Protocol (ZAB) plays a crucial role, ensuring consensus and reliability in important systems such as Apache ZooKeeper, HDFS, and Kafka. The importance of ZAB in these distributed environments cannot be overstated, with its role in maintaining robustness and scalability.

Apache ZooKeeper[1], a fault-tolerant distributed coordination service, stands as a testament to ZAB's significance. ZooKeeper encapsulates distributed coordination algorithms to provide essential services such as bootstrapping, configuration data storage, process status tracking, group membership management, synchronization primitives implementation, and failure recovery orchestration [2]. The ensemble nature of ZooKeeper, with total database replication performed across a cluster of host servers, ensures high availability and reliability. ZAB, as the backbone of ZooKeeper, governs atomic updates to replicas, manages update transactions, and orchestrates recovery from crashed states to valid ones [5], [6].

Similarly, Hadoop Distributed File System (HDFS), pivotal for distributed storage and processing, relies on ZAB to maintain consistency and fault tolerance[3]. Kafka, a distributed streaming platform, employs ZAB [4]for efficient leader election, synchronization, and replication of log entries across broker nodes.

ZAB's prominence arises not only from its foundational role within these systems but also from its ability to achieve consensus efficiently, especially through the application of asynchronous log replication. Unlike Raft, another consensus protocol, ZAB's design enables the leader node to commit log entries without waiting for acknowledgments from all other nodes, leading to higher throughput.

Motivated to simplify ZAB's complexities and provide a powerful educational resource, we designed ZABViz. Unlike previous efforts that primarily offered static diagrams or written explanation of the protocol, ZABViz transforms the learning experience by offering an dynamic visualization of ZAB's critical functionalities.

The significance of ZABViz is multi-faceted. For system developers, it streamlines the debugging process and aids in devising optimized configuration strategies for ZAB-based systems. Educators, on the other hand, gain a robust tool to elevate students' understanding of distributed systems, fostering an environment conducive to effective learning and experimentation.

This report thoroughly explores ZABViz, covering its design principles, key features, and how it's implemented. It includes a detailed case study showcasing ZABViz in action, incorporating valuable insights from user feedback and innovative solutions to development challenges. A comparison with traditional understanding highlights the unique benefits of ZABViz. The research outlines the tool's potential implications, offering a comprehensive view of its motivations, design, features, implementation details, real-world applications, user feedback, challenges, and future scope.

## II. ZooKeeper Atomic Broadcast (ZAB)

### A. Atomic Broadcast

A broadcast algorithm transmits a message from one process – the primary process – to all other processes in a network or in a broadcast domain, including the primary. Atomic broadcast is a primitive in distributed computing, ensuring either correct message broadcast or a safe abortion without side effects [7]. It plays a vital role in group communication and is synonymous with reliable broadcast satisfying total order properties.

Properties of Atomic Broadcast:

- Validity: If a correct process broadcasts a message, all correct processes will eventually deliver it.
- Uniform Agreement: If a process delivers a message, all correct processes eventually deliver that message.
- Uniform Integrity: Every process delivers a message at most once, and only if it was previously broadcast by the sender.
- Uniform Total Order:If processes deliver messages m and m', then the order of delivery by one process is the same as another.

### B. Choosing ZAB over Paxos and its Design Decisions

ZooKeeper considered Paxos [8] but opted for ZAB due to specific shortcomings in Paxos that were critical for ZooKeeper's requirements [9]:

1) Handling Multiple Outstanding Client Operations:
   - Paxos Limitation: Paxos did not inherently support multiple outstanding transactions efficiently.
   - ZAB Solution: ZAB introduced a transaction identification scheme, enabling the handling of multiple outstanding client operations more effectively.
2) Efficient Recovery from Crashes:
   - Paxos Challenge: In Paxos, the manipulation of the sequence of transactions during recovery from primary crashes was deemed inefficient.
   - ZAB Improvement: ZAB addressed this issue by employing a transaction identification scheme that significantly enhanced the efficiency of the recovery process.
3) Performance Considerations:
   - Paxos Trade-offs: Paxos, while robust, faced challenges in meeting ZooKeeper's performance requirements, including low latency, good throughput under bursty conditions, and smooth failure handling.
   - ZAB Alignment: ZAB was designed with a focus on meeting these performance criteria, ensuring optimal system responsiveness and reliability.

### C. ZAB Properties

ZooKeeper Atomic Broadcast (ZAB) is designed to operate within a crash-recovery model for Zookeeper. The properties inherent in ZAB ensure the consistency, reliability, and efficiency of the system [10].

1) Consistency and Integrity:
   - Integrity Assurance: ZAB maintains the integrity of communication channels by ensuring that a process receives a message only if it was sent by another process (integrity property).
   - Consistency Guarantee: Every delivered state change ($\langle$v, z$\rangle$) in ZAB corresponds to a prior broadcast by the primary process, ensuring the consistency of the distributed ensemble.
2) Total Order and Agreement:
   - Total Order Principle: ZAB enforces a total order on delivered state changes, such that if process A delivers $\langle v, z\rangle$ before $\langle v', z'\rangle$, any process delivering $\langle v', z'\rangle$ must have previously delivered $\langle v, z\rangle$.
   - Agreement among Processes: When different processes ($p_i$ and $p_j$) deliver distinct state changes $\langle v, z\rangle$ and $\langle v', z'\rangle$, the system ensures that either $p_i$ delivers $\langle v', z'\rangle$ or $p_j$ delivers $\langle v, z\rangle$, establishing agreement.
3) Primary Order Properties:
   - Local Primary Order: Within ZAB, local primary order ensures that if a primary broadcasts $\langle v, z\rangle$ before $\langle v', z'\rangle$, any process delivering $\langle v', z'\rangle$ must have delivered $\langle v, z\rangle$ before $\langle v', z'\rangle$.
   - Global Primary Order: ZAB maintains global primary order by dictating that if a primary $\rho_i$ broadcasts $\langle v, z\rangle$ and a subsequent primary $\rho_j \succ \rho_i$ broadcasts $\langle v', z'\rangle$, any process delivering both $\langle v, z\rangle$ and $\langle v', z'\rangle$ must deliver $\langle v, z\rangle$ before $\langle v', z'\rangle$.
4) Primary Integrity:
   - Ensuring Historical Consistency: ZAB guarantees primary integrity, ensuring that if a primary $\rho_e$ broadcasts $\langle v, z\rangle$ and another process delivers a state change $\langle v', z'\rangle$ previously broadcast by $\rho'_e \prec \rho_e$, then $\rho_e$ must have delivered $\langle v', z'\rangle$ before broadcasting $\langle v, z\rangle$. This ensures historical consistency across primary processes and their broadcasts.

## III. Implementation of ZAB and ZABViz

### A. Overview

In the ZAB protocol, the coordination within a cluster is orchestrated through the roles of a leader and followers. The leader accepts state changes from clients and replicates them to itself and followers. Clients can read from any ZooKeeper node [11] and submit write requests, which are forwarded to the leader [12]. Transactions, representing client state changes, are replicated to followers to maintain consensus.

### B. Key Definitions

- Leader and Followers: One node in the ZooKeeper cluster acts as the leader, while the others assume follower roles. The leader is responsible for accepting and replicating state changes, while read requests are load-balanced among followers.
- Transactions: Represent client state changes propagated by the leader to followers.
- Epoch (e) and Counter (c): Used for ordering transactions and ensuring consistency across the cluster.
- F.history: Follower's history queue used for committing incoming transactions in the order they arrived.
- Outstanding Transactions: Set of transactions in F.history with a sequence number smaller than the current COMMIT sequence number.

### C. ZABViz Integration

ZABViz introduces a visual layer to the ZAB protocol, offering a real-time and intuitive representation of its execution. This visualization tool simplifies the understanding of complex consensus algorithms through dynamic visualizations of state transitions.

ZABViz utilizes a variation of the two-phase-commit protocol for replicating transactions to followers. It allows users to read from any ZooKeeper node and submit write requests, forwarding state changes to the leader. The tool captures the essence of the ZAB protocol by visualizing transactions, leader elections, and state transitions.

ZABViz mirrors the ZAB protocol's phases [13], [14]:


Fig. 1. Legend for all states

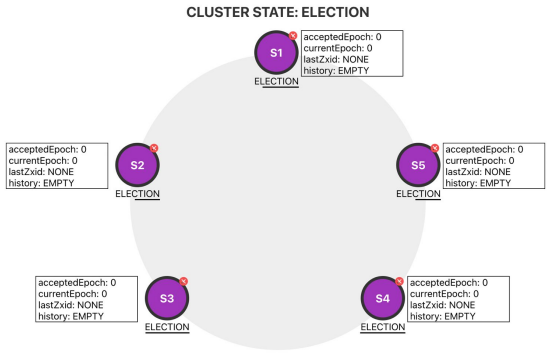


Fig. 2. Phase 0: Leader Election

1) **Leader Election:** As in Fig 2, Prospective leader election to initiate the ZAB protocol.
2) **Discovery:** As in Fig 3, Followers communicate with the prospective leader to gather information on recent transactions.
3) **Synchronization:** The recovery phase, synchronizing replicas in the cluster using the leader's updated history.
4) **Broadcast:** As in Fig 4 and Fig 5, Peers stay in this phase indefinitely, performing broadcasts of transactions [15].

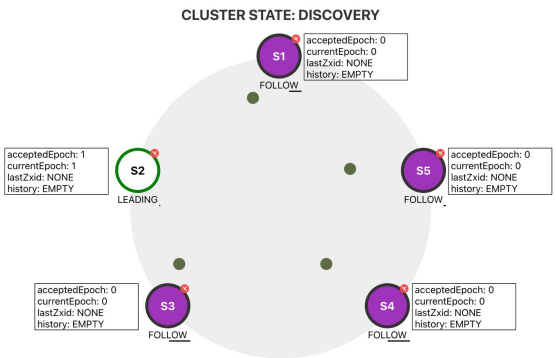


Fig. 3. Phase 1: Leader Election

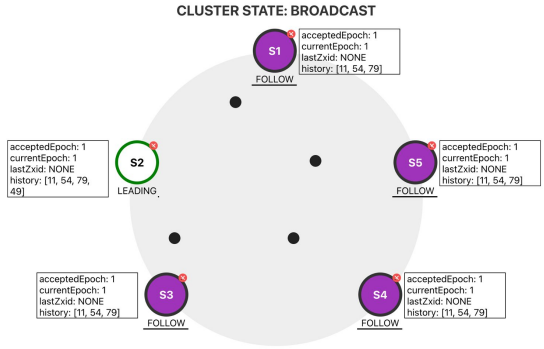


Fig. 4. Phase 3: Broadcast - Message Passing

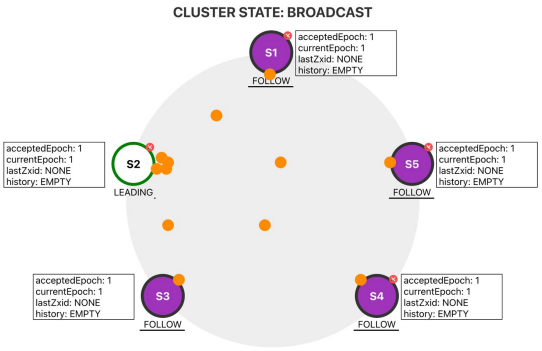


Fig. 5. Phase 3: Broadcast - Heartbeat

## IV. Technical Details for ZABViz

The technical foundation of ZABViz lies in opting for web technologies to provide a dynamic visualization. In contrast to Apache ZooKeeper, which primarily employs Java, ZABViz is built from scratch using HTML, CSS, Javascript, and React. This contemporary technology stack ensures a seamless user experience, demonstrating the adaptability of modern web development in the context of distributed systems.

### A. ZABViz Architecture

*1) Frontend Implementation:* React, a JavaScript library designed for building user interfaces, serves as the cornerstone of ZABViz's frontend architecture. The modular nature of React facilitates a scalable and maintainable design. The combination of HTML and CSS contributes to the structural integrity and aesthetics, resulting in an intuitive and visually coherent interface.

*2) Hidden States for UI Enhancement:* ZABViz employs hidden states to enhance the user interface. These discreet states play a crucial role in visualizing statistical positions, providing a foundation for potential additions such as heartbeat indicators, timers, and animations. This design choice enriches the user interface, presenting complex information in an accessible manner.

*3) Leveraging JavaScript Events:* ZABViz utilizes JavaScript events to manage message handling, user interactions, and dynamic animations. This strategic

integration ensures real-time updates and synchronization with the underlying ZAB protocol, providing users with an immersive and responsive experience.

*4) Function Calls for Seamless Interactivity:* The tool orchestrates function calls to govern interactions across components seamlessly. This architectural choice guarantees a smooth flow of data and events, enhancing the overall interactivity of ZABViz. Users navigate through various stages of the ZAB protocol, gaining nuanced insights into the intricacies of distributed consensus algorithms.

## V. User Feedback and Usability

We focused on the practical aspects of ZABViz, exploring user feedback and its overall usability.

### A. Educational Tool

ZABViz serves as a valuable educational resource, catering to researchers, developers, and students. It simplifies the complexities of ZAB, enhancing the learning experience in distributed systems.

### B. Debugging Assistance

For those actively involved in ZAB implementation, ZABViz acts as a powerful debugging aid. Users can identify potential issues and bottlenecks through visual representations of the protocol.

### C. Protocol Optimization

The visual representation of ZAB's dynamics enables a nuanced understanding of performance characteristics. ZABViz becomes instrumental in identifying areas for protocol optimization and enhancements.

## VI. Challenges and Solutions

### A. Overcoming the Protocol's Learning Curve

The implementation of ZABViz introduces challenges associated with its technical complexity, presenting a potentially steep learning curve for developers unfamiliar with distributed consensus algorithms. To address this, comprehensive documentation, tutorials, and an intuitive user interface will be essential. These resources aim to facilitate a smoother onboarding process, mitigating the complexities linked to ZABViz implementation.

### B. Adoption of New Technology Stack

Transitioning from a Java-centric implementation to a JavaScript stack demanded a nuanced understanding of both technologies, requiring attention to detail.

### C. Dynamic Visualization Challenges

The real-time nature of ZABViz's visualization introduced challenges in managing dynamic state transitions and ensuring accurate representation. Achieving synchronization with the ZAB protocol while maintaining a responsive and visually coherent interface necessitated meticulous algorithmic design.

## VII. Comparison with Traditional Understanding

In this pivotal section, we illuminate the distinctive advantages of ZABViz over traditional approaches to understanding the ZooKeeper Atomic Broadcast Protocol.

### A. Visual vs. Traditional Learning: A Paradigm Shift

In our work towards understanding distributed consensus algorithms, ZABViz is a tool offering an immersive visual learning experience while promoting intuition. It eliminates ambiguity by providing unprecedented clarity, guiding users through live executions rather than relying on traditional textual descriptions. The real-time insights into dynamic state transitions go beyond conventional learning methods and offer better experience. Additionally, ZABViz ensures enhanced retention through visualizations [16], translating abstract concepts into tangible insights.

### B. Practical Implications: Bridging Theory and Application

ZABViz excels in integrating theoretical knowledge into real-world scenarios, making abstract concept of Zookeeper Atomic Broadcast protocol tangible and relatable. This capability positions ZABViz as a superior alternative to traditional methods, which often struggle to establish a link to practical understanding. Furthermore, this makes ZABViz distinguishes itself by offering invaluable insights into debugging and optimization. It provides value by offering visualization features, a capability not found in any other tool currently available.

## VIII. Conclusion

In conclusion, ZABViz stands as a pioneering contribution to the understanding of distributed systems, particularly the Zookeeper Atomic Broadcast Protocol, by integrating it with state-of-the-art web technologies. This user-friendly tool represents ZAB's core functionalities which can be used by researchers, developers, and educators to navigate the complexities of consensus algorithms. The paper highlights the transition from theoretical exploration of ZAB to the tangible realization of ZABViz, providing a comprehensive bridge between theoretical understanding and practical implementation.

## IX. Future Scope

Our future goal is to enhance its interactivity. We envision ZABViz becoming an interactive platform, allowing users to pause the visualization at specific points, delve into detailed states of individual peers, and dynamically update configuration settings. As we continue to refine our ZAB implementation and ZABViz tool, ongoing efforts focus on improving user experience, addressing feedback, and incorporating advanced features.

## ACKNOWLEDGMENT

We would like to express our sincere gratitude to Professor Seo Jin Park for his invaluable guidance and support throughout the course. His expertise, encouragement, and insights have greatly contributed to the successful completion of this project.

## REFERENCES

[1] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-free coordination for internet-scale systems," in Proceedings of the 2010 USENIX conference on USENIX annual technical conference, USENIX-ATC'10, 2010, pp. 11–11.

[2] The Apache Software Foundation. "Applications and organizations using ZooKeeper," January 2012. [Online]. Available: http://wiki.apache.org/hadoop/ZooKeeper/PoweredBy.

[3] K. Shvachko et al., "The Hadoop Distributed File System," in Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010. [Online]. Available: https://storageconference.us/2010/Papers/MSST/Shvachko.pdf.

[4] J. Kreps et al., "Kafka: a Distributed Messaging System for Log Processing," in Proceedings of the 6th International Workshop on Networking Meets Databases, 2011. [Online].

[5] The Apache Software Foundation, "Apache Jira issue ZOOKEEPER-335," March 2009. [Online]. Available: https://issues.apache.org/jira/browse/ZOOKEEPER-335.

[6] The Apache Software Foundation, "Apache Jira issue ZOOKEEPER-1154," August 2011. [Online]. Available: https://issues.apache.org/jira/browse/ZOOKEEPER-1154.

[7] X. Defago, A. Schiper, and P. Urban, "Total order broadcast and multicast algorithms: Taxonomy and survey," ACM Comput. Surv., vol. 36, no. 4, pp. 372–421, 2004.

[8] L. Lamport, "Paxos made simple," ACM SIGACT News, vol. 32, no. 4, pp. 18–25, 2001.

[9] "Zab vs. Paxos," The Apache Software Foundation. [Online]. Available: https://cwiki.apache.org/confluence/display/ZOOKEEPER/Zab+vs.+Paxos.

[10] A. Godoy, "Consensus in ZooKeeper: From Atomic Broadcast to ZooKeeper," Aalto University, School of Science, 2012. [Online]. Available: http://www.tcs.hut.fi/Studies/T-79.5001/reports/2012-deSouzaMedeiros.pdf.

[11] Y. Souza Medeiros, "Architecture of ZAB (Zookeeper Atomic Broadcast Protocol)," June 20, 2015. [Online]. Available: https://distributedalgorithm.wordpress.com/2015/06/20/architecture-of-zab-zookeeper-atomic-broadcast-protocol/.

[12] Y. N., "ZAB Consensus Algorithm," LinkedIn, [Online]. Available: https://www.linkedin.com/pulse/zab-consensus-algorithm-yeshwanth-n/

[13] F. P. Junqueira, B. C. Reed, and M. Serafini, "Dissecting Zab," Yahoo! Research, Sunnyvale, CA, USA, Tech. Rep. YL-2010-007, Dec. 2010.

[14] F. P. Junqueira, B. C. Reed, and M. Serafini, "Zab: High-performance broadcast for primary-backup systems," in DSN, 2011, pp. 245–256. ISBN 978-1-4244-9233-6.

[15] B. Reed and F. P. Junqueira, "A simple totally ordered broadcast protocol," in Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware LADIS 08, 2008, p. 1.

[16] T. Branoff, J. Mohammed, and J. Brown, "The role of spatial visualization ability in course outcomes and student retention within technology programs," Journal for Geometry and Graphics, vol. 26, no. 1, pp. 159-170, 2022.