



ResVision - Where BFT meets Clarity

Divyanshu Malik, Karamjeet Gulati, Akshit Parmar,
Aditya Sharoff, Devashree Kataria

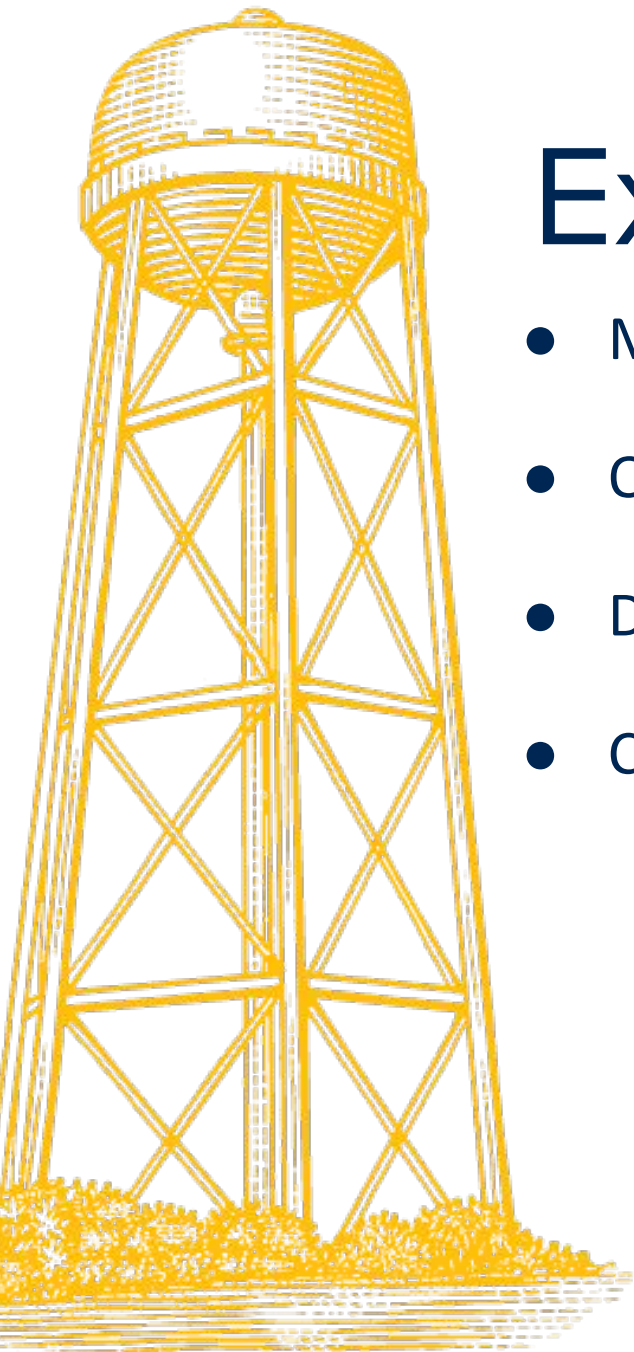
Problem Statement

- Visualize PBFT consensus protocol, showing various communication phases on real data.
- Create intuitive and scalable visualization framework to visualize SMR protocols
- Traditional methods of teaching and comprehending PBFT rely heavily on theoretical descriptions and static illustrations.
- There exists a gap in our ability to provide an intuitive and real-time understanding of how the PBFT protocol operates, communicates, and adapts to changing network conditions.

Benefits

- Educational Value: Enhance the learning experience by offering a visually intuitive tool for students and practitioners to comprehend the PBFT consensus protocol.
- Insightful Analysis: Enable users to observe and analyze the protocol's behavior in real-time.
- Teaching Aid: Facilitate instructors in conveying complex concepts effectively, fostering a better understanding of distributed systems and consensus algorithms.



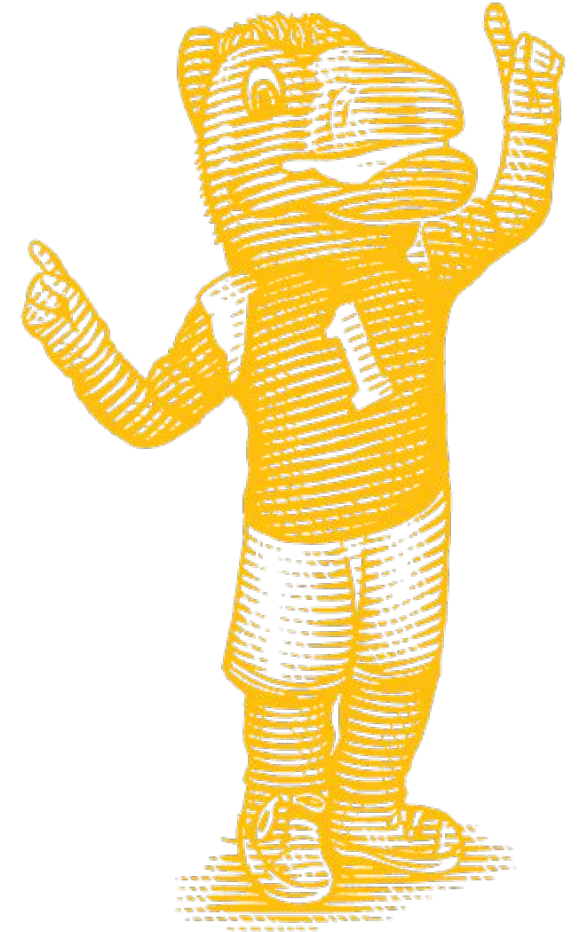


Existing approaches

- Most current visualization tools show a phase divided flow diagram
- Only PBFT is visualised
- Difficult to extend for visualising other protocols
- Cannot breakdown communication phases for better visualization

Our Approach...

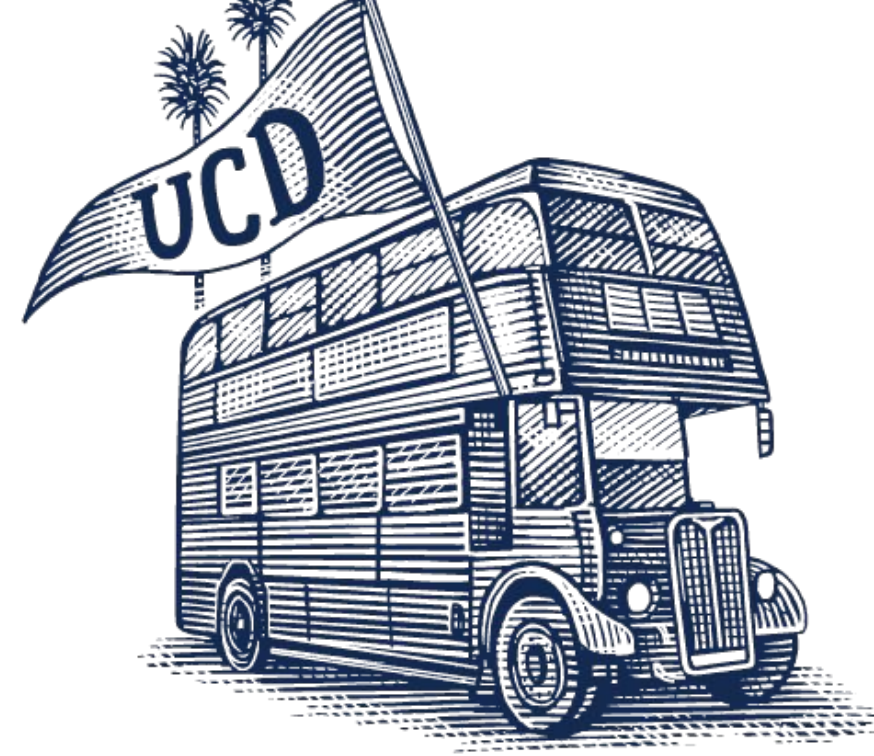
- Visual representation of node communication, highlighting message flow and types.
- Dynamic display of state transitions, providing insights into the protocol's progression.
- Uses actual communication logs from ResilientDB for visualization
- Easily extendable to make a more richer visualization
- Can be used to visualize other protocols like HotStuff, Zyzzyva etc



Technologies Used

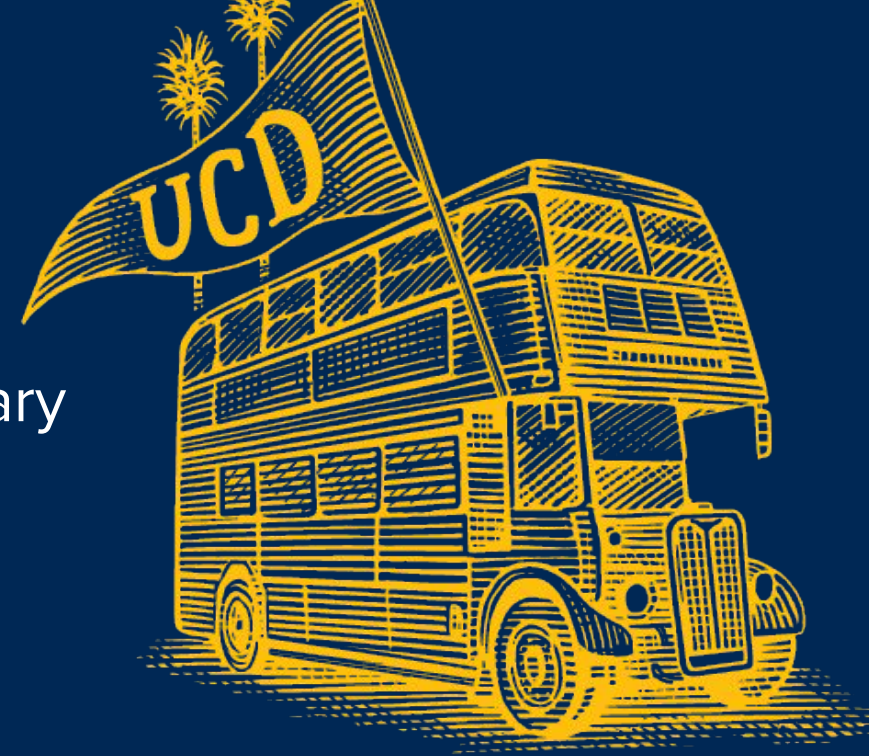
Front-end : React.js, Css, Javascript, D3.js, Figma

Back-end : Node.js, Resilientdb, Google Cloud
Platform



Front-End

- We used react.js which is a front-end Javascript library for creating user interfaces.
- Figma designs serve as the initial blueprints for our website, providing a visual representation of the intended user interface and user experience.
- The website holds information about BFT, PBFT and the phases of PBFT



Front End

- In order to create the BFT visualization we used D3.js for producing Dynamic interactive data visualizations in web browser. D3 makes use of Scalar Vector graphics, HTML5 and Cascading Style Sheets
- Developed a client-side WebSocket for receiving and parsing real-time data.
- We connected our D3.js visualizations with the data coming from the backend to simulate different transactions and phases included in the consensus protocol such as Pre-prepare, prepare, commit, and response.

Backend

- Developed Bash scripts for automating key processes:
 - Downloading Resilient DB.
 - Running Docker containers.
 - Copying files between containers and local directories.
 - Stopping and starting containers.
- Streamlined interactions with Resilient DB for increased efficiency.



Back-End

- Developed a parser for server log files, specifically focusing on transaction data.
- Implemented a server-side WebSocket server integrated with Resilient DB.
- Established a file system monitoring component for observing log file changes.



Pretty PBFT Attempts

- Pretty PBFT uses dummy data rather actual data
- Changed File path to implement our custom parsed data
- We ran into issues getting it to accept custom data even when we formatted it in such a manner that was acceptable to the front end
- Ultimately, we were **ABLE** to fully realize the Pretty PBFT made by our classmates last year with real time data



Demo

React App
React App

Demo

What is BFT?

A Byzantine fault (also Byzantine generals problem, interactive consistency, source congruency, error avalanche, Byzantine agreement problem, and Byzantine failure) is a condition of a computer system, particularly distributed computing systems, where components may fail and there is imperfect information on whether a component has failed. The term takes its name from an allegory, the Byzantine generals problem, developed to describe a situation in which, to avoid catastrophic failure of the system, the system's actors must agree on a concerted strategy, but some of these actors are unreliable.

The possibilities are beyond your imagination

localhost:3002/#home

Home What Is BFT PBFT Visualization Contribution

ResVision : Where BFT meets Clarity

Empowering Clarity in Complex Systems: Our BFT Visualization Tool brings the intricate world of Byzantine Fault Tolerance into sharp focus. Designed for both experts and newcomers, it transforms complex network dynamics into clear, interactive visualizations. Experience unparalleled insight into BFT mechanisms, facilitating deeper understanding, efficient problem-solving, and innovation in secure network design.



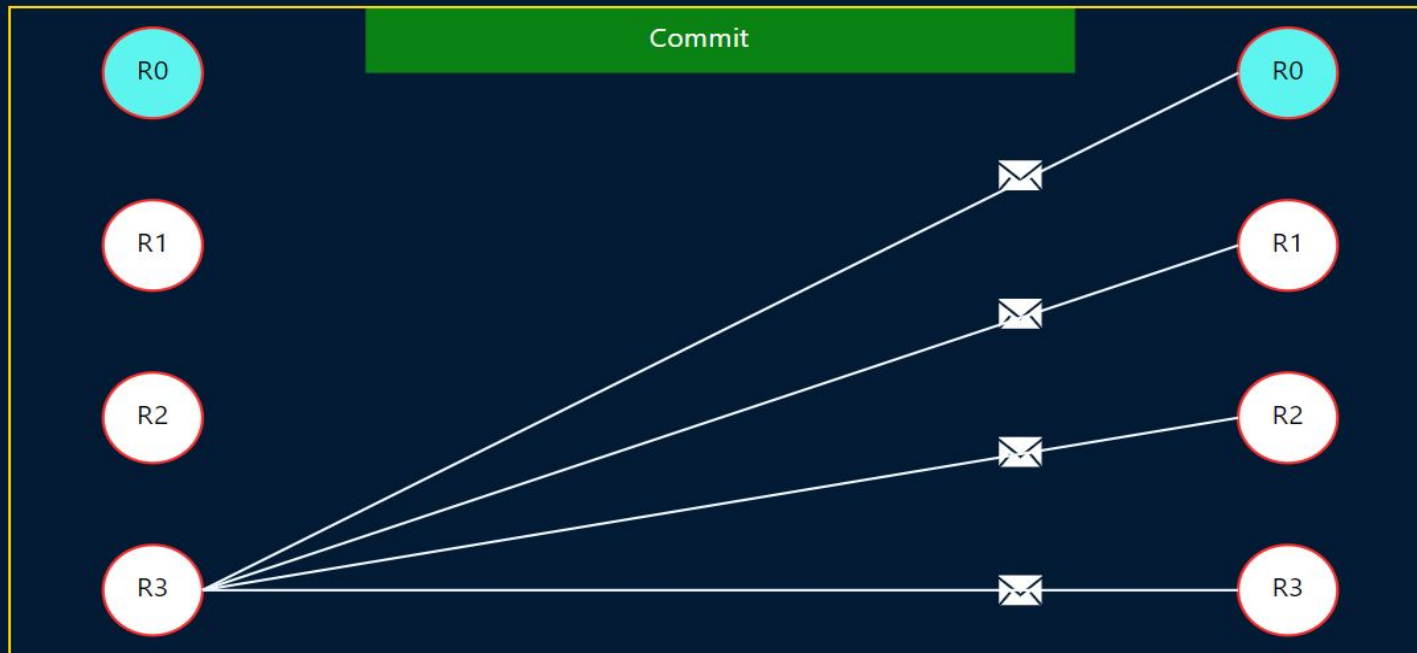
The Future is Now and You
Just Need to Realize It. Step
into Future Today & Make it
Happen.



BFT Protocols Visualization

Key:

Value:



View Change Protocol

PBFT operates in rounds or "views." Each round consists of a sequence of steps, including a view change protocol that allows the system to recover in case of faulty behavior. If a primary node (leader) is suspected of being faulty, the system can switch to a new primary in a new view.

Request & Pre-Prepare

A client initiates a request, and the primary node for the current view assigns a sequence number to the request. The primary sends a "pre-prepare" message to other nodes, indicating the proposed order and content of the request.

Prepare

Upon receiving a pre-prepare message, each honest node broadcasts a "prepare" message to the network, indicating that it has seen the pre-prepare message and accepts the proposed request.

Commit

Once a node collects enough prepare messages ($2f + 1$, where f is the maximum number of faulty nodes the system can tolerate), it broadcasts a "commit" message, indicating that it has reached a consensus on the order and content of the request.

Response

Once a node receives enough commit messages, it responds to the client indicating that the request has been processed and agreed upon by the network.