Applying Blockchain to Energy Delivery Systems

Team: sddec21-01

Website: sddec21-01.sd.ece.iastate.edu

Faculty Advisor: Dr. Ravikumar Gelli

Client: Grant Johnson, Cyber Security Research Manager, Ames Laboratory

Project Problem Statement

- Our client wants to view an implementation of a blockchain network that can store real-time data for energy delivery systems.
- Our client also wanted a way to measure this implementation using a framework known as Hyperledger Caliper.

Our Solution

- Our group used HyperLedger Fabric and Docker in order to build smart contracts and establish a network.
- We then created tests using HyperLedger Caliper and pointed it at this implementation in order to test its performance.

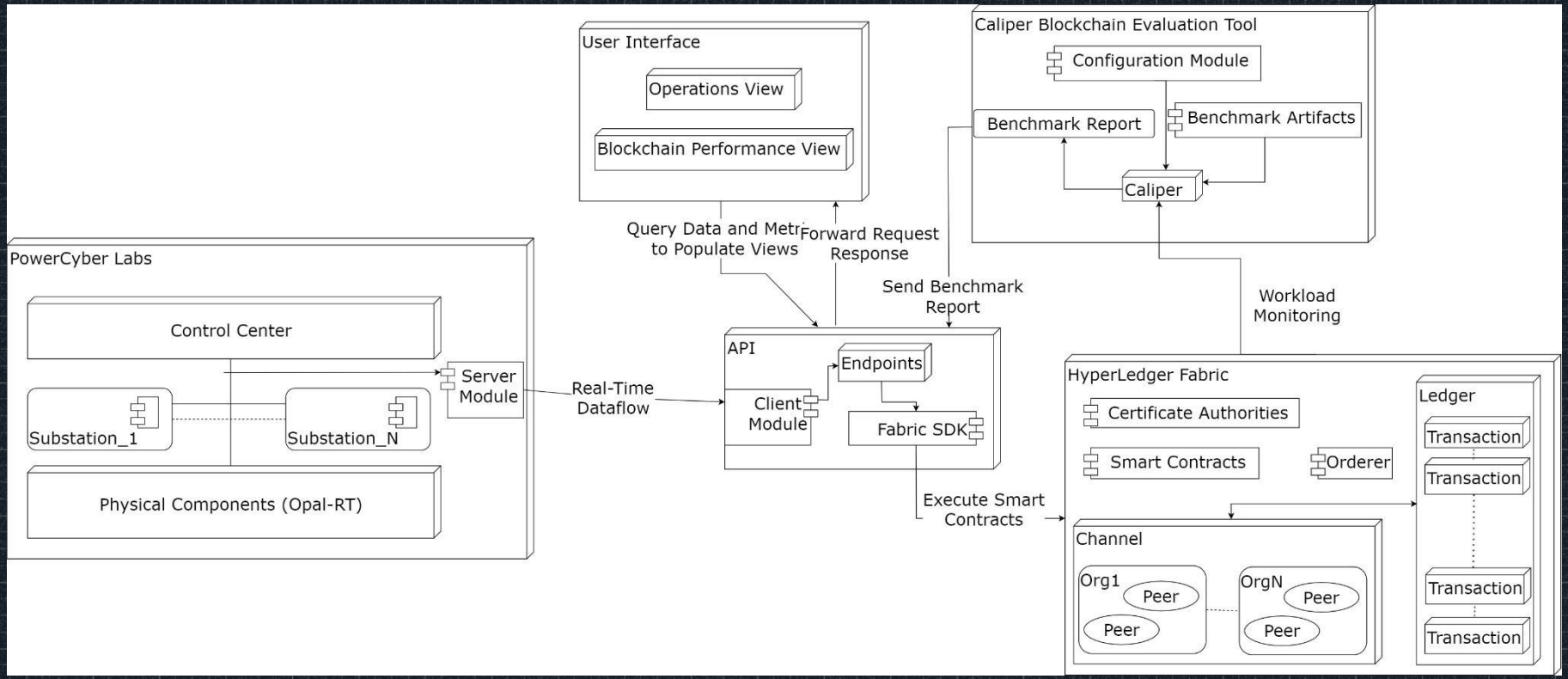
Functional Requirements

- HyperLedger Caliper makes performance metrics.
- Blockchain network consists of five organizations.
- Smart Contracts allow manipulating ledger data.
- Smart Contracts shall employ more than one endorsing node to reach consensus.
- User access limited to assigned channels.

Non-Functional Requirements

- Blockchain Network utilizes Docker for distribution.
- API queries shall not exceed 10 seconds.
- Complete documentation at project website for client’s convenience.

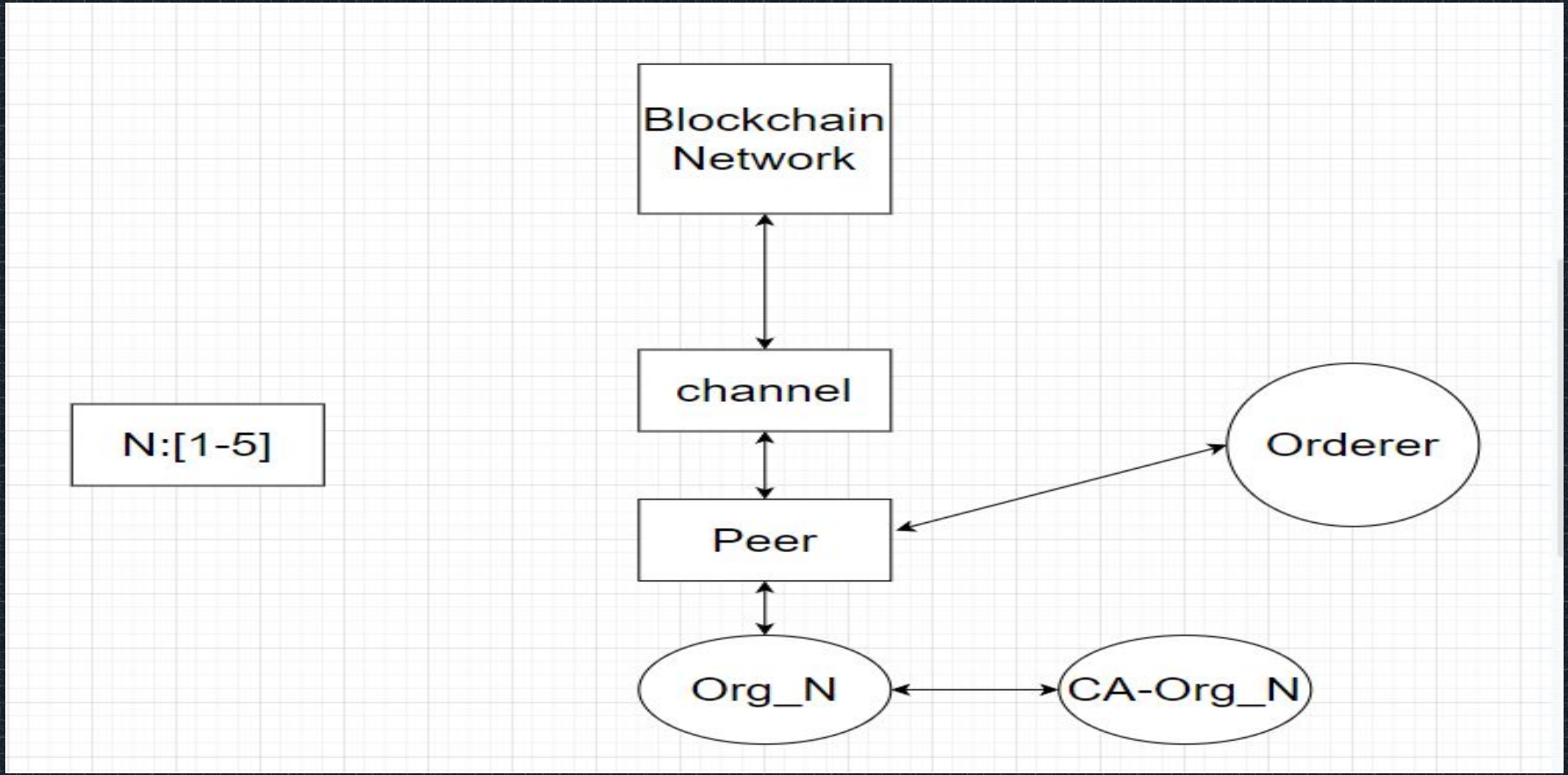
Functional Decomposition



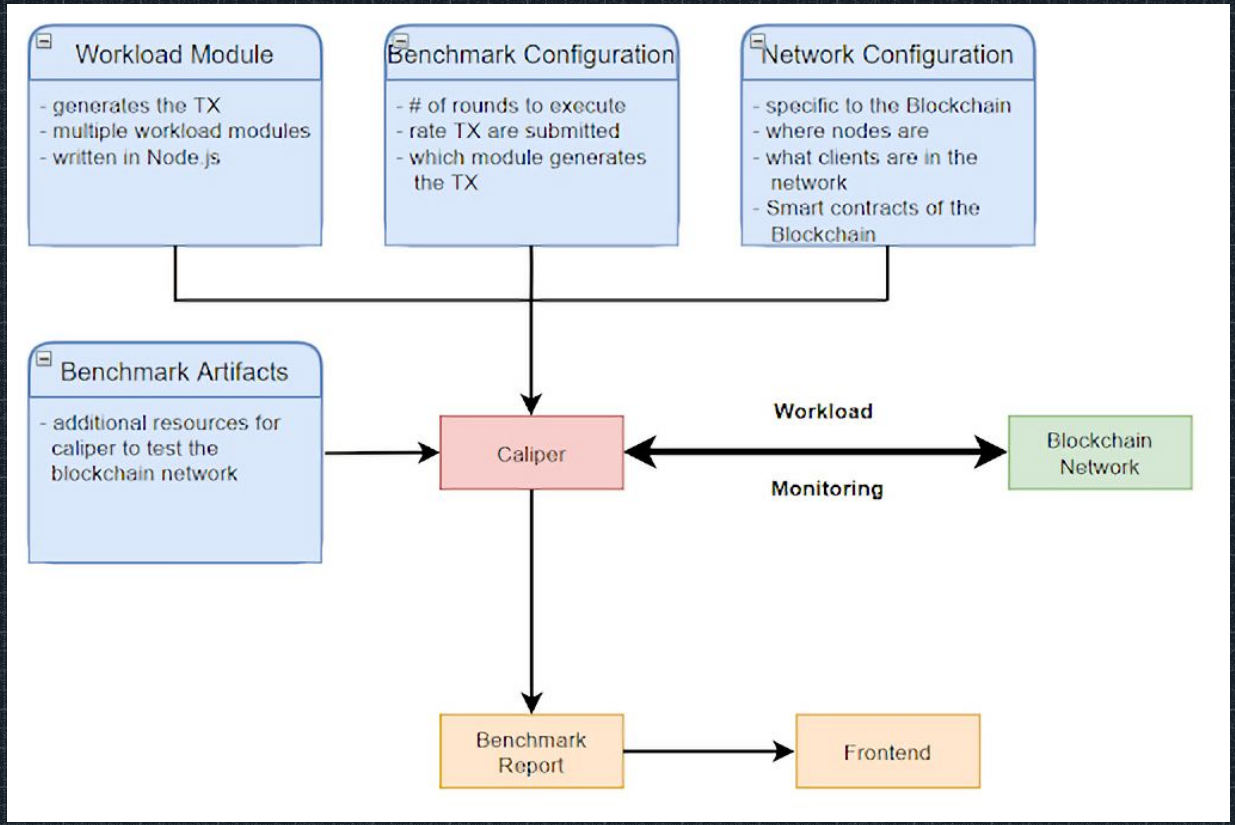
Detailed Look: Blockchain Network

- Acts as the environment for users/peers to exist on
- Built through various .yaml files
- Deployed using a wide range of shell scripts that utilize those .yaml files

Detailed Look: Blockchain



Caliper Diagram



Detailed Look: Caliper

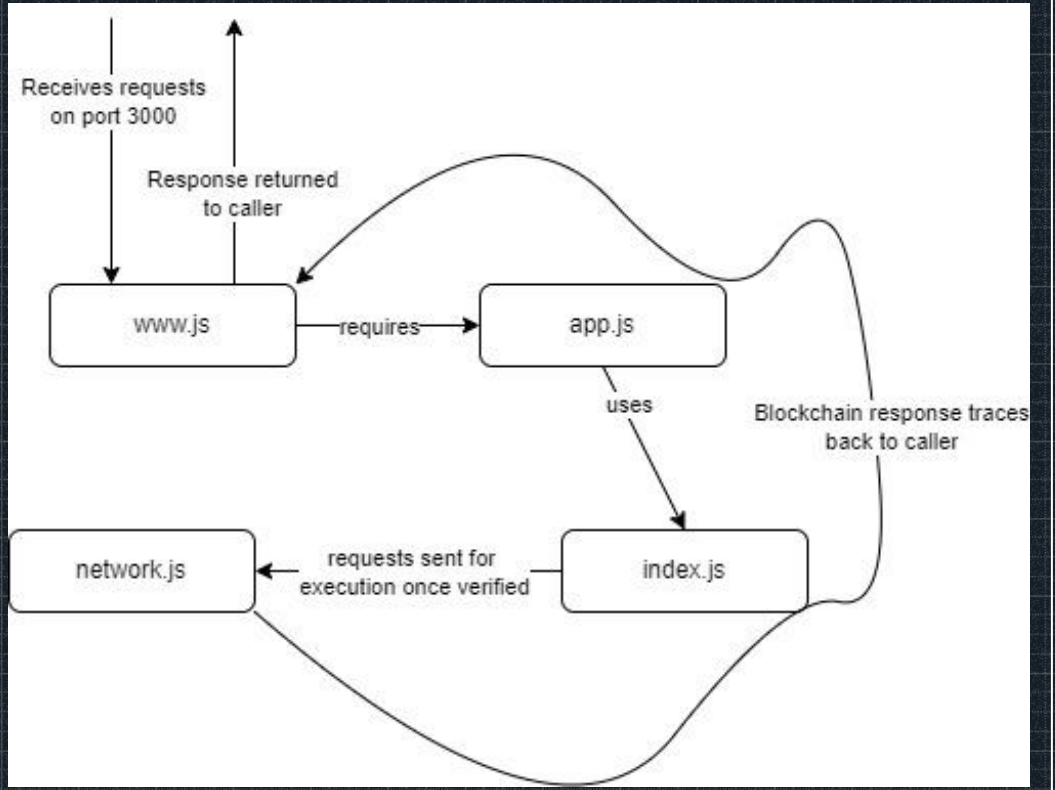
- Connect to: Hyperledger Burrow, Hyperledger Besu, Ethereum, Hyperledger Fabric, Hyperledger Iroha, Hyperledger Sawtooth, FISCO BCOS
- Metrics: success rate, transaction + read throughput, transaction + read latency, and resource consumption
- Custom use cases

Detailed Look: API

- In charge of maintaining wallet and certificate information for users/organizations
- RESTful API on the Node.js framework
  - Most notable packages being express.js and fabric-network SDK
- Documented using Swagger.IO, posted to team website

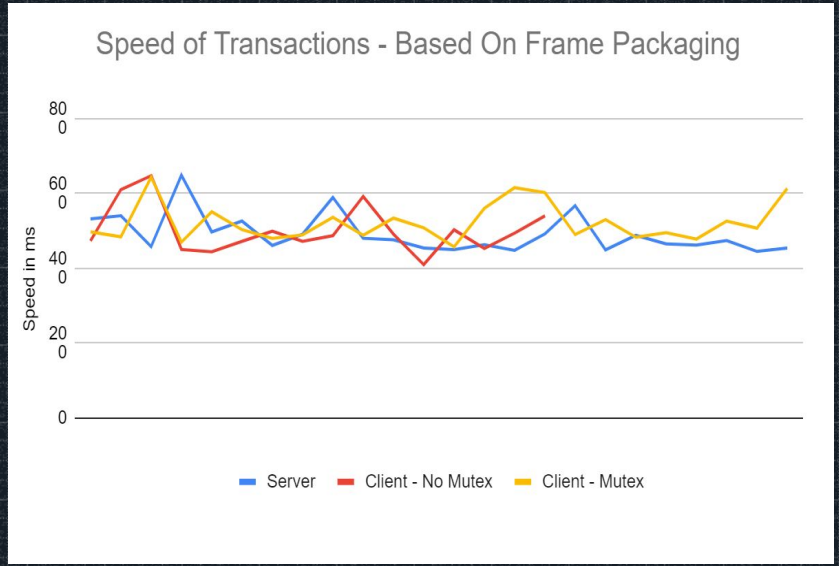
Detailed Look: API

- Testing
  - Mocha
  - Chai



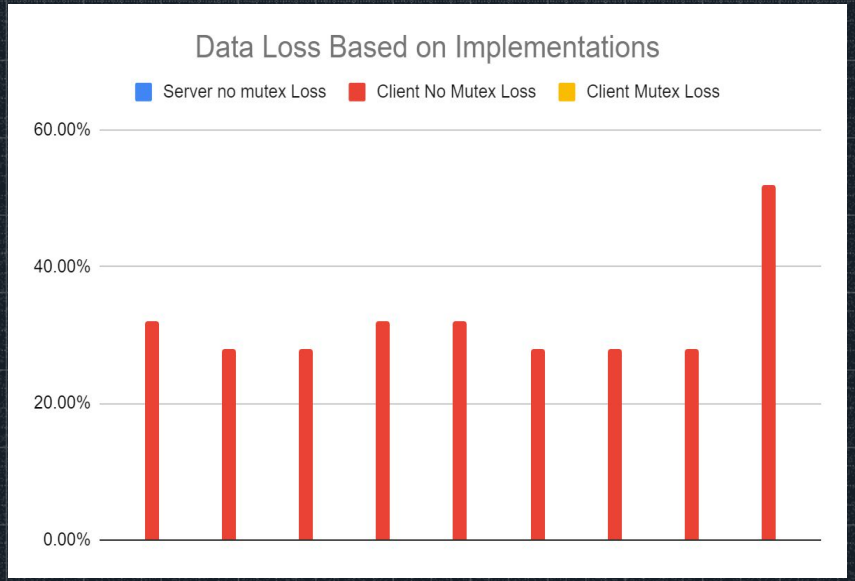
Detailed Look: Smart Contracts

- Configure
  - Data types
  - Frames
- Contracts
  - New transactions
  - Speed/Verification
- Testing
  - Results



Detailed Look: Integration

- What We Tried:
  - RabbitMQ (Queue)
  - IEEE C37.118
- What Was Final:
  - Imitation Server
  - Network - No IBM
- What Is Next:
  - Real-time data
  - Speed up contract

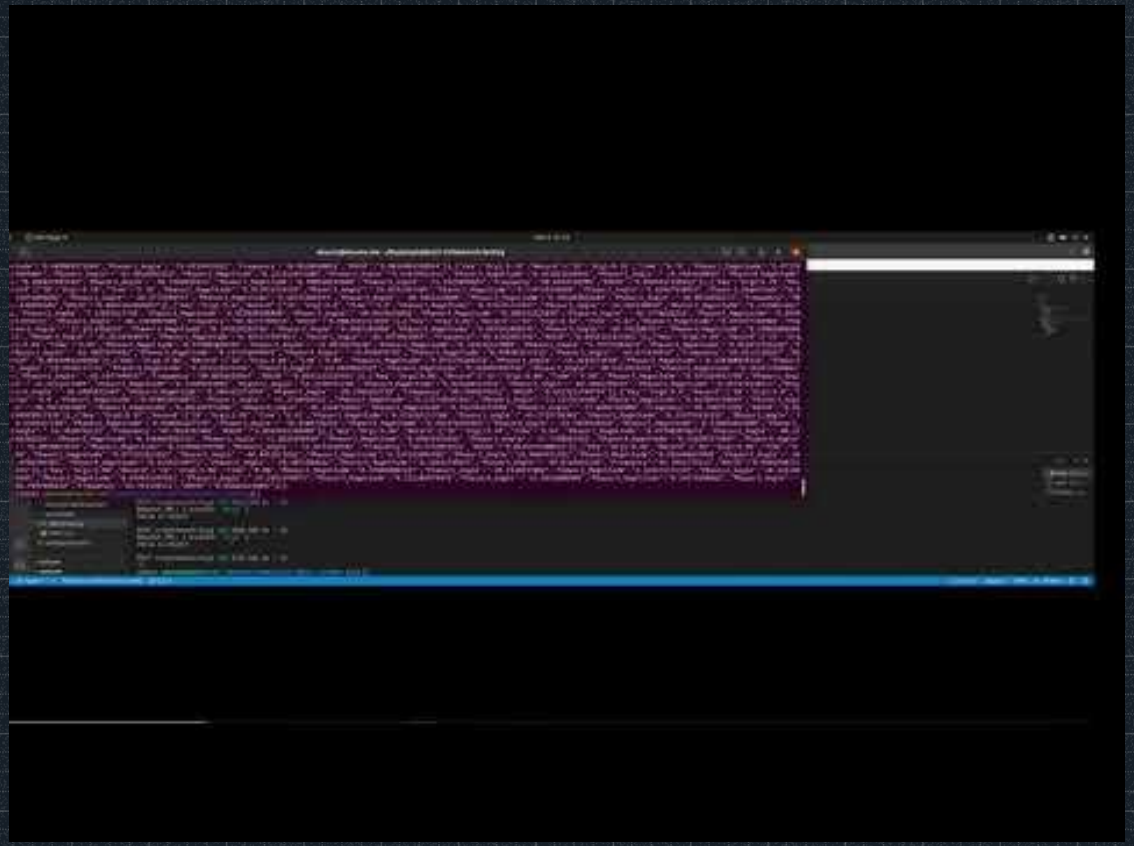


Detailed Look: UI

- Developed with ReactJS and makes use of Material UI Component Library
  - Retrieves data from API through HTTP GET Request using Axios
- Made up of 3 pages:
  - Dashboard - access to other pages
  - Blockchain Metrics - query for phasor data
  - Caliper Report - generate report

Detailed Look: UI

Project Demonstration



Lessons Learned

- Rule out design alternatives as early as possible
- Continuously integrate parallel efforts
- Confirm end-to-end connectivity from the Lab early in project lifespan

Project Q&A