Applying and Evaluating Blockchain in Energy Delivery Systems

DESIGN DOCUMENT

Team Number: 1

Client: Grant Johnson

Adviser(s): Gelli Ravikumar

Team Members:

Dylan McCormick - Report Manager

Emilieo Xiao - Test Engineer

Josh Edwards - Meeting Scribe, Blockchain Evaluator

Noah Frederiksen - Meeting Facilitator

Owen Synder - Test Engineer

Thai Pham - Chief Engineer

Team Email: sddec21-01@iastate.edu

Team Website: N/A

Executive Summary

Development Standards & Practices Used

- Agile Software Development
- Test-driven Development
- Continuous Integration and Development
- Peer Reviews

Summary of Requirements

List all requirements as bullet points in brief.

- API
  - Change API to function with non-simulated data if applicable
- **Blockchain Evaluation**
  - Create a measurement of data loss from transactions
  - Create a measurement for reliability and speed of transactions
- Smart Contract
  - Change smart contracts to function with non-simulated data if applicable
- UI
  - Display transaction data to user and show system performance measurements
  - Allow users to interact with data through queries
  - Update UI for possible new data from non-simulated environment

Applicable Courses from Iowa State University Curriculum

- COM S 309: Software Development Practices
- COM S 319: Construction of User Interfaces
- SE 329: Software Project Management

- SE 339: Software Architecture and Design

New Skills/Knowledge acquired that was not taught in courses

- Knowledge of Blockchain technologies
- Knowledge of HyperLedger Fabric
    - Smart Contracts
    - Consensus
    - Permissioned Networks
- Knowledge of HyperLedger Caliper
- Knowledge of Energy Delivery Systems
    - Specifically PowerCyber Labs

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

Our team would like to thank Gelli Ravikumar for advising and providing us with guidance, PowerCyber Labs for letting us use their resources and explaining how their hardware will interact with us, and Grant Johnson for giving us the opportunity to work on this project and partake in his ideas.

1.2 Problem and Project Statement

1.2.1 Problem Statement

Energy Delivery Systems are deployed in an environment that is geographically distributed through public internet infrastructure. The integrity of measurements, commands, and authenticity of control devices performing communication are critical for trusted operations. Blockchain is a new and growing technology that could be applied to maintain or improve that operational integrity.

1.2.2 Proposed Solution

The purpose of this project is to continue work on a blockchain based Energy Delivery System solution to solve the problem statement. Blockchain systems are very secure and attacks against them are difficult to implement, especially so in a permissioned blockchain environment. It also provides a high level of defense against loss of data, since blockchain is decentralized and stored across a network. Hyperledger Fabric's ledger is immutable, so we can trust that our records will not be altered or tampered with. This project should attempt to showcase how blockchain functions with actual PowerCyber labs hardware data and whether blockchain is an effective option for Energy Delivery Systems.

1.3 Operational Environment

The operating environment for this project will be on servers that will store the project and other data associated with it. Barring extreme natural disaster, the servers will not be exposed to harsh weather and will be properly taken care of. The servers in question will be Linux-based and exist on a secure network for the use of communication between nodes.

The technical environment involves receiving and maintaining records of data for Energy Delivery Systems through the use of blockchain technology. There is a focus on using nodes in a network to

reliably track this information and have it available for viewing through an internet/website based implementation.

1.4 REQUIREMENTS

1.4.1 Functional

Blockchain Evaluation

Caliper will be used for at least two major tests dealing with transactions or resource utilisation.

Blockchain Network

The Blockchain Network will consist of at least five nodes.

The Blockchain Network will have at least one orderer for every organization.

Smart Contract Layer

Smart Contracts will be functionally responsible for read, update, delete and query data stored on the ledger.

User access control will be limited to assigned channels where the users can employ Smart Contract functionality specified by the channel.

Every Smart Contract will implement more than one endorsing node.

User Interface

The User Interface will be developed with commonly used web development tools.

The User Interface will distribute specified metrics and/or measurements to authenticated and authorized users.

The User Interface will issue authorized user specific operator commands to modify/process the ledger.

API

There will be an API for operations and an API for blockchain evaluation

The API will solicit the Smart Contract functions to the requests made by the web application.

All calls to the API will require authentication.

Any calls made to the Blockchain System during downtime will be provided with an appropriate error message detailing status of the Blockchain System.

1.4.2 Non-Functional

Blockchain Network

The Blockchain Network will be implemented in docker containers allowing for flexibility when running networks on different machine systems.

The Blockchain Network will be deployed to PowerCyber resources by using a CI/CD pipeline.

Smart Contract Layer

The Smart Contract Layer will have automatic testing and deployment.

The Smart Contract Layer will push updates to the Blockchain Network.

The Smart Contract Layer controllers and models will have detailed documentation that describes functionality and intended use.

The Smart Contract Layer will be subject to unit testing for all controllers.

User Interface

The User Interface will display the current status of the blockchain network.

API

The API will produce query requests that do not exceed 10 seconds.

The API will have continuous integration and deployment.

Maintainability

There will be freely and readily available documentation via the project wiki containing a detailed description of the project should the client have the need to make modifications.

The project will be developed in such a way that making changes would not compromise the integrity of the project.

Continuous integration and deployment will serve to thoroughly test and deploy.

1.5 Intended Users and Uses

The intended use of the Energy Delivery System will be for human users to interact with devices that are outputting quantifiable data. We will consider both humans and devices to be users with the blockchain service. We will define several use cases for the users:

1.5.1 Human users will be required to authenticate themselves by entering their login credentials into the web application.

1.5.2 Human users will query metrics and measurements contained within the blockchain network.

1.5.3 Devices will periodically output updated measurement data. Potentially, the users may need to make changes to the domain specific data in cases of error with metrics or measurements.

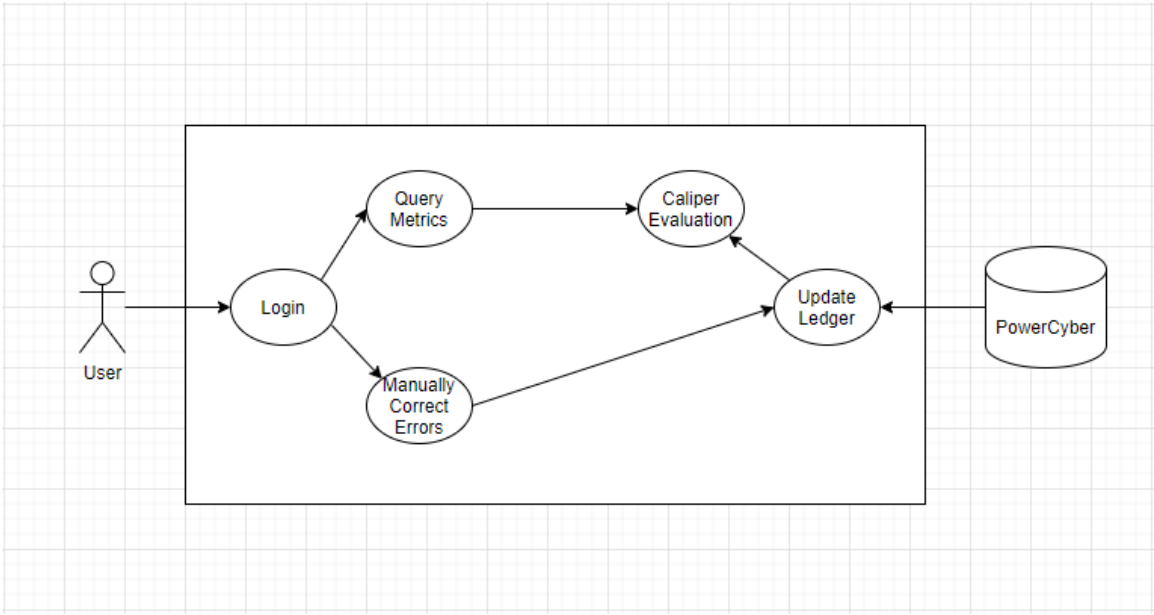

Figure 1: Use Case Diagram (https://sdmay20-12.sd.ece.iastate.edu/docs/V3_Design_Document.pdf)

1.6 Assumptions and Limitations

1.6.1 Assumptions

- The server hardware and operating system present in the PowerCyber labs will be made readily available
- Will be provided access to devices within the PowerCyber labs

1.6.2 Limitations

- The project has no budget, therefore we are limited to the PowerCyber resources
- HyperLedger Fabric must be used as the permission-based distributed ledger framework

1.7 Expected End Product and Deliverables

1.7.1 - Fully Functional Blockchain System

There shall be a system of Nodes working within virtual machines in the PowerCyber Lab. Our system will depart from the previous simulated PMU design, and will be converted to be fully functional with a real datastream from the PowerCyber lab.

1.7.2 - Authentication and Powercyber Integration APIs

Our system will operate through 2 APIs. One will host Smart Contract CRUD operations for data within the system, and the other will be focused on handling incoming data from PowerCyber and exposing functionality from that system.

1.7.3 - In-Depth Blockchain Evaluation

Our project shall include an emphasis on the evaluation of the Blockchain system via Hyperledger Caliper, which will help analyze the performance and health of the Network.

1.7.4 - User Interface

The web based user interface will provide several pages devoted to displaying different information about the blockchain system. These shall include system health/security, performance data, and API testing.

1.7.5 - Project Documentation

The project will be documented mainly through the senior design website, where all design documents and API documents will likely be uploaded.

1.7.6 - CI/CD Pipeline Integration

Development of the project and version control will be continuously streamlined throughout the project term through the integration of a CI/CD pipeline in Git to maintain cleanliness and functionality within the project.

2 Project Plan

2.1 Task Decomposition

2.1.1 Front End Development

The focus of the front end will be handling the web based framework and further implementing the past team's UI and enhancing the viewing and monitoring of the data being displayed to physical devices. Details of data being viewed being the following.

- Operations: Centered around the health of the system; available PMUs, proper communication, and status of the system.

- Cyber Security: Monitoring the data stream in order to detect anomalies and keeping track of any attacks or outliers that get introduced to the system.
- Blockchain Performance: Create metrics to determine standards of performance metrics relating to the blockchain services and protocols.

2.1.3 Back End Development

The back end team's focus will be to ensure the systems involved are working properly on their own and that they communicate data effectively and efficiently to our front-end environment.

- Nodes function properly and the loss or gain of a node will not collapse the network.
- API can interface requests to the front-end GUI with required formatting.
- HyperLedger Fabric contracts and consensus are storing data effectively and the data can be viewed to check for inconsistencies.

2.2 Risks And Risk Management/Mitigation

Risk: Limited in-person collaboration and access to PowerCyber resources could hinder progress in dev cycle

Probability: 0.1

Action: Avoid

Risk: Team's knowledge level about Blockchain systems and PowerCyber could lead to flaws in the project

Probability: 0.3

Action: Mitigate - by keeping in contact with the Faculty supervisor and continuing research on the domain, the team can successfully avoid the risk. Consulting experts in the field and utilizing resources pertaining to Blockchain systems will aid in this.

Risk: Over-Reliance on the previous team's work and design decisions could lead to inherent flaws that could not be easily fixed or worked around

Probability: 0.3

Action: Mitigate - Thoroughly analyze the previous Blockchain project and determine what needs to be changed in order to meet the updated requirements. Frequent consultation of the client and Faculty Advisor will be necessary to determine past shortcomings and fixes that will solve them.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Successful implementation of smart contracts
    - The task given to the smart contract is successful and reliable
- Communication between nodes in the blockchain network
    - Each node can successfully send and receive a smart contract between each other
- User can monitor the blockchain network through the frontend api
    - Data is displayed to the user in real-time
    - Frontend Application is easy to navigate
    - Caliper is successfully integrated
- User can send commands to modify the blockchain api
    - Commands are successfully sent
    - Commands are successfully received
    - Easy for the user to send commands
    - Data displayed is altered correctly based on the commands
- Finished Integration into PowerCyber Lab
    - the network can work successfully without human intervention
    - long periods of up-time, over a few days
    - Successfully displays data of PowerCyber Lab to the user

2.4 Project Timeline/Schedule

Tasks	1/25 - 2/19	2/22 - 3/19	3/22 - 4/30	8/23 - 9/3	9/6 - 9/17	9/20 - 10/1	10/4 - 10/22	10/25 - 11/5	11/8 - 11/26
Gathering Requirements									
Research and Design Development									
Prototyping									
Network Development									
Smart Contract API									
PowerCyber Utility API									
PowerCyber Integration									
UI Development									
System Testing									

Table 1

2.5 Project Tracking Procedures

Our group will likely use a combination of Discord and Github/Git in order to track our progress throughout the course and next semester.

2.6 Personnel Effort Requirements

Task	Time to Perform Correctly
Congregate Requirements	**Total Person-Hours: 30**
Collect Functional Requirements	10
Collect Non-functional Requirements	20
Familiarize with Resources	**Total Person-Hours: 65**
Introduction to PowerLabs	5
Familiarize with Linux VMs	10
Familiarize with Hyperledger Fabric	20
Familiarize with Calliper	15
Familiarize with React/Javascript	15
Implement Blockchain	**Total Person-Hours: 25**
Understanding previous team's implementation	15
Further research on implementation to energy systems	10
Implement the API	**Total Person-Hours: 50**
Create endpoints for easy integration and expansion	15
Create the Smart Contract Layer	30
Create commands to request data	5
Implement the UI	**Total Person-Hours: 40**
Return specified data to the user	15
Use the data to create meaningful metrics	15

Sends commands to the specified nodes	10
Testing	**Total Person-Hours: 45**
Stress loading	10
Monitor the performance through Calliper	25
Check information returned to the user	5
Test all commands	15
Integration to PowerCyber Lab	**Total Person-Hours: 50**
Load all software onto PowerCyber Lab's network	15
Testing all functionality	30
Customization according to the situation	5

Table 2

2.7 Other Resource Requirements

From our faculty advisor, the group will receive VMs with Linux installed that has Docker containers to simulate nodes. Likewise, the group will be given access to PowerCyber Lab if necessary. Lastly, the group will have access to real-time data from the lab.

2.8 Financial Requirements

There are no financial requirements for this project.

3 Design

3.1 Previous Work And Literature

- The previous team succeeded in designing a client application and API, however, in order to add more simplicity, we will need to split them up into a framework of two APIs instead of one. This will add more clarity and structure by distributing the APIs into two explicit roles rather than one encompassing API.
- Another aspect the previous team had created was the UI, however, similarly to the API abstraction there needs to be further development to obtain more functionality and analysis of the blockchain's performance. The previous team sufficiently exercised smart contracts and got the metrics to display, but they fell through on bringing in displaying those metrics in a more organized manner. This is where we will come in to enhance the

UI and make it easier for users to evaluate the data being displayed from the blockchain network.

3.2 Design Thinking

3.2.1 “Define”

- The client defined a need for a Blockchain evaluation tool for secure operational analysis on PowerCyber devices.
- The client defined HyperLedger fabric as the framework for the Blockchain service

3.2.2 “Ideate”

- We shall develop a more enhanced UI to display the evaluated metrics from the blockchain network to allow easier testing, troubleshooting and clarity.
- Abstract client application to APIs to further improve the structure of the product.
- Creating integration in terms of physical devices to set up the environment of physical data collection.

3.3 Proposed Design

The design we propose will include the following components: a user interface, two APIs, a Blockchain Network, and a Blockchain evaluation tool. We will be redeveloping the user interface from previous work to encompass three views: an operations view, a cyber security view, and a Blockchain performance view. We shall define one API for the operations view to expose smart contract functionality to the operational needs of the PowerCyber devices. We shall define a second API for the Blockchain performance view that will allow the client to analyze and monitor the performance of the available Blockchain service. Furthermore, the purpose of the cybersecurity view will allow the client identify and mitigate attacks, unauthorized requests, and malicious data.

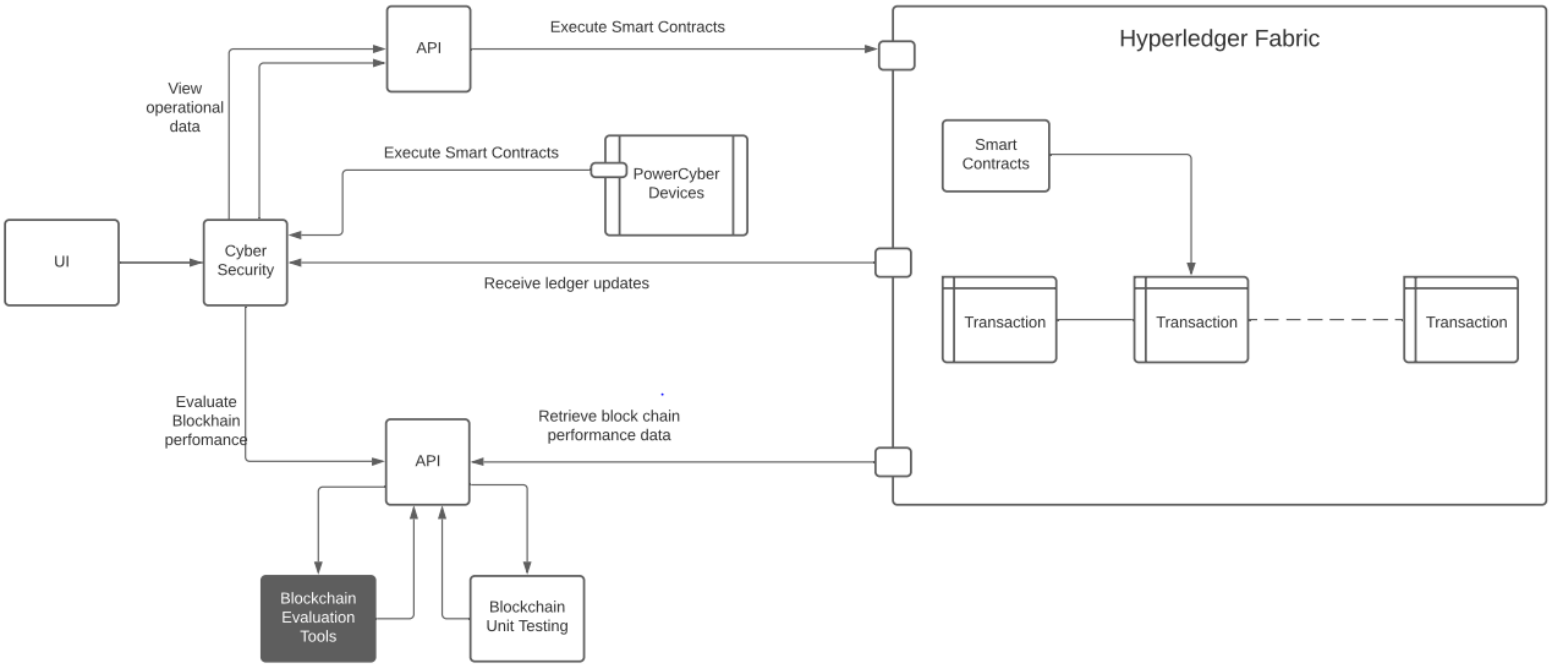

Figure 2

API For Operations View

- The API shall execute smart contracts on the data outputted from the PowerCyber devices
- The API shall decompose, convert, and process packet data from devices and format into a digital Blockchain object
- The API shall fit user case needs in terms of processing speeds from PowerCyber devices

API for Blockchain Evaluation View

- The API shall integrate with blockchain evaluation tool Caliper
- The API shall provide application to the Blockchain unit testing

Blockchain Network

- The Blockchain node system shall report ledger updates to the API
- The Blockchain node system shall receive and execute authorized and authenticated requests to run smart contracts from the API

Smart Contract Layer

- The Smart Contract shall query entries contained in the ledger if the user request has been authenticated and authorized.
- The Smart Contract shall update entries in the ledger if the user request has been authenticated and authorized.

User Interface

- The user interface shall allow an authenticated and authorized user to submit requests for device metrics and measurements from the PowerCyber labs.
- The user interface shall allow the client to view analysis and performance of the current Blockchain service by using Caliper Blockchain evaluation tool.

PowerCyber Devices

- The PowerCyber devices will consistently report metrics and measurement data to the Blockchain Network.

3.4 Technology Considerations

HyperLedger Fabric will be used for configuring, building, and deploying our blockchain network, implementing smart contracts, and adding API functionality. In previous years, HyperLedger Fabric did not support all of these features. Some of the features were split between HyperLedger Composer and HyperLedger Convector. Later on, Composer was deprecated and its functionality was integrated with Convector. Now, Convector is deprecated and HyperLedger Fabric does all the features required for setting up and deploying a blockchain network. Having all of the features in one software creates fewer problems than multiple software working together. However, this makes it more complicated in implementation as the features aren't split between softwares.

HyperLedger Fabric utilizes YAML configuration files for structuring and setting up the network. HyperLedger Fabric also uses Docker Composer for defining, creating, and deploying the containerized nodes to ensure all environment requirements are met. HyperLedger Fabric works using order nodes that exist as the central communication for the network. These nodes ensure consistency is maintained with the state of the Ledger. HyperLedger Fabric has a key value database called CouchDB to deal with transaction logs and ledgers (NoSQL document store). CouchDB is the default database for HyperLedger Fabric, therefore the best choice when working with Fabric.

In addition to Fabric, the team will be using HyperLedger Caliper to monitor and test our blockchain network. As another subset of HyperLedger, Caliper makes it easier to test Fabric's smart contracts and API. Caliper gives the team the ability to monitor resource allocation, transaction latency, transactions per second (TPS), and other performance tests. Therefore, Caliper is the best software to test the blockchain functionality.

For our frontend design, the team has decided to use React.js as our frontend framework. React.js gives us many advantages over normal Javascript. React allows the team to create complicated applications. For the scope of this project, normal Javascript does not give us the capabilities to build an application that allows the user to monitor, send commands, and modify the network. React is designed for more complicated applications which will help facilitate the creation of the web application. Therefore, ReactJS will be our frontend framework.

3.5 Design Analysis

As discussed with the Client, Hyperledger Fabric will be a good starting point in developing the blockchain network. It is likely that the team will continue forward using this framework, however if more research is done and a tool that is more suited to the project is discovered, then the design may be changed to incorporate it.

Similarly, the design of the front end UI may be altered in the future, as ReactJS is suitable for our needs, but is mainly intended for single screen applications, which contradicts our intended UI design of having multiple pages that show different analytic data about the blockchain system. Perhaps in a future iteration of the design a different library will be used that will be better for implementing our current design.

3.6 Development Process

The project will be developed using agile programming practices. Each functional requirement will be completed in 2-3 week sprints, with continuous testing and updating of requirements throughout the development cycle. This method will allow the project to be flexible if any design decisions change or if there are obstacles in development. Daily development and communication from each member of the team will allow us to consistently produce quality software that is delivered often

3.7 Design Plan

After thorough requirement gathering and planning, we will implement the solution that best coincides with each individual use case (Fig. 1) and architectural diagrams (Fig. 2).

We will begin our initial solution by defining two APIs to be used in operational and Blockchain performance view within the user interface . The first API will be for the operational view and will output metric and measurement data pertaining to the PowerCyber devices. We define a second API that will be used in the Blockchain performance view. This API will process user requests to submit evaluation criteria and will display the performance data from the Caliper evaluation tool. Both API will require calls and requests to the API to be authorized and authenticated except for when a user needs to authenticate login credentials.

The Blockchain network will contain at least five nodes and at least one orderer per organization to supply sufficient communication with peers. To support Crash Fault Tolerance the Blockchain system will enforce Raft Ordering Service to confirm ledger ordering. The global state of the blockchain ledger will be maintained on CouchDB. We will be using the HyperLedger Fabric cryptographic generation tools that implement self-signed certificates which eliminates the need for a third-party certificate authority. The device metric and measurement data from the PowerCyber Labs will be kept and maintained as a digital object on the Blockchain ledger. This will ensure the data stored on the ledger is immutable and protected from malicious intent.

While the Blockchain service is running, the Smart Contract Layer will send persistent ledger updates to the respective API. Updates will include metric and measurement data which come directly from physical data streams outputted by PowerCyber devices. At least 3 nodes out of a minimum of 5 nodes will need to come to a consensus for ledger updates. Smart Contract functions will be written in ReactJS that ultimately provides users with the

ability to read, write, query, and delta information stored on the ledger. The Smart Contracts will need to be installed on at least one peer node and instantiated on at least three other nodes for endorsement.

The user interface will allow the following functionality: login/logout, request to query authorized metric and measurement data, evaluate Blockchain performance. The user interface shall remain active should the API be down at any point in time.

4 Testing

4.1 Unit Testing

– All components of the application will be unit tested to verify consistency and performance guidelines are met.

- API: Once familiarized with the framework and what is going on in the previous teams code, we will utilize Postman which allows for automated API testing. We will be testing UI functionality, as well as HTTP requests made by those connecting to our network to request information about PowerCyber systems.
- Blockchain Network: Testing in this sector will most closely relate to the analysis that is being carried out. New chaincode will be launched upon restarting the blockchain network, and this is where the functionality of the network will be determined and verified through the use of Hyperledger Caliper.
- Smart Contracts: Smart Contract assertions will be verified through the use of Chai assertion library, instantiated with the Mocha test framework. This will allow us to verify that the contracts dealing with the PMU data are doing so properly and alleviate headaches when it comes to debugging the communication between the hardware and software.
- PowerCyber Hardware: We are waiting to get a tour of the actual facility and speak more in depth about the hardware, after that in-person meeting, we will be able to pin down both metrics and possibly a framework for testing the data collection and communication surrounding the hardware.

4.2 Interface Testing

– Discuss how the composition of two or more units (interfaces) are to be tested. Enumerate all the relevant interfaces in your design.

The relevant interfaces in our design were mentioned above, being the User Interface, the API, the Blockchain Network, Smart Contracts, and the PowerCyber hardware. The composition of the UI and API will be tested with Postman, which allows for fully automated API testing. The blockchain side of things will be tested with a Chai/Mocha combo (default for Hyperledger), while the communication between the blockchain network and the physical hardware will be taken care of by Hyperledger Caliper.

4.3 Acceptance Testing

The client will have the testing suite available to them, as some of the major reasoning behind this project is research and development, thus a range of tests applicable to a range of blockchain use cases surrounding energy deployment is important. In terms of functional requirements, we are going to discuss reasonable thresholds for latency of transactions and ensure that those are met by

using event triggered testing with the Hyperledger Caliper. The non-functional side of things will be taken care of by traditional unit testing as mentioned above.

4.4 Results

As of now, we haven't implemented any testing yet. We are still getting our feet wet with the technology, and furthering our understanding of blockchain as a whole. We plan to have a suite of various testing applications reaching through the entirety of the application which can pinpoint a component and test the correctness of the functions within that component, or look into a communication route between and verify validity of that communication through simulated data. This will allow an overall look at the health of the application and although it will not happen overnight and will most certainly have to be built out and expanded upon over time, will be one of the bigger emphasis of this project.

5 Implementation

The first thing that we will do is adapt the previous teams UI and API to fit our proposed framework. What has been supplied to us has to be expanded upon to support data collection from the physical components of PowerCyber. We will most likely utilize the previous teams data generation code to allow us to immediately begin working with the UI's and ensure their correctness before connecting that with the blockchain network, and thus the PowerCyber components. In parallel with the UI improvements, the blockchain team/specialist will be implementing smart contracts to match our goals and deploying them onto the blockchain network. Along with this, they will be looking into a multi organizational version of the blockchain in which nodes are able to be added/removed without causing issues to the network as a whole.

6 Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly reiterate your goals. Then, reiterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

At this point in time, the team has finished gathering all the functional and non-functional requirements from the client. Currently, the team is split into different parts to familiarize with HyperLedger Fabric, HyperLedger Caliper, React, and PowerCyber Lab. When the team is finished, some of them will use HyperLedger Fabric to create the blockchain network and smart contracts. Others will monitor and test the performance of the network as it is being built using Caliper. Lastly, the rest will work on the frontend application which will let a user monitor and modify the blockchain network. Afterwards, the team will work on the integration of the different softwares together as one API and start integrating the API into PowerCyberLabs. This is the best form of action based on the previous team's success.

6.2 References

E. A. IBM, E. Androulaki, Ibm, I. B. M. V. Profile, A. B. IBM, A. Barger, V. B. IBM, V. Bortnikov, C. C. IBM, C. Cachin, K. C. IBM, K. Christidis, A. D. C. IBM, A. D. Caro, D. E. IBM, D. Enyeart, C. F. IBM, C. Ferris, G. L. IBM, G. Laventman, Y. M. IBM, Y. Manevich, Srinivasan Muralidharan State Street Corp. and IBM, S. Muralidharan, S. S. C. and IBM, State Street Corp. and IBMView Profile, C. M. IBM, C. Murthy, Binh Nguyen State Street Corp. and IBM, B. Nguyen, M. S. IBM, M. Sethi, G. S. IBM, G. Singh, K. S. IBM, K. Smith, A. S. IBM, A. Sorniotti, C. S. IBM, C. Stathakopoulou, M. V. IBM, M. Vukolić, S. W. C. IBM, S. W. Cocco, J. Y. IBM, J. Yellick, Contributor MetricsExpand All Elli Androulaki Columbia University in the City of New York Publication Year, Elli Androulaki Columbia University in the City of New York Publication Years2007 - 2020Publication counts16Available for Download6Citation count951Downloads (cumulative)23, and Authors: Elli Androulaki IBM IBMView Profile, "Hyperledger fabric: a distributed operating system for permissioned blockchains," *Hyperledger fabric | Proceedings of the Thirteenth EuroSys Conference*, 01-Apr-2018. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3190508.3190538. [Accessed: 09-Mar-2021].

6.3 Appendices

Libraries and Frameworks:

CouchDB:

http://docs.couchdb.org/en/stable/

HyperLedger Fabric:

https://hyperledger-fabric.readthedocs.io/en/release-1.4/

HyperLedger Composer (deprecated):

https://hyperledger.github.io/composer/latest/introduction/introduction.html

HyperLedger Caliper:

https://www.hyperledger.org/use/caliper

PowerCyber Labs:

http://powercybersec.ece.iastate.edu/powercyber/welcome.php

React.JS:

https://reactjs.org/docs/getting-started.html

Raft

https://raft.github.io/raft.pdf