Narrated By: Dakota

IOWA STATE UNIVERSITY

CPRE/SE/EE 492 Spring 2020

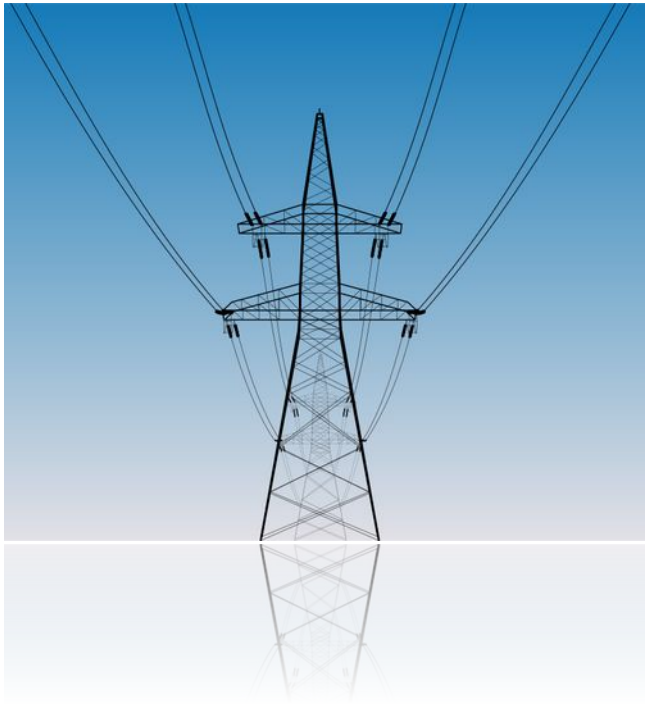
Applying Blockchain to Energy Delivery Systems

Team: sdmay20-12

Website: sdmay20-12.sd.ece.iastate.edu

Faculty Advisor: Professor Manimaran Govindarasu

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



Concerns With Energy Delivery Systems:

- Current Systems Have Flaws:
  - Uses public internet infrastructure
  - Single point of failure
  - They're susceptible to attacks
- Our Client Hypothesizes Blockchain Can Strengthen Systems:
  - Allows creation of a private & permissioned network
  - Blockchains are hosted over several nodes
- Our client wants research on how to develop a blockchain & it's performance
  - It could be wrong or possibly add more problems than it solves.

What's Wrong & How We Plan To Address It

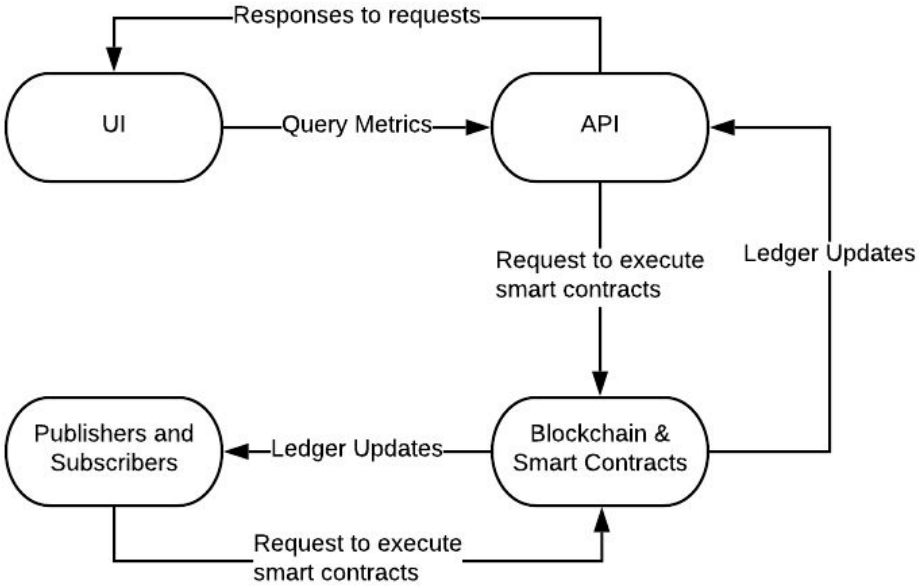
- Problem Statement:
  - Energy Delivery Systems are deployed in an environment that is geographically distributed utilizing public internet infrastructure for communications. The integrity of measurements, commands, and authenticity of control devices performing communication are critical for trusted operations.

- Proposed Solution:
  - Develop a secure network for a geographically diverse Energy Delivery System by using blockchain to develop this network to make it more reliable and secure.

Our Solution:

Our plan is to develop 4 modules:

1. UI: Uses API to interact with blockchain via smart contracts
2. API: Allows UI to interact with blockchain via smart contracts
3. Blockchain: Multiple, separate nodes, each containing a ledger, make up the blockchain. Smart Contracts allow authorized devices (&API) interact with the BC
4. Publishers: Decodes and Posts data from authorized devices to the BC when actively running

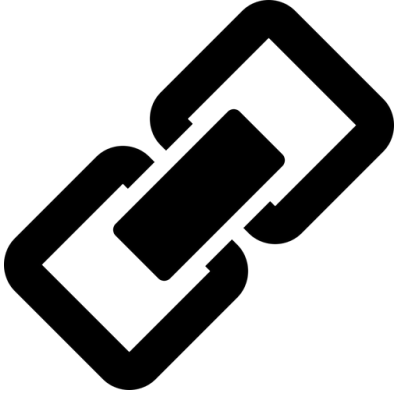


Functional Requirements:

UI	API	Blockchain (BC) & Smart Contract (SC)	Publishers & Subscribers

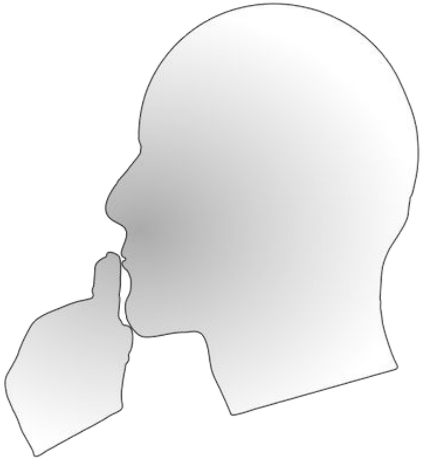
• Used via web browser
• Works independently
• Users can:
  ○ Make requests
  ○ Display returned data
  ○ Issue commands to Publishers to start/stop publishing data
	• Requires authentication for use
	• Uses BC Nodes to authenticate
	• Allows use of Smart Contract functions
		• Has ≥5 organizations
		• Uses raft consensus
		• Has multiple nodes as orderers
		• Supports read, update, delete, & query data functions
		• Stores records on the ledger
		• Defines 2+ endorsing nodes for consensus
			• 3 Publishers running, each publishing to 1 BC organization
			• Publishers sends BC the data decoded from some PCAP file.
			• Publisher sends newly published record IDs to Subscriber
			• Upon receiving new record ID, Subscriber queries the BC using the new ID

Constraints & Considerations



- Constraints:
  - No budget
  - Must use JavaScript, blockchain, & Hyperledger Fabric
  - Hardware hosting the network has storage limitations

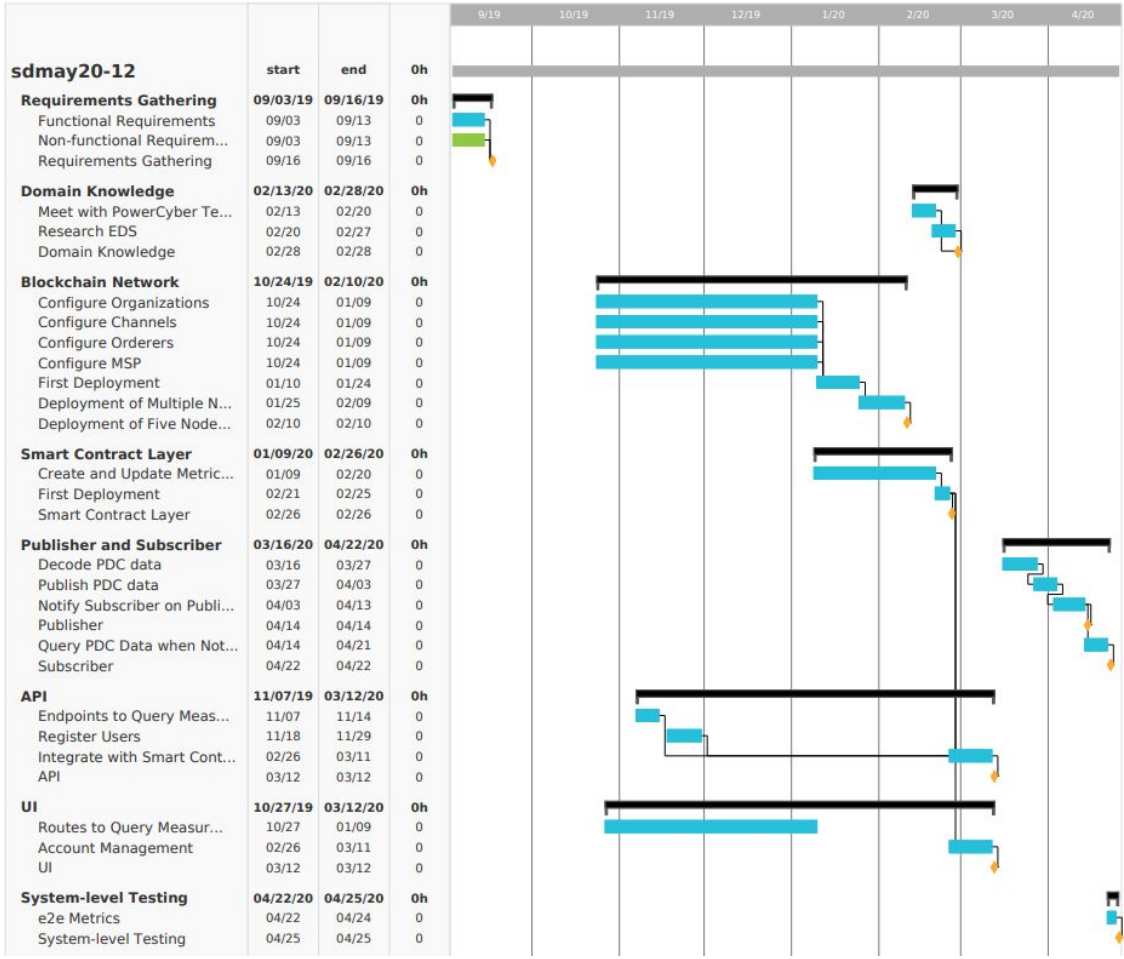
- Considerations:
  - Original software choice for framework was depreciated
  - Software choices must work with an API
  - Working with live data, possibly operated on in real time



Possible Risks	Mitigation Action
• Blockchain's possibly wrong for a solution.	• Work with client & PowerCyber to determine if there's some use case for implementing blockchain.
• Lack of domain knowledge could lead to simple mistakes.	• Visit PowerCyber, interview domain experts, asking questions to the client and advisor when necessary.
• Integration could fail	• Automate integration, maintain documentation and acceptance tests.

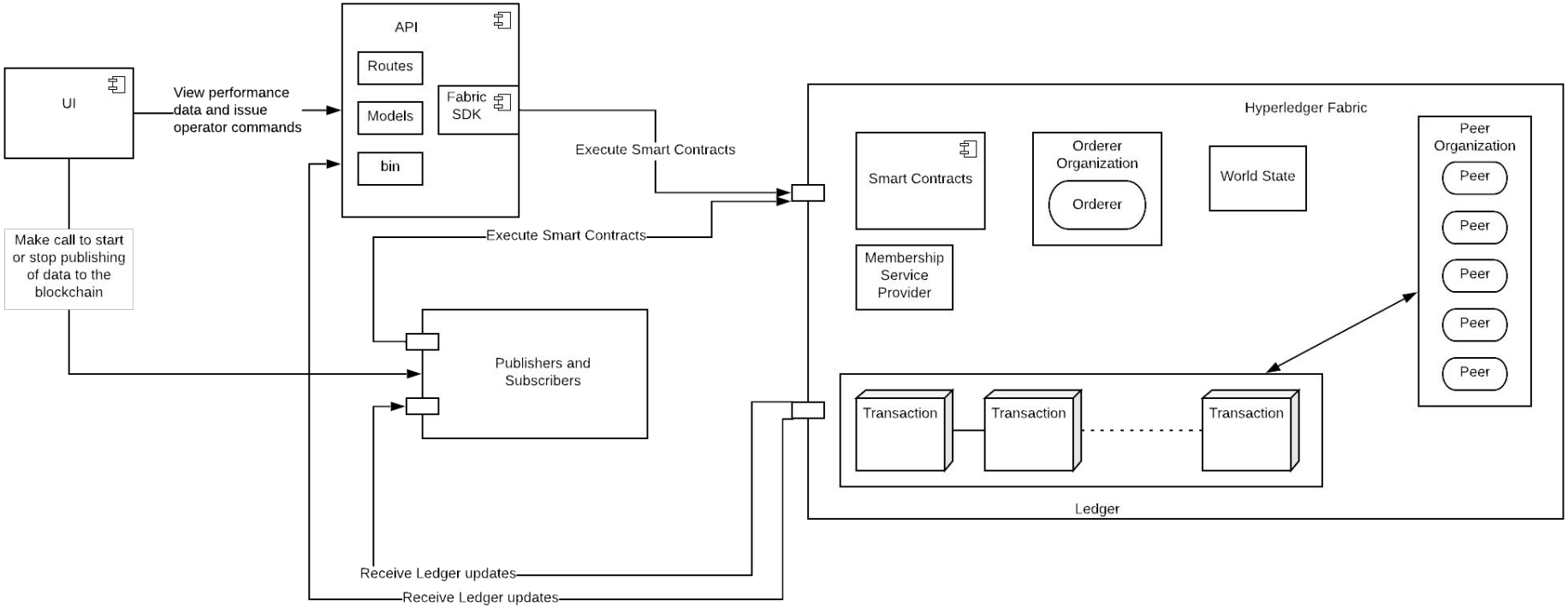
Project Schedule:

- Milestones:
  - Gather all requirements
  - Understanding of the domain
  - Configuring an example BC
  - Initial Deployment of BC
  - BC works with API & devices
  - BC has 5+ orgs
  - API initial build
  - API exposes S.C functions
  - UI displays in web browser
  - UI works with API to use S.C.
  - UI displays correct data
  - Integrate BC into given system
  - Successful end-to-end test



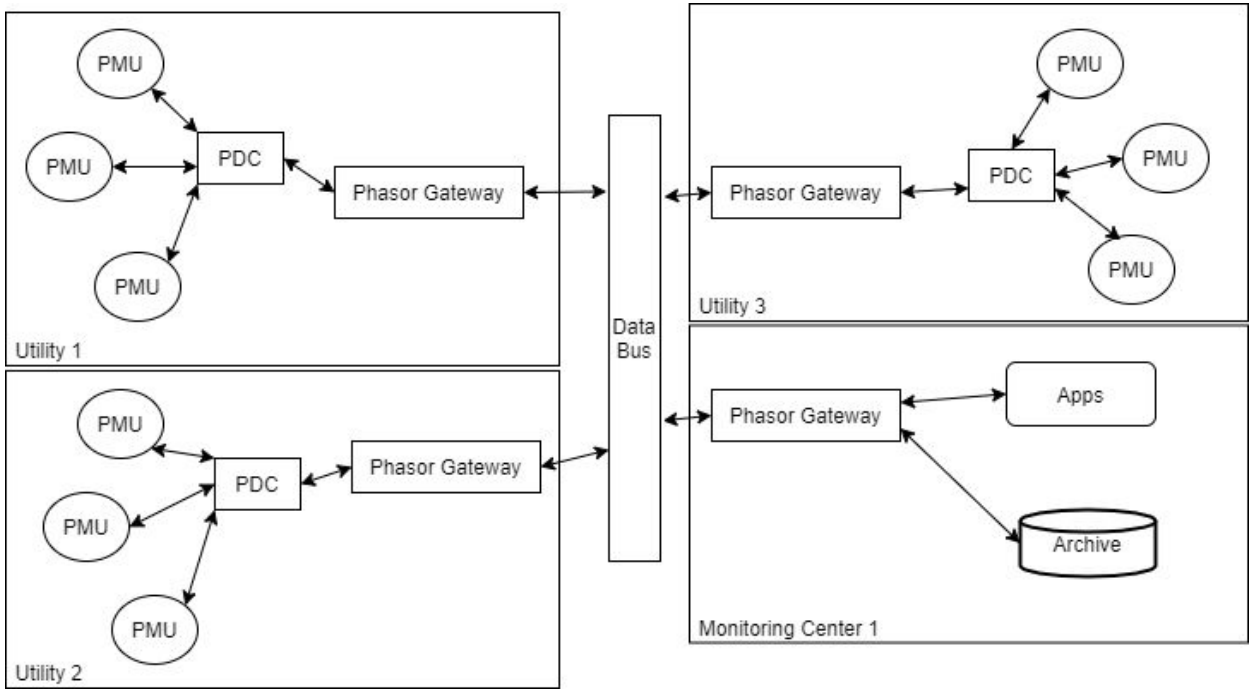
sdmay20-12	start	end	0h
Requirements Gathering	09/03/19	09/16/19	0h
Functional Requirements	09/03	09/13	0
Non-functional Requirem...	09/03	09/13	0
Requirements Gathering	09/16	09/16	0
Domain Knowledge	02/13/20	02/28/20	0h
Meet with PowerCyber Te...	02/13	02/20	0
Research EDS	02/20	02/27	0
Domain Knowledge	02/28	02/28	0
Blockchain Network	10/24/19	02/10/20	0h
Configure Organizations	10/24	01/09	0
Configure Channels	10/24	01/09	0
Configure Orderers	10/24	01/09	0
Configure MSP	10/24	01/09	0
First Deployment	01/10	01/24	0
Deployment of Multiple N...	01/25	02/09	0
Deployment of Five Node...	02/10	02/10	0
Smart Contract Layer	01/09/20	02/26/20	0h
Create and Update Metric...	01/09	02/20	0
First Deployment	02/21	02/25	0
Smart Contract Layer	02/26	02/26	0
Publisher and Subscriber	03/16/20	04/22/20	0h
Decode PDC data	03/16	03/27	0
Publish PDC data	03/27	04/03	0
Notify Subscriber on Publi...	04/03	04/13	0
Publisher	04/14	04/14	0
Query PDC Data when Not...	04/14	04/21	0
Subscriber	04/22	04/22	0
API	11/07/19	03/12/20	0h
Endpoints to Query Meas...	11/07	11/14	0
Register Users	11/18	11/29	0
Integrate with Smart Cont...	02/26	03/11	0
API	03/12	03/12	0
UI	10/27/19	03/12/20	0h
Routes to Query Measur...	10/27	01/09	0
Account Management	02/26	03/11	0
UI	03/12	03/12	0
System-level Testing	04/22/20	04/25/20	0h
e2e Metrics	04/22	04/24	0
System-level Testing	04/25	04/25	0

A Functional Decomposition



Detailed Design: Blockchain

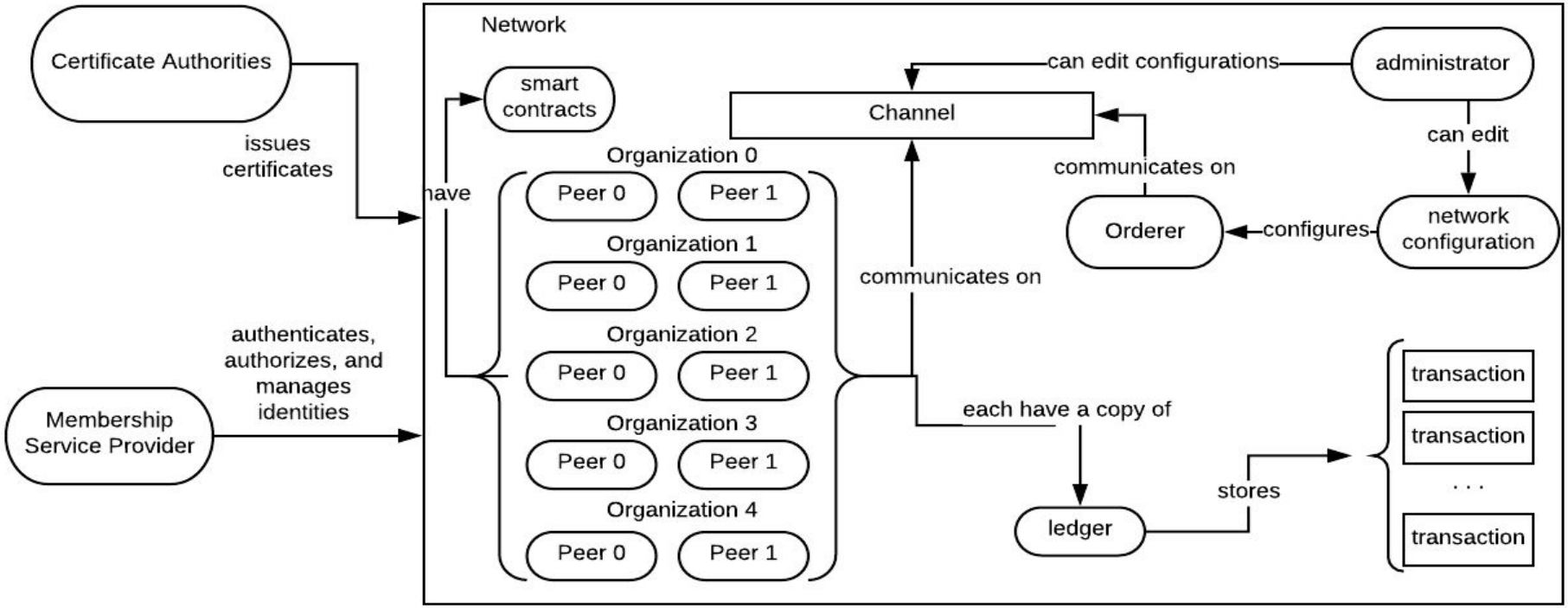
- NASPInet:
  A standard for the infrastructure of synchrophasor communication networks.



Detailed Design: Blockchain

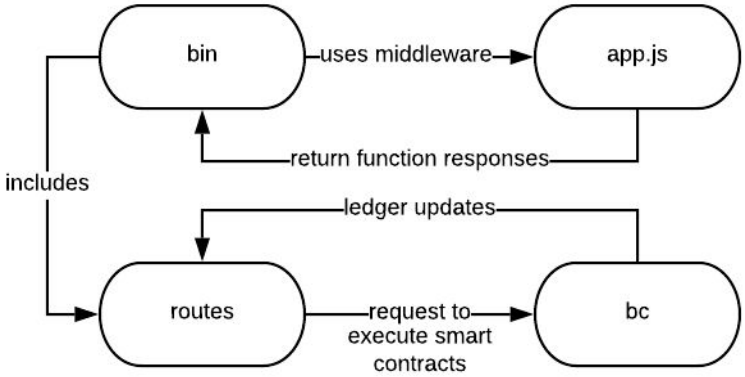
- We used HyperLedger Fabric for a permissioned distributed framework containing:
  - **Organizations:** groups interacting in the system.
  - **Peers:** hosts ledger & chaincode. The ledger holds our database. The chaincode, or Smart Contract, is a modular set of code (contract) executed for reading/writing from/to the Ledger.
  - **Certificate Authority Servers:** primary authentication method for users/external systems to interact with the blockchain network. Uses certificates for authenticating users.
  - **Orderers**: used for validating Ledger updates by peers.
  - **Ordering Service:** formed from all orderers together, validates a transaction submitted by a peer using raft consensus.
  - **Publisher:** a tool for publishing synchrophasor data to the blockchain.

Detailed Design: Blockchain



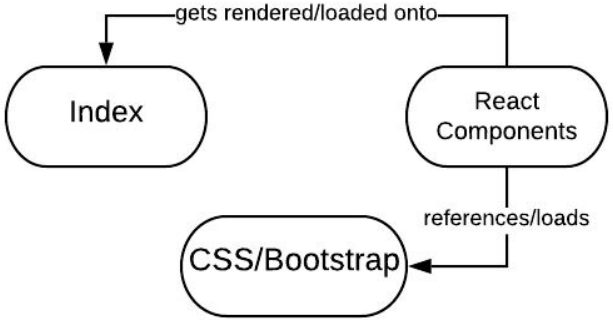
Detailed Design: API

- A RESTful web service utilizing Node.js, Express.js, and sdmay-fabric-wrapper.
- Accepts query/user registration requests from UI.
  - API selects appropriate certificate & formats a request to execute a smart contract
  - Hands off returned value to UI
- Maintains a wallet, storing registered users' certificates.



Detailed Design: UI

- The UI is composed of two main sections:
  1. The Dashboard contains:
     - A station status board
     - A line graph showing stations' frequency deviations as well as one showing blockchain publisher frequencies
     - A metric outlier list
     - An area displaying phasor data for the stations.
  2. The Blockchain Command Center
     - Allows user to retrieve metrics, register a blockchain user, and/or to start the publishers.



Operating Environment

- API, UI, and blockchain run on linux-based virtual machines

- Hardware Used:
  - Provided Servers
  - Phasor Data Concentrator Publishing Data
  - Personal & provided work PCs

- Software Used:
  - JavaScript
  - CouchDB
  - React.JS
  - Node.JS
  - Hyperledger Fabric
  - Ubuntu Server

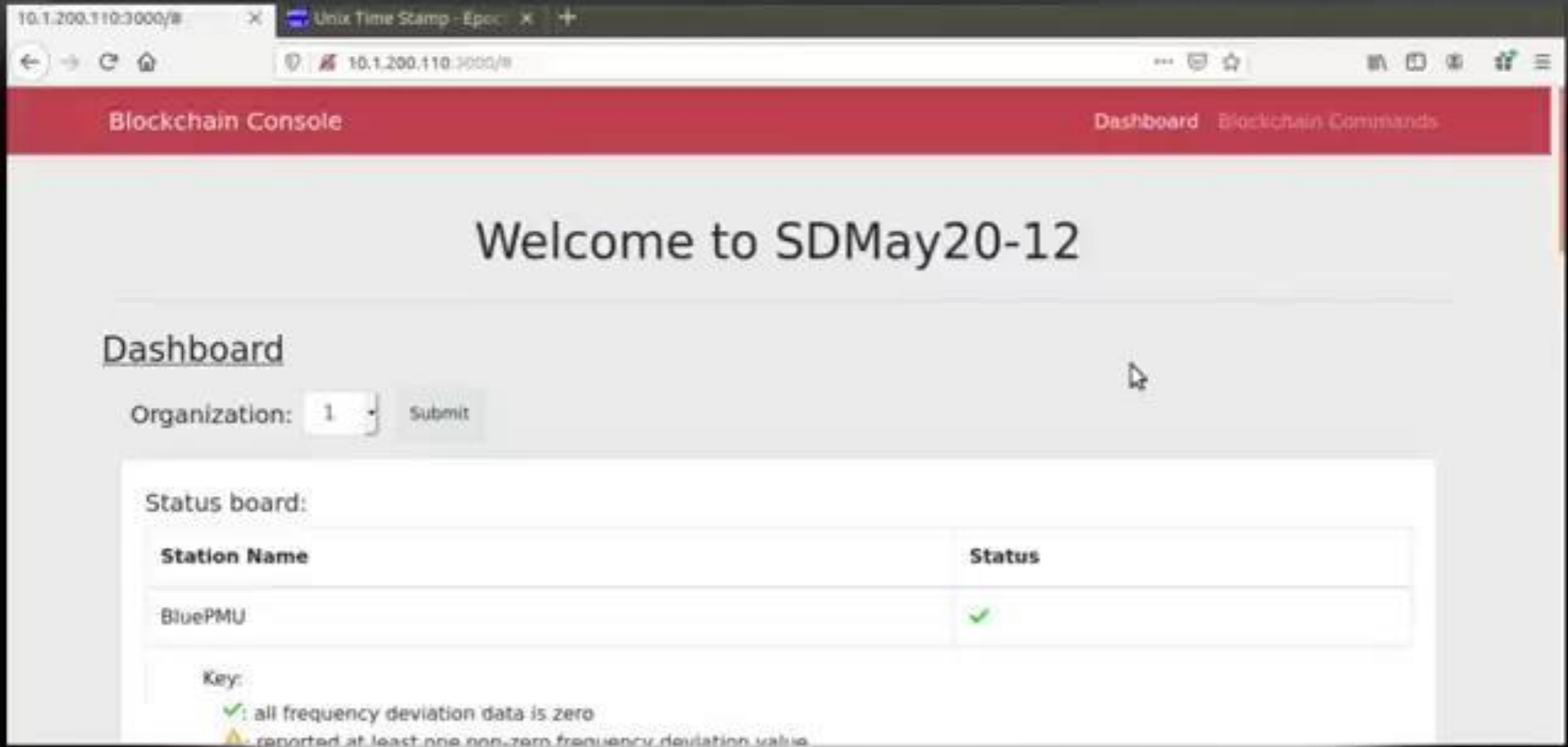
Testing Process

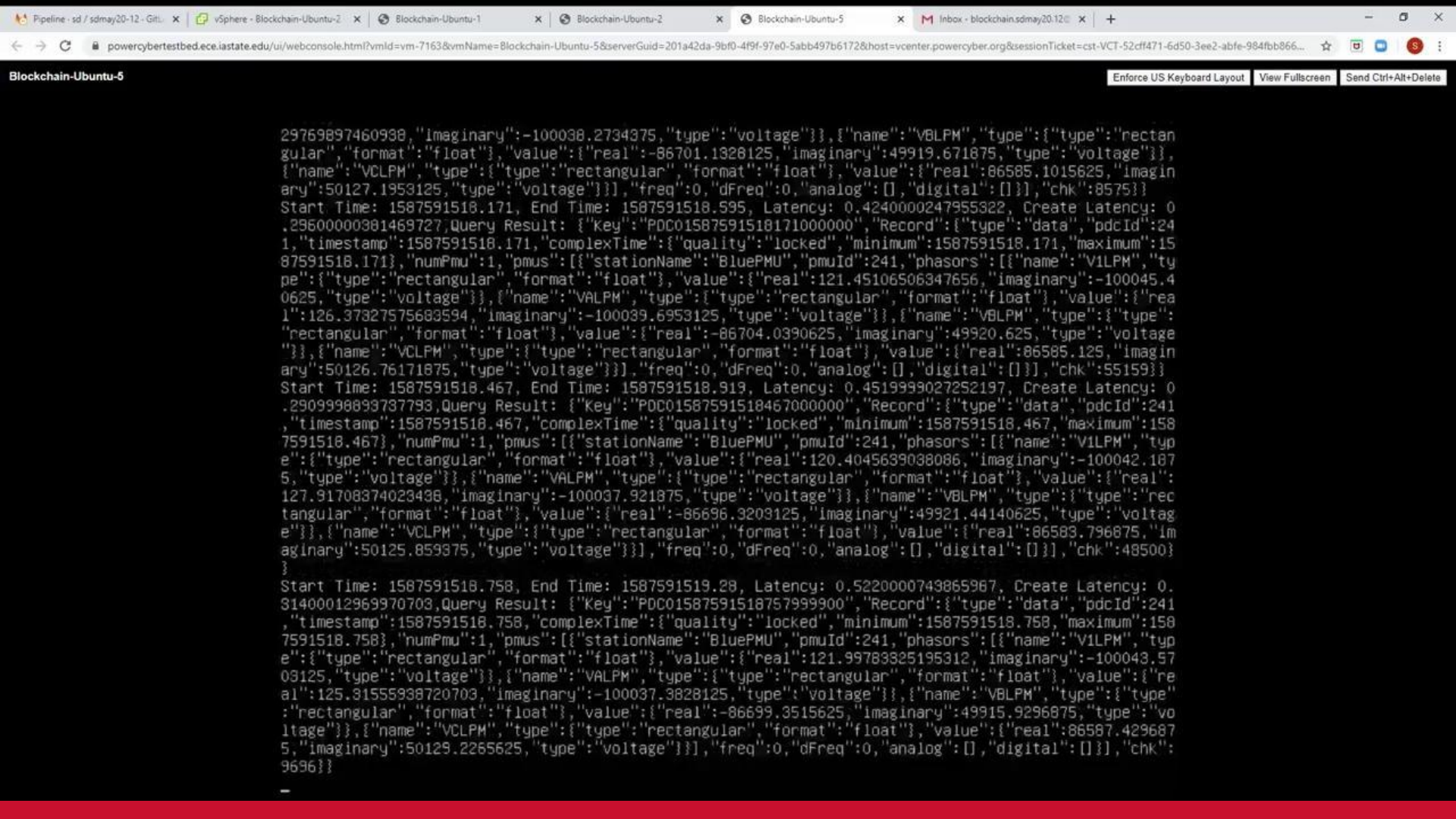
- Functional Testing:
  - Unit Testing: All individual tasks had 1+ tests for each component.
  - Integration Testing: We tested a deployed blockchain network to ensure correctness when integrating each module.
    - Tested if blockchain can accurately execute smart contracts & update all nodes.
    - The API & UI tested separately by running commands through the UI or API and comparing the results against known data.
  - System Testing: We did end-to-end tests, including a test to ensure user has access the UI, can query data with the UI, using the API to execute said query on the blockchain, then comparing results with the expected results.

Testing Process Continued

- Non-functional Testing
  - API: Unit tests for query response times for each endpoint.
  - Blockchain: It's difficult.
    - Used HyperLedger Fabric CLI to test basic queries & establishing a connection with the network.
    - We tested latency and throughput as well to get the correct configuration.
  - Smart Contract Layer: Used an eslint rule to ensure that there is JSDoc for all controllers, models, functions, classes, etc.
  - UI: Automated acceptance tests in place that simulate loss of communication between the UI and API, and loss between blockchain and API.

Project Demonstration


10.1.200.110:3000/#
Unix Time Stamp - Epoc
10.1.200.110:3000/#
Blockchain Console
Dashboard
Blockchain Commands
Welcome to SDMay20-12
Dashboard
Organization:
1
Submit
Status board:
Station Name
Status
BluePMU
Key:
: all frequency deviation data is zero

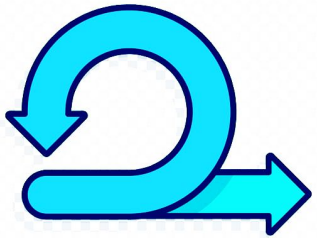

Pipeline · sd / sdmay20-12 · GitL
vSphere - Blockchain-Ubuntu-2
Blockchain-Ubuntu-1
Blockchain-Ubuntu-2
Blockchain-Ubuntu-5
Inbox - blockchain.sdmay20.12@
powercybertestbed.ece.iastate.edu/ui/webconsole.html?vmId=vm-7163&vmName=Blockchain-Ubuntu-5&serverGuid=201a42da-9bf0-4f9f-97e0-5abb497b6172&host=vcenter.powercyber.org&sessionTicket=cst-VCT-52cff471-6d50-3ee2-abfe-984fbb866...
Blockchain-Ubuntu-5
Enforce US Keyboard Layout
View Fullscreen
Send Ctrl+Alt+Delete
29769897460938,"Imaginary":-100038.2734375,"type":"voltage"}},{"name":"VBLPM","type":{"type":"rectangular","format":"float"},"value":{"real":-86701.1328125,"imaginary":49919.671875,"type":"voltage"}},{"name":"VCLPM","type":{"type":"rectangular","format":"float"},"value":{"real":86585.1015625,"imaginary":50127.1953125,"type":"voltage"}}],"freq":0,"dFreq":0,"analog":[],"digital":[]}],"chk":8575}}
Start Time: 1587591518.171, End Time: 1587591518.595, Latency: 0.4240000247955322, Create Latency: 0.29600000381469727,Query Result: {"Key":"PDC01587591518171000000","Record":{"type":"data","pdcId":241,"timestamp":1587591518.171,"complexTime":{"quality":"locked","minimum":1587591518.171,"maximum":1587591518.171},"numPmu":1,"pmus":[{"stationName":"BluePMU","pmuId":241,"phasors":[{"name":"V1LPM","type":{"type":"rectangular","format":"float"},"value":{"real":121.45106506347656,"imaginary":-100045.40625,"type":"voltage"}},{"name":"VALPM","type":{"type":"rectangular","format":"float"},"value":{"real":126.37827575683594,"imaginary":-100039.6953125,"type":"voltage"}},{"name":"VBLPM","type":{"type":"rectangular","format":"float"},"value":{"real":-86704.0390625,"imaginary":49920.625,"type":"voltage"}},{"name":"VCLPM","type":{"type":"rectangular","format":"float"},"value":{"real":86585.125,"imaginary":50126.76171875,"type":"voltage"}}],"freq":0,"dFreq":0,"analog":[],"digital":[]}],"chk":55159}}
Start Time: 1587591518.467, End Time: 1587591518.919, Latency: 0.4519999027252197, Create Latency: 0.29099988937377930,Query Result: {"Key":"PDC01587591518467000000","Record":{"type":"data","pdcId":241,"timestamp":1587591518.467,"complexTime":{"quality":"locked","minimum":1587591518.467,"maximum":1587591518.467},"numPmu":1,"pmus":[{"stationName":"BluePMU","pmuId":241,"phasors":[{"name":"V1LPM","type":{"type":"rectangular","format":"float"},"value":{"real":120.40456390380860,"imaginary":-100042.1875,"type":"voltage"}},{"name":"VALPM","type":{"type":"rectangular","format":"float"},"value":{"real":127.91708374023438,"imaginary":-100037.921875,"type":"voltage"}},{"name":"VBLPM","type":{"type":"rectangular","format":"float"},"value":{"real":-86696.3203125,"imaginary":49921.44140625,"type":"voltage"}},{"name":"VCLPM","type":{"type":"rectangular","format":"float"},"value":{"real":86583.796875,"imaginary":50125.859375,"type":"voltage"}}],"freq":0,"dFreq":0,"analog":[],"digital":[]}],"chk":48500}}
Start Time: 1587591518.758, End Time: 1587591519.28, Latency: 0.5220000743865967, Create Latency: 0.31400012969970703,Query Result: {"Key":"PDC01587591518757999900","Record":{"type":"data","pdcId":241,"timestamp":1587591518.758,"complexTime":{"quality":"locked","minimum":1587591518.758,"maximum":1587591518.758},"numPmu":1,"pmus":[{"stationName":"BluePMU","pmuId":241,"phasors":[{"name":"V1LPM","type":{"type":"rectangular","format":"float"},"value":{"real":121.99783325195312,"imaginary":-100043.5703125,"type":"voltage"}},{"name":"VALPM","type":{"type":"rectangular","format":"float"},"value":{"real":125.81555938720703,"imaginary":-100037.3828125,"type":"voltage"}},{"name":"VBLPM","type":{"type":"rectangular","format":"float"},"value":{"real":-86699.3515625,"imaginary":49915.9296875,"type":"voltage"}},{"name":"VCLPM","type":{"type":"rectangular","format":"float"},"value":{"real":86587.4296875,"imaginary":50129.2265625,"type":"voltage"}}],"freq":0,"dFreq":0,"analog":[],"digital":[]}],"chk":9696}}

What We Learned

- Initially handling recording 0.4metrics/second
- Adjusting blockchain network configuration landed us a final rate of 3m/s
- The goal: record 60m/s on 3 different organizations, concurrently.
- We modified chaincode & system publishing to the network to send 1 request/second, each request containing 60 metrics.

Engineering Standards and Design Practices

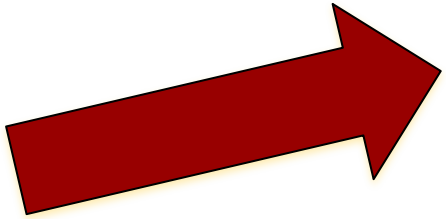
- Agile Software Development
- Test-driven Development
- Continuous Integration & Deployment
- Peer Reviews
- NASPInet Standard
- IEEE-C37.118-2005



Task Responsibilities

- **Anthony Cosimo** - (Test Engineer):
  - Worked on API, testing for UI, and Publisher. Also worked on PCAP data decoding.
- **Jacob Dawson** - (Project Manager):
  - Allocated issues to team, worked on API, wrapper, decoding and publishing PCAP data.
- **Keegan Bloedel** - (API Architect):
  - Worked on API, wrapper, and decoding PCAP data that we are getting from PowerCyber
- **Katherine Ringgenberg** - (UI Architect):
  - Designing, implementing, and testing the user interface
- **Steven Rein** - (Blockchain Architect):
  - Implemented blockchain network & chaincode. Worked on decoding PCAP data & minor API changes.
- **Dakota Moore** - (Cybersecurity Manager):
  - Worked on BC research, security aspects/implementations, and developed most of the class deliverables.

Moving Forward



- Real time data aggregation mechanisms
  - Linking in real time data sources
- Experimentation with consensus settings
- Targeted cyber security testing
- Performance of Cloud hosting vs. current solution
- Blockchain publishing performance with multiple subscribers

Project Q&A