CyHELICS

Senior Design Team 28

Design Document

Dr. Gelli Ravikumar

Justin Templeton

Tyler Atkison

Kaya Zdan

Zach Hirst

Thomas Keeshan

Matthew Nevin

sdmay24-28@iastate.edu

sdmay24-28.sd.ece.iastate.edu

April 27th, 2024 | Revision 2.0

Executive Summary

Development Standards & Practices Used

- HELICS and PandaPower use an open source BSD-3 clause license.
- OpenDSS and Dss-python are open source and have BSD-3 clause license.
- Python is an industry standard interpreted scripting language.

Summary of Requirements

Functional Requirements:

- Use CyHELICS to combine multiple substream programs and run concurrently
- Include both power grid model analysis tools and cyber security focused programs.
- Create a power grid with several transmission models that connect with several distribution models and demonstrate proper power flow.
- Power Grid must include multiple load types.
- The power grid interface must be able to simulate different grid set ups.
- The simulation must be tested in a VM environment.
- The simulation must be set up in a dockerized environment.

Nonfunctional Requirements:

- The interface must be easy to use for non technical users (city planners, grid designers).
- The user must be able to select how much of the grid they want to simulate an outage for, with specialized attacks for each one.
- The simulation must give feedback to the user about the state of the simulation (failed, complete, in progress etc.).

Applicable Courses from Iowa State University Curriculum

Cyber Security:

CPRE 230

CPRE 231

CPRE 308

CPRE 489

CPRE 539

Electrical Engineering:

EE 303

EE 455

EE 456

EE 457

New Skills/Knowledge acquired that was not taught in courses

- Docker knowledge
- Flask usage
- Connecting multiple programs to accomplish one task (ie HELICS, dss-python, Pandapower)
- HELICS usage
- DSS-Python distribution grid design and analysis
- PandaPower transmission grid design and analysis
- Use HELICS to model electrical vehicle load profiles

Table of Contents

List of figures/tables/symbols/definitions

- HELICS: a co-simulation tool that allows multiple simulators to run simultaneously and off of each others' results. This tool is necessary because otherwise, the simulators running by themselves would not accurately portray a whole and singular electric power grid.
- Pandapower: a tool that simulates the power generation and transmission of a power grid.
- DSS-Python: a tool that simulates the power distribution and load characteristics of a power grid, python wrapper for OpenDSS.
- OpenDSS: an electric power distribution system simulator used to design distribution systems and simulate their usage.
- Docker: Containerized solution to run programs on many platforms easily. Simple to create, tear down, and connect environments.
- Virtualization: Running a guest operating system on a host machine, available to Iowa State Students to safely run their programs.
- Flask: A basic python web application package that can be used with both simple and complex applications alike. It allows many other packages to be used on the python platform to run complex methods.
- Ubuntu: the most popular and flushed out linux distribution that is one of the two industry standards besides for Red Hat linux.

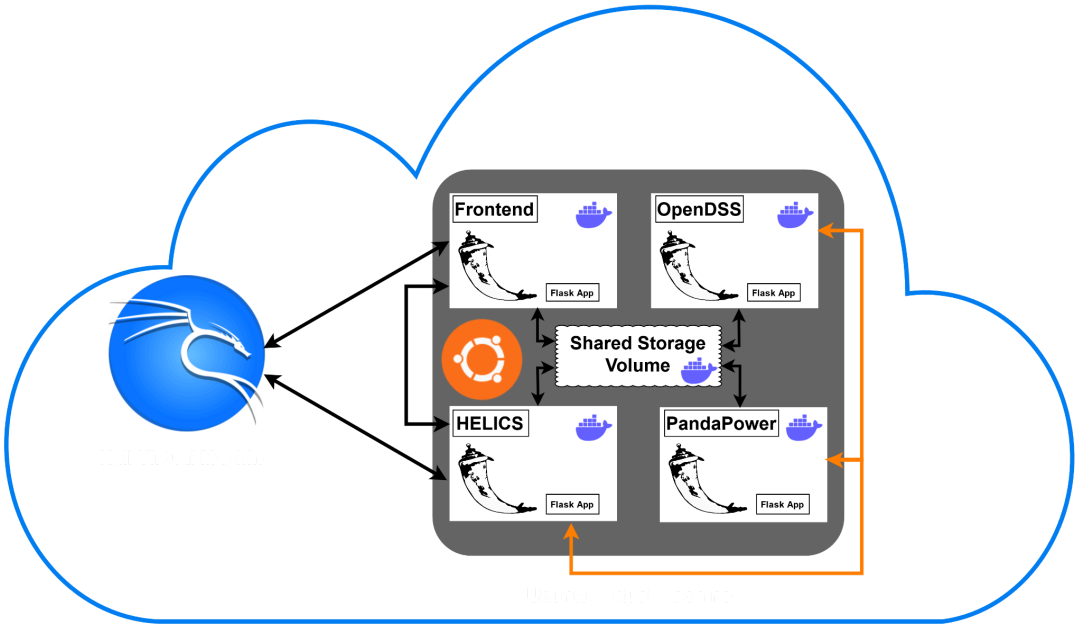


Figure 1: Basic Overview of Project

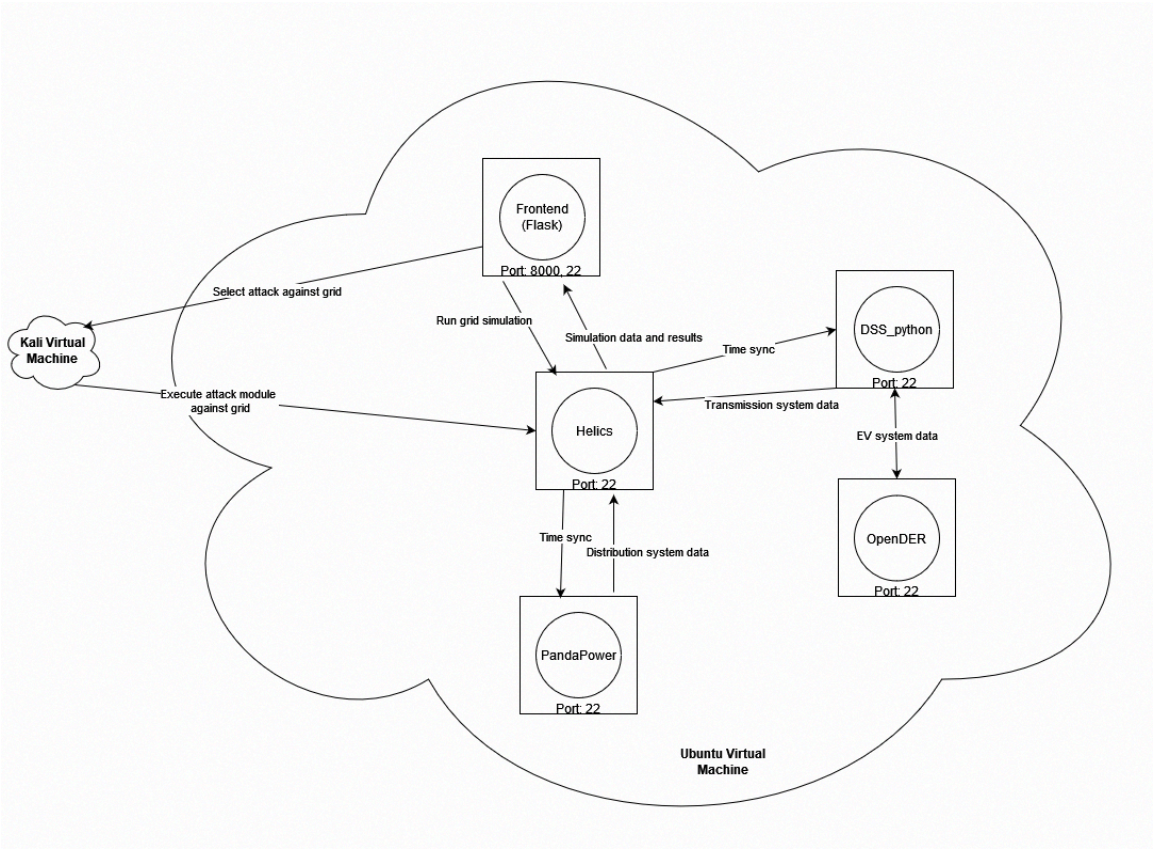


Figure 2: Our original design

1 Team, Problem Statement, Requirements, and Engineering Standards

1.1 Team Members

Justin Templeton

Tyler Atkison

Kaya Zdan

Zach Hirst

Thomas Keeshan

Matthew Nevin

1.2 Required Skill Sets for Your Project

Coding proficiency

Power grid analysis and creation

Analysis of cyber attacks

HELICS proficiency

OpenDSS proficiency

Generation of cyber attacks

Web Development (frontend and backend)

Docker

DSS-Python proficiency

PandaPower

1.3 Skill Sets covered by the Team

Coding proficiency (All)

Power grid analysis and creation (Matthew, Thomas)

Analysis of cyber attacks (Tyler, Zachary, Justin)

Generation of cyber attacks (Tyler, Zachary, Justin)

Web Development (frontend and backend) (Tyler, Zachary, Justin)

HELICS proficiency (Kaya, Justin, Matthew)

OpenDSS proficiency (Matthew, Thomas)

DSS-Python proficiency (Matthew, Thomas, Kaya)

Pandapower proficiency (Matthew, Thomas, Kaya)

Communication (All)

Docker (Justin)

1.4 Project Management Style Adopted by the team

Agile - Scrum (sprint based)

1.5 Project Team Roles

Justin - DevOps Manager and Frontend Web Developer

Kaya - HELICS Connection Manager

Zachary - Cyber Attack Simulation Manager

Tyler - Cyber Attack Generation Manager

Matthew - Power Grid Analysis Manager/Electric Vehicle Expert

Thomas - Power Grid Creation Manager

Name	Role	Contributions
Justin	DevOps Manager and Frontend Web Developer	Created development Docker instances for all members' virtual machines and set up a running Docker environment for the project. Also helped create the frontend flask app and helped out elsewhere needed.
Kaya	HELICS Connection Manager	Experimented and researched using HELICS, Pandapower, and DSS-python, and set up connecting the three tools. Worked closely with the electric team to get the simulation to have proper power flow.
Zachary	Cyber Attack Simulation Manager	Helped generate the list of preconfigured attacks against the grid. Developed the frontend website used to run the simulation, attacks, and display results. Worked on designing attacks that fit our model and exist in the real world. Finally, helped the Electrical team where needed with coding.
Tyler	Cyber Attack Generation Manager	Helped create a list of attack vectors and attacks to be used. Also helped research attacks and how they could be used in our environment. Finally, implemented one of the attacks with different severities.
Matthew	Power Grid Analysis Manager/Electric Vehicle Expert	Researched and ran simulations focusing on Electric Vehicle load profiles on HELICS.. Modified the given electric vehicle example to simulate 500 EVs. Added accurate battery and charger characteristics that apply to real world applications.. Worked closely with Thomas and Kaya to help create the grid.
Thomas	Power Grid Creation Manager	Ran simulations of existing grid examples for distribution using DSS-Python. Created transmission model using Pandapower and analyzed data. Worked closely with Kaya to get HELICS connected and simulating properly.

2 Requirements and Engineering Standards

2.1 Problem Statement

Cyber attacks against the power grid are a growing concern and can cause substantial damage to the power grid. To combat this, we created Cyhelics, a tool to help electric companies and cities to test the impact of cyber attacks against a simulated power grid.

Our group is creating a virtual distribution & transmission power grid that we can simulate cyber attacks against to help showcase potential attack and defense scenarios. It ensures that the companies running their power grid have substantial protection against attacks.

2.2 Requirements & Constraints

Functional Requirements:

- Use CyHELICS to combine multiple substream programs and run concurrently
- Include power grid model analysis tools
- Create a power grid with a transmission model that connects with the Santa Fe distribution model and demonstrates proper power flow.
- The grid must be able to handle different load types.
- Use HELICS to simulate a 500 electric vehicle load profile and inject it as a load into the Santa Fe distribution grid.
- The distribution grid must have at least 50,000 nodes.
- The simulation must be tested in a VM environment.
- The simulation must be set up in a dockerized environment.
- The user must be able to pick an attack to run using the frontend.
- There must be at least 1 attack included.
- The attack must have different severity levels.
- The attack must be focused on emulating the outcome based on real world scenarios.
- The frontend must be able to show the results of the simulation.
- Frontend must have downloadable packages for results.
- Frontend must have an archive mode to quickly look at the effects of cyber attacks.

Nonfunctional Requirements:

- The interface must be easy to use for non technical users (city planners, grid designers).
- The simulation must give feedback to the user about the state of the simulation (failed, complete, in progress etc.)

2.3 Engineering Standards

- HELICS and PandaPower use an open-source BSD-3 clause license.
- DSS-Python is open source, with no listed license.
- Python is an industry-standard interpreted scripting language.

2.4 Intended Users and Uses

The people who benefit from the results of our project are the power grid companies, the city, and the general population. Our simulation allows companies to look at the effects of various attacks and how they will affect their grid, allowing them to see where they need to focus their defense. It also helps companies see where they need to focus backup systems. Those who directly interact with the software or its productions are as follows: Power grid companies, local utilities, city planners, maintenance companies, city politicians, city citizens, and researchers. This output data can be used to find weaknesses in the input grids and to test future expansions of the grids, checking for errors along the way.

3 Project Plan

3.1 Task Decomposition

Our project had a structured sequence of events. Beginning with the setup of virtual machines within a dockerized environment. We then established connections between HELICS, Pandapower, and DSS-Python using the virtual machines we have already created within the Docker environment. This was followed by creating a transmission model that worked seamlessly with the SantaFe distribution model, which was then implemented with HELICS using our connections that we already established. Next, we set up a frontend to be able to run the simulation from and check the progress from, as well as see the results of the simulation. Finally, we built the attacks into the frontend so that users can run the attacks from the frontend.

3.2 Project Management/Tracking Procedures

Our team used an Agile Project Management approach. This gave us the flexibility to accommodate multiple adjustment periods in our project. The adjustment periods ensured a smooth transition for our team to acclimate to the proposed software. Agile development allowed for small incremental parts of our project to be developed and tested. As goals and tasks became more advanced, the agile management structure allowed us to break these tasks down into smaller attainable goals. To streamline our Agile management style, we used Gitlab for version control and centralized project management.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Our team identified milestones that we deem fit for evaluation criteria. First, in the preliminary grid phase, we simulate pre-existing transmission, distribution, and load models. We used these early simulations to learn the libraries needed to create the code needed to work with our grid setup. Moving forward into the grid design phase, we dived into designing a transmission model, ensuring proper power flow, and conducting analyses encompassing power flow, fault simulations, harmonics, and unbalanced power flow. We also made sure the SantaFe distribution model simulates properly with proper power flow. A dynamic load profile is created, emulating real-world scenarios, including electric vehicle usage and residential neighborhood power consumption, with fluctuations based on actual data. In the simulation setup phase, each simulation is dockerized individually. The attack modules phase entailed researching outcomes of past electrical grid cyber attacks (ex. the cyber attack on Ukraine's electrical infrastructure) and how to implement the outcomes into our existing grid code. Finally, we created a frontend website that allows for a pleasant and easy end user experience, and used to run simulations and select attacks. These milestones collectively shaped our project's progression, ensuring its success and security at each stage.

3.4 Project Timeline/Schedule

	1/22-29	2/5-12	2/19-26	3/4-11	3/18-25	4/1-8	4/15-22	4/29-5/6	5/13-20
Analyze pre-existing transmission and distribution models	Deliver by 1/29								
Analyze the power flow and design of models		Deliver by 2/12							
Co-Simulation between transmission and distribution models		Deliver by 2/12							
Working simulation				Deliver by 3/11					
Set up VM and Dockerized Environments	Deliver by 1/29								
Create and run attacks							Deliver by 4/22		
Frontend for attack modules									Deliver by 5/20

3.5 Risks And Risk Management/Mitigation

As our entire project is run within a dockerized VM environment, there are no significant risks associated with this project.

Risks:

- Losing our VMs and our progress - 0.3
    - Mitigation: Create regular backups and VM snapshots
- Using too many resources, causing the department to be mad at us - 0.3
    - Mitigation: Being aware of our resources and making sure nothing runs out of hand
- Bad actors can see where faults are in the powergrid - 0.5
    - This is a normal worry with all research and open-source projects that are released for the public to see.
    - Mitigation: Implement proper defense mitigations into the grid so even though attackers could possibly gain access to this tool, companies will be able to properly handle the incoming attacks.

3.6 Other Resource Requirements

Our team identified a few other resource requirements for our project to function effectively. At a base, we needed access to the Power Cyber infrastructure that was provided by our client. Additionally, our team needed a multitude of virtual machines. We needed six Ubuntu and Windows 10 machines for hosting HELICS, Panda Power, & DSS-Python, as well as a master Ubuntu machine to use for clean testing of the final product.

4 Design

4.1 Design Content

Our design content includes the design and architecture of our project. This includes the software architecture of the interworking components, as well as the design of our simulated electrical grid.

For the software architecture, we need to establish the connections between our various open source software systems, HELICS, pandapower, DSS-Python, OpenDER, establish the docker containers, and connect this system to a user interface, where the client is able to launch the attack modules and run the simulation of the grid.

As for the electrical design, we need to design an integrated power grid that includes various transmission lines on PandaPower with varying voltages integrated with DSS-Python for distribution.

4.2 Design Complexity

1. Simulation
    a. We use a Dockerized environment for our simulation to function on many different types of ecosystems with minimal adjustments being made. These Docker containers utilize a frontend, HELICS, PandaPower, and Python-DSS instances to properly simulate a power grid.

2. Grid Creation

    a. Our Solution comes out of the box with a premade power grid that users can experiment with. We also would like to allow users to input their grids, but this feasibility needs to be tested.

3. Electric Vehicle Load Profile Modeling

    a. HELICS provides an example that was used as the base of the load profile model. Modifying the battery and charger characteristics along with increasing the number of electric vehicles allowed us to model a larger load profile. This profile is closer to something that would be seen in a residential area such as Santa Fe, and has more realistic load characteristics.

4. Attack Outcomes

    a. Since we changed to an emulation standpoint, we are not able to focus on specific attack vectors. Instead, we are focusing on the outcomes of attacks. The following are the specific attack outcomes that we had planned for this project.

        i. Line tripping: by “tripping” a line, you are disconnecting the line from the grid, forcing all the power to flow to the destination from a different route. This could cause cascading effects if the power isn’t distributed evenly.

ii. Generator short-circuit: by short-circuiting a generator, you disable that generator and cause lower power generation overall. This could cause the other generators to step up their power generation in order to keep enough power flowing into the grid to keep up with demand.

iii. Load shedding: by feeding false data into the grid by meddling with the State Estimation values, a generator could be told the grid requires more power than the generator is currently generating, causing a ramp up in power generation. This could lead to more power than a line can handle at once, causing the line to shed power in order to keep below the limit of the line.

4.3 Modern Engineering Tools

- HELICS: a co-simulation tool that allows multiple simulators to run simultaneously and off of each others' results by allowing each software to speak the same language. This tool is necessary because otherwise, the simulators running by themselves would not accurately portray a whole and singular electric grid. This tool can also be used to simulate electric vehicle load profiles.
- Pandapower: a tool that simulates a transmission grid.
- DSS_python: a tool that portrays a distribution grid, python wrapper for OpenDSS.
- Docker: Containerized solution to run programs on many platforms easily. Simple to create, tear down, and connect environments.
- Virtualization: Running a guest operating system on a host machine, available to Iowa State Students to safely run their programs.
- Flask: A basic python web application package that can be used with both simple and complex applications alike. It allows many other packages to be used on the python platform to run complex methods.
- Ubuntu: the most popular and flushed out linux distribution that is one of the two industry standards besides for Red Hat linux.

4.4 Design Context

Area	Description	Examples
Public health, safety, and welfare	Many health and safety solutions rely on the power grid to function, from hospitals to homes.	Increased power grid reliability due to decreased chance of cyber-attacks impacting users.
Global, cultural, and social	People of all groups, within cities and rural areas, expect electricity to be delivered at all times if needed.	Less power outages caused by cyberattacks. More efficient designs can be created.
Environmental	Reduced attacks on power grids will lead to less power waste. This will also help with the overall design of the grid to be safer and more efficient.	Protecting power grids from cyber attacks will decrease the need for generators. Using the principle of economies of scale, this will waste less fuel.
Economic	If the electric grid gets taken down or disabled by attackers, it will result in financial losses for the power companies and any companies using that grid.	Protecting against cyber attacks will allow less outages and economic loss.
Academia	Our project will allow for more lightweight simulations to be ran for further research into cyber security for the electrical grid.	By outputting an open source simulation tool, it allows researchers to set up more simulations and conduct more research, which benefits cyber security overall.

4.5 Prior Work/Solutions

There are a couple papers that did similar projects to ours that our client suggested we look into. They are:

- HELICSAuto: Automating the Development of Cyber-Physical Co-Simulation Framework for Smart Grids
    - This paper delves into automating the HELICS API, and tested its usage by simulating Pandapower, PowerWorld, OpalRT, and PyDNP3 with Helics.
    - This shows us that Pandapower can be utilized with HELICS, as well as gives examples of other similar programs, but it is not implementing a full electric grid nor simulating cyber attacks on it.
- Defense-in-Depth Framework for Power Transmission System against Cyber-Induced Substation Outages
    - This paper takes an IEEE 14 bus system and uses it to evaluate cyber attacks. This perspective of this paper focuses on the defensive side, and what can be done to protect an electric grid against a cyber attack, as well as what portions of the grid

need to be particularly paid attention to. The IEEE 14 bus system was simulated using MatLab 2019a.
    - This paper is similar to our project in the sense that we are both simulating attacks against a power grid to check for weaknesses, however our project goes into specific types of attacks, as well as using tools such as HELICS, Pandapower, and DSS_python to simulate a more accurate and large scale power grid.
- Next-Generation CPS Testbed-based Grid Exercise - Synthetic Grid, Attack, and Defense Modeling
    - This paper focuses on creating a test-bed environment for the industry to practice incident response for power grid cyber attacks. This paper is the most similar to our project, however still has some differences.
    - The program in the paper focuses on how specific parts of the power grid go offline in the events of an attack, to help simulate a real time attack against a power grid. Our project focuses more on how specific attacks fare against the power grid, as well as allowing different types of power grid models to be tested against these attacks.

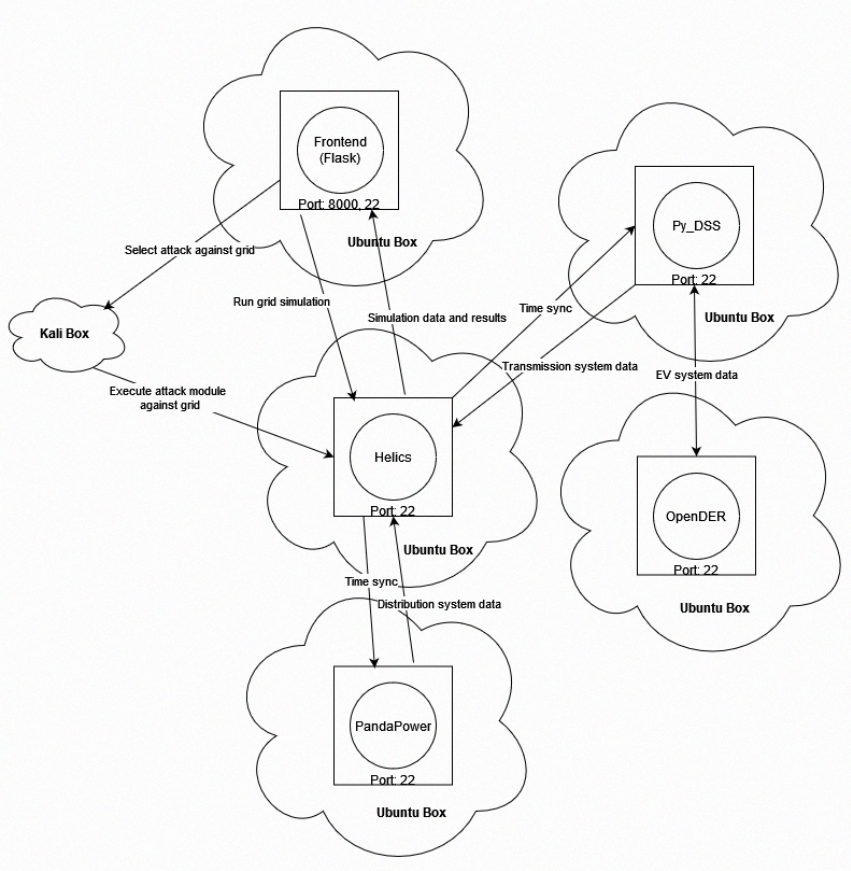
4.6 Design Decisions

- Run each program in its own Docker container in order to keep different packages separate. Keeping the transmission and distribution grids in separate Docker environments also makes the simulation more accurate as these in reality would be separate machines.
- Run the same category of programs in their own virtual machines, because it is easier to perform attacks on the virtual machines, and simulate disengaging certain parts of the grid.
- We've decided on the open source simulators that we did, since it was suggested by our advisor, many research papers have used the same tooling, and since they're open source, they incur no fees.

4.7 Proposed Design

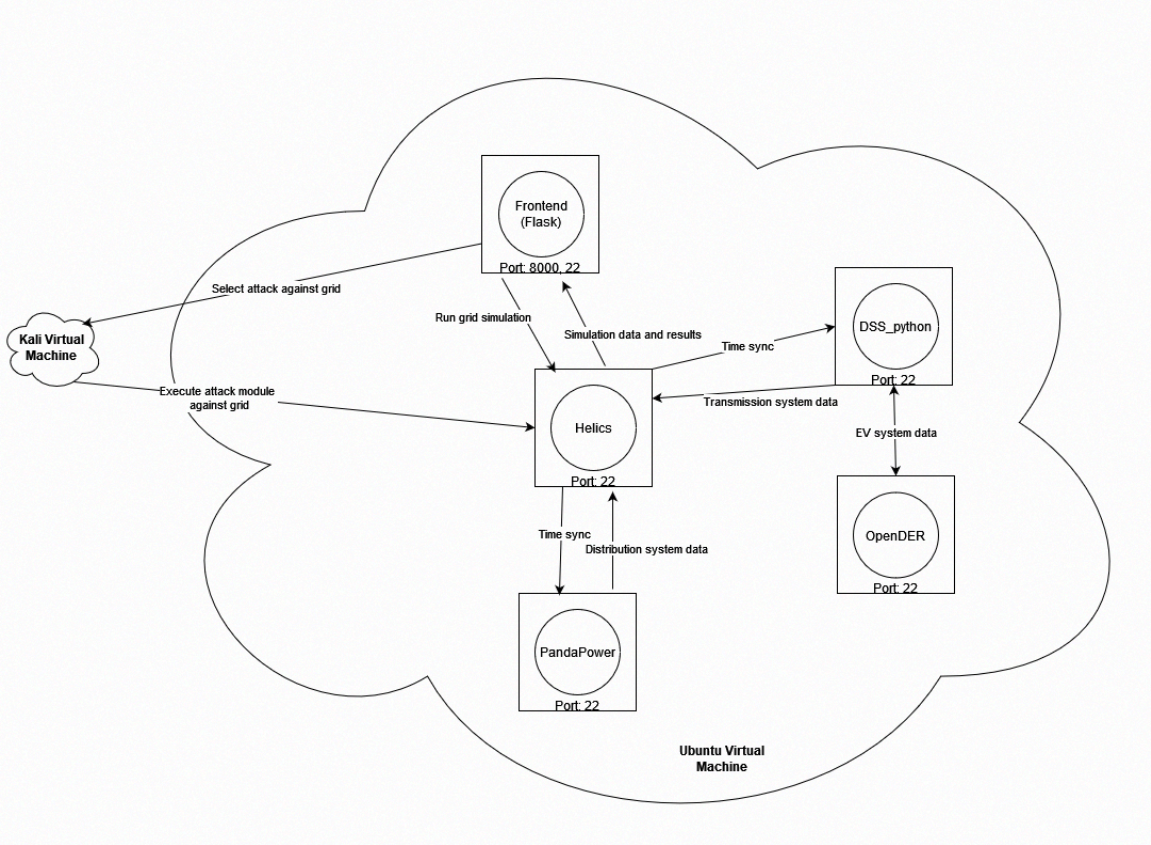
- We have experimented using HELICS, pandapower, dss_python (python wrapper for openDSS), and OpenDSS.
- We have dockerized containers for HELICS, pandapower and dss_python and are continuing to make improvements.
- We are starting experimentation with connecting HELICS, pandapower and dss_python to get a fully functioning grid.
- We are currently designing a basic grid layout.
- We are identifying potential attack vectors.
- We got access to the client provided virtual machines and are setting them up.

4.7.0 Design 0



This initial design included sectioning off each program into its own virtual machine Ubuntu box. dss_python is used to simulate our distribution systems, and is used in conjunction with pandapower, a simulator for our transmission system. The transmission system is how the high voltage energy moves across the grid, whereas the distribution system is how the voltage gets converted to lower voltages at substations to allow for distribution to all of the end users on the grid. These systems communicate with each other through HELICS, which facilitates the communication and timing of the simulation results. OpenDER provides dss_python with electric vehicle information to add more of a strain on the power grid, as well as mimic a real life scenario now that EV charging is becoming more widespread. This only communicates with the distribution simulation (dss_python) because it is one of the end “users” that are on the grid and only needs to interact with the distribution simulation. HELICS communicates the results of the simulation with our flask frontend, which is our user interface that allows clients to run attack modules against the grid simulation. The “Time Sync” arrows are information that HELICS sends out to the individual simulations to make sure they are stepping in the correct time increments and making sure all simulations are at the same time step. The time will be stepped by default in milliseconds. This is important as this is what helps it simulate an energy grid in real time, with data mimicking real world use. When an attack module is chosen and executed, this command is sent to a Kali box that will run this command against the grid simulation.

4.7.1 Design 1

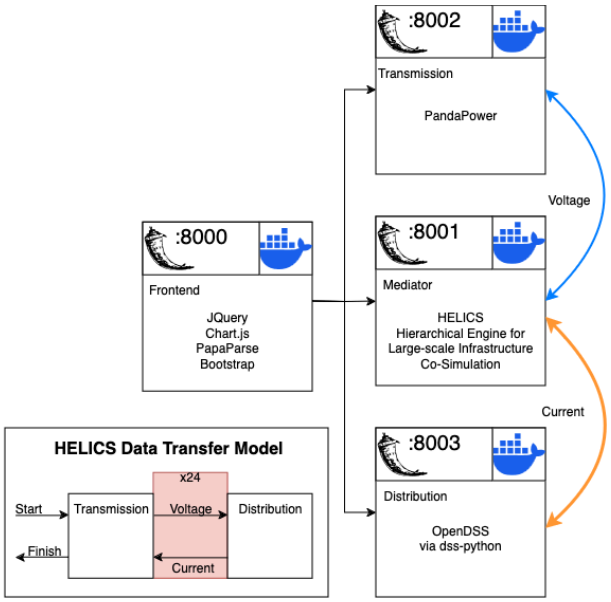


The major change made was modifying the virtual machine structure from including each program in its own virtual machine, to having each program included in one virtual machine.

This change was made for:

- Simplicity, since it's easier to manage one virtual machine with all of the programs in it, rather than tons of virtual machines all talking to each other.
- This will also result in easier connections, since we won't have to manage the communication between several virtual machines. All of the connections are done over the localhost.

4.7.2 Design 2



- This design is an improvement as we had decided to cut out the Kali VM entirely.
    - The team did not really understand that we did not need the Kali VM to attack the other VM as instead of actually attacking the VM, we were tasked with emulating a cyber attack from the inside of the system.
- We switched from using port 22 to separate HTTP endpoints on port 8000 + x for x number of Docker Instances.
- OpenDER has been removed due to time constraints, the EVs are coded in dss-python instead
    - See Appendix V for more details.
- Time synchronization has been removed from the figure as it is completely obfuscated and handled by the HELICS package housed and run in its Docker instance. To edit the timing of the simulation, it can be done by editing the HELICS configuration JSON file for each part of the simulation.
- As we are using HTTP endpoints for extra-docker communication, using Flask applications for communication is the easiest and best way to enable communication between the Docker instances. The frontend Docker consists of a full-stack approach for the web design, and all other instances consist of a backend-only API for communication.

4.8 Technology Considerations

- Using Docker instances makes the overall design much more complex, but makes it easy to deploy on a wide range of platforms, from servers of different architectures or operating systems.
- Using a Docker shared volume comes with a few downsides when it comes to how each instance of the program gets pushed to the Docker instances, but this makes it easy to keep network traffic clean and only related to the simulation.
- Having the network traffic occur over localhost allows for it to be sniffed on the network - Docker does have a way to do network traffic inside of the Docker Daemon, but this would make the simulation awfully unrealistic and cyber attacks near impossible.

4.9 Design Analysis

Our first design uses multiple VMs, which would work as well, but would add a lot of complexity, setup time, and confusion to the overall design of the system. This unneeded complexity would make the project much harder to implement and explain both to other developers and outside observers attempting to understand it. Using Dockers has the connections made to the same host, but on different ports that each App is running on.

Our design works with the existing softwares. We have HELICS sending the results from its Docker instance with Frontend and displaying graphs and other data on said Frontend. There are research papers that have used HELICS alongside PandaPower and DSS-Python, so they have worked together in the past using older versions and wrappers; we want it to work in a Dockerized environment which will be much more adaptable for others to use. We know how to use PandaPower and DSS-Python. We are currently trying to figure out how to use HELICS itself in order to connect the two - which is its main purpose.

5 Testing

The CyHELICS project focuses on simulation integration between several softwares in real time. Our project proposes a few unique software complications that are tested extensively. Due to the Dockerized environment we utilize predominantly unit testing, integration testing, and system testing. Unique challenges that we faced during the testing phase of our design included making sure that data is properly and efficiently transferred between softwares, making sure that each software correctly interprets that data transferred, and that our system works correctly as a whole.

5.1 Unit Testing

Unit Testing was done by making sure that all software packages can return their versions. Checking for errors at this step makes sure that all software packages are installed correctly on their respective Docker containers. Making sure that the HTTP endpoints are all available for sending and receiving data from the starting of the Docker instances is also vital to the program's success, and this is accounted for and tested. Unit testing is a bit difficult for our Dss-Python and PandaPower Docker instances as the modules installed are hard to check if they are required until a simulation is underway.

```python
import sys

packages = ["flask",
"flask_cors",
"numpy",
"pandas",
"pytest",
]

def test_pack():
    for i in packages:
        if not i in sys.modules.keys():
            assert False
    assert True
```

Figure 1

Unit Tests for Installed Packages

5.2 Interface Testing

We need to keep close track of what information is being sent between the Docker containers in order to make sure that there are no mistranslations in the system. We set up the expected values to

be sent out of the containers and make sure that files are of the correct format (i.e. CSVs are not being parsed as JSON). These tests can be run using packages such as pytest.

```python
import json
import pandas as pd
import io

def test_pow0(app, client):
    res = client.get('/power0')
    assert res.status_code == 200
    try:
        df = pd.read_csv(io.StringIO(res.get_data(as_text=True)), sep=",")
        json.loads(df.to_json())
    except ValueError as e:
        assert False
    assert True

def test_pow1(app, client):
    res = client.get('/power1')
    assert res.status_code == 200
    try:
        df = pd.read_csv(io.StringIO(res.get_data(as_text=True)), sep=",")
        json.loads(df.to_json())
```

Figure 2

Interface Tests for Return Types

5.3 Integration and System Testing

We set up tests for the integration through testing HELICS, as HELICS requires all software (pandapower, python_dss, Docker, Flask frontend) to be correctly integrated with each other to work. Since HELICS is a bridge software, if one of the software components does not work, HELICS does not function properly. These tests are run through a series of basic calculations to ensure correct output. The results of HELICS are displayed on a Flask frontend application, which further displays the proper connection between HELICS and the frontend.

5.4 Regression Testing

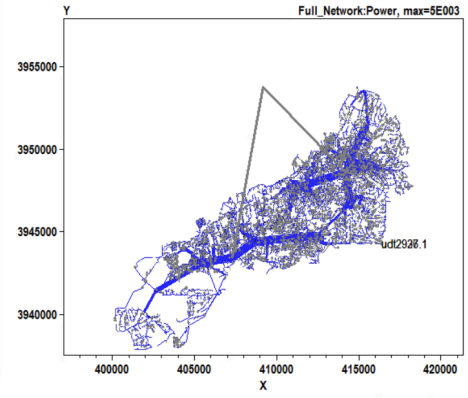
We are using Docker to ensure that new additions and features can be added at any point. It is rather easy to add a new Docker container to the existing nodes that we have set up. Altering the code will be easy to add new features with Docker as its connections will be made over localhost. We will test the localhost ports for the connections being established.

We need to ensure that the existing features are not broken by new Docker containers being added, but due to the nature of the hardcoded port values, unless the new addition is misconfigured, it will run as intended.

5.5 Acceptance Testing

Our client has been working with us throughout the semester to ensure that we are using the correct tools in order to complete the tasks. Building the correct framework out of the correct libraries is critical to getting the job done. We tested whether the requirements are met to the client's satisfaction and see if the pre planned attacks fulfill the planned goals.

Our tests ensure that our product can continue to be developed and help users determine the security of their power grids, regardless of their system specifications or tech stack. For compliance with the project, we must make sure that all parts of the system are able to function individually and communicate with each other - which leads us to a proper simulation. We can test this through a myriad of ways and present it to the user when a test is successful at affecting the grid. For example, when a certain test is established the diagrams produced by PandaPower, OpenDSS, and Open DER will adjust to those conditions so we see changes in the model with line thickness (shown on the first figure) and the data results on each individual lines (shown on the second figure).




```
Values for:  LINE.L(R:UDT10535-UHS7_1247)
Bus1= udt10535-uhs7_1247x.1.2.3
Bus2= udt10535.1.2.3

Distance from Meter:   0.000 kM (  0.000 mi)
Power Flow:          3031.622 kW + j  509.173 kvar

Phase Currents:       138.211 123.168 156.702 Amps
Phase Voltages(L-N):    7.361   7.347   7.358 kV
Per Unit Voltages:      1.022   1.021   1.022

Losses for this Element:
TotalLosses:   22.486 kW + j  33.612 kvar

Number of Customers: 0
Total Customers downline: 0
```

5.6 Results

Our testing ensures that upon creation or composing - our project automatically tests itself to ensure that it will be set up for success and will be able to serve those using it. This is done by running the pytest script in the Dockerfile so that the program will exit upon failure. This results in the user not even knowing that testing is taking place unless something is going wrong, in which it provides useful information in which Docker instance is the root of the problem.

6 Implementation

6.1 Functionality

Frontend Functionality:

- Ability to run power grid simulations with a gamut of different attacks to be set upon the grid.
- Automatic and manual downloading features to locally save the output of the simulation.
- Output is placed within a zip file.
- Archive mode to check example output of each attack on a grid.
- EV mode to view examples of electric vehicle loads.
- Updating graphs to show the user the output of the graph.
- Carousel to view different aspects of the graph over time.

Grid Functionality:

- Grid can simulate a power flow over a 24 hour time span.
- Grid can handle multiple load types.
- Properly transfers data between the transmission and distribution grid.
- Limitation: no variable load, no EV integration, only works with Santa Fe load.

6.2 Notes

On the topic of attacks:

For attack implementation, we created copies of the existing SantaFe distribution model code and modified values to simulate lines tripping. Due to time limitations, we were unable to get all work implemented into the final design.

On the topic of Docker Containers:

A useful thing for debugging the docker containers is to open them in a shell (which can be done either from the command line of the Docker Extension in Visual Studio Code (would highly recommend doing for non-technical users)). Opening the Docker in a bash shell allows the users to check what it is seeing in its application directory (is located at root and its name can be checked in the docker-compose file).

On the topic of the power grid:

We are simulating the power grid at intervals of 1 hour. While this is not ideal for many applications - we believe that this strikes the best balance between performance of the simulation and giving good information to the user.

7 Professionalism

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

Area of Responsibility	**Definition**	**NSPE Canon**	**IEEE**
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.	Point 6 states that we must not only maintain but also improve our technical competence, and only complete tasks if qualified by training, experience, or have disclosed all limitations.
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	Point 5 states that we must be honest and realistic in our claims and estimates, and adds that we should fairly credit the contributions of others.
Communication Honesty	Report work truthfully, without deception, and understandable to stakeholders	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.	Point 9 states that we should avoid injuring others' property, reputation, or employment with our actions, rumors, or any other forms of abuse.
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	Point 1 states that we should hold paramount the safety, health, and welfare of the public .
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	Point 5 includes that we should fairly credit the contributions of others.

Sustainability	Protect environment and natural resources locally and globally.		Point 1 states that we should comply with ethical design and sustainable practices, not only to the environment, but also to the public and their privacy.
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.	Point 2 states that we should improve the understanding of individuals and society of the capabilities and implications of new technologies, including intelligent systems.

7.2 Project Specific Professional Responsibility Areas

Responsibility	Application	Performance
Work Competence	Yes	Medium
Financial Responsibility	No	N/A
Communication Honesty	Yes	High
Health, Safety, Well-Being	Yes	High
Property Ownership	No	N/A
Sustainability	Yes	High
Social Responsibility	Yes	High

- Work Competence
    - Our project must not be deceptive in its topics, we must all have knowledge about what types of grids we are working with and how the code connects them.
    - A lot of our project so far has been learning about the project itself, hence this only deserves a medium.
- Financial Responsibility

    - This project uses a provided testbed for virtualization using VMWare. There are no costs besides the processing resources used by this project, hence it is not applicable.

- Communication Honesty
    - While we do not have any stakeholders, we are reporting to our advisor and client every week on what work we have performed and what will be done in the future. We also get guidance on how to better achieve the goal of the project. Therefore, we must do well in our communication with the client for the project to be successful.
    - We have done a good job with our weekly reports to our client and advisor as well as meeting with him outside of the regularly scheduled meeting time.
- Health, Safety, Well-Being
    - The power grid has many different uses when it comes to medicine, safety, and modern civilization. Our project may be used to help develop newer power grids or modify existing ones in order to make them more secure. We must be understanding and serious about the changes this could cause and make sure we emphasize that this project is not the be-all-end-all when it comes to power grid security checking.
    - We have not really had to show this yet, as we have not gotten to the stage of full product implementation, but we are aware of the weight this holds.
- Property Ownership
    - Our project does not have any property required to own, therefore this does not apply.
- Sustainability
    - In relation to the Financial Responsibility of the project, we have to have the sustainability of the project in mind to make sure that no simulation is kept running on the virtual machines and wasting needless amounts of energy.
    - We have been keeping tabs on the virtual machines and making sure that there are no active running programs when we are not working.
- Social Responsibility
    - Similar to Health, Safety, Well-Being, we must be cognizant that the product of this project will be used to help develop power grids. This means that we must do our best to be mindful of the impact that this will have on power grid developments and the social impact and weight that holds.
    - We have not really had to show this yet, as we have not gotten to the stage of full product implementation, but we are aware of the weight this holds.

7.3 Most Applicable Professional Responsibility Area

Health, Safety, Well-Being and Social Responsibility

This is because the purpose of our project is to benefit the health, safety, and well being of society. Our project's goal is to provide a simulator that can help protect the current power grid against

cyber attacks, as well as test potential future designs. Health, safety, well-being and social responsibility is the backbone of our project.

8 Closing Material

8.1 General Reflection

We are very proud of the work that we have achieved on this project. We can all agree that the tools we are using have been challenging for us to learn and that we gave it our all. Throughout the project, we ran into many roadblocks but we were able to overcome them as a team. We combined very different majors and were able to work concurrently to achieve the goals set by our client.

9 Appendices

Appendix I - Operation Manual

To run the CyHelics software, download the software off of our git repository (https://git.ece.iastate.edu/sd/sdmay24-28) and run the command “docker compose up –build”. This will begin the dockerized application environment. This consists of 4 applications being run on ports 8000, 8001, 8002, 8003 on the localhost. Port 8000 is the frontend application, please enter this into the web browser at “localhost:8000/”. The application at port 8001 hosts the HELICS application, port 8002 is for the transmission application, and port 8003 hosts the distribution application. To see the configuration for CyHelics, please take a look at the docker-compose.yaml file for more details.

The frontend consists of two pages, a main control panel and an archive page. The main page is how one interacts with the live components of the CyHelics software. The Archive mode is a way to view previous runs of CyHelics. This currently is just stagnant data, however, in the future it would be simple to store the data to a database.

Figure A1

Main Page

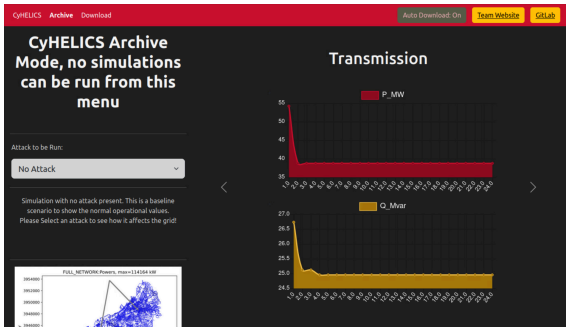


Figure A2

Archive Page

Main Page

The main page the the home page and root directory (/) of the website located at localhost. Please enter “localhost:8000/” into the url of the web browser to access the frontend of CyHelics. Once on the main page, you are able to select an attack from the dropdown menu.

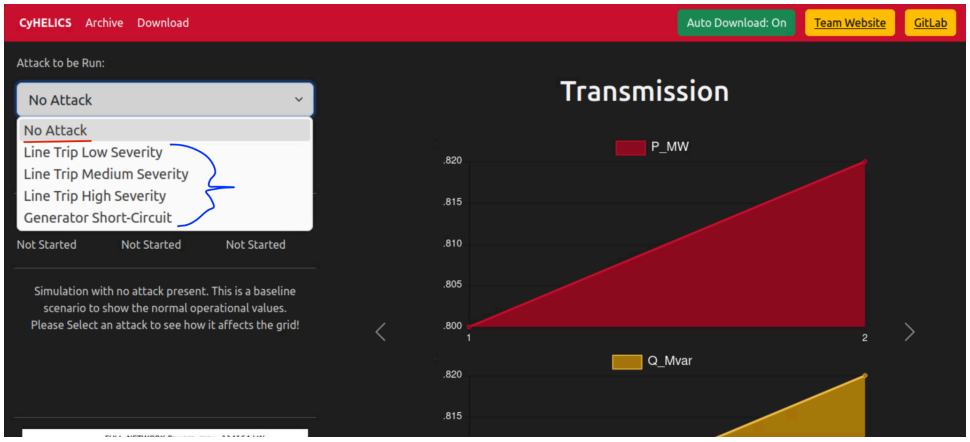


Figure A3

Attack Selection

Once an attack is selected, you can hit the “Run Simulation” button in order to start the application. At this point on, hitting the “Run Simulation” button will no longer work until the simulation is complete.

Figure A4

Main Buttons

Other things to check out are all located on the header. Downloading is accessible at any point, even during a simulation run. You can also toggle the “Auto Download” feature on and off. You can also check out our website, GitLab, and the archive mode while you wait for the simulation to complete. Please note the tags underlined in blue seen in Figure 5. These tags are for the status of

the simulation, they will update automatically throughout the run of the simulation. The graphs should show only having 1 hour within 30 seconds of starting the simulation.

Figure A5

Running Indicators

Once the “Started!” tags have appeared, the graphs will update every 30 seconds until the simulation finishes. The simulation takes roughly 20 minutes to complete on the current hardware it is running on, but this may change. The Distribution grid will be the first to finish. Helics and the Transmission will then exit and the “Started!” tag will be replaced with “Complete”, in which the auto download will download a zip file of all of the comma separated values used for the graphs. At this point, the graphs will no longer update until a new simulation is run.

Archive Page

The archive page is much simpler than the main page in that there are no calls to the other parts of the application (on ports 8001, 8002, 8003). Once you select an attack, the graphs will update to a stagnant graph. Feel free to explore this page once you are running a simulation.

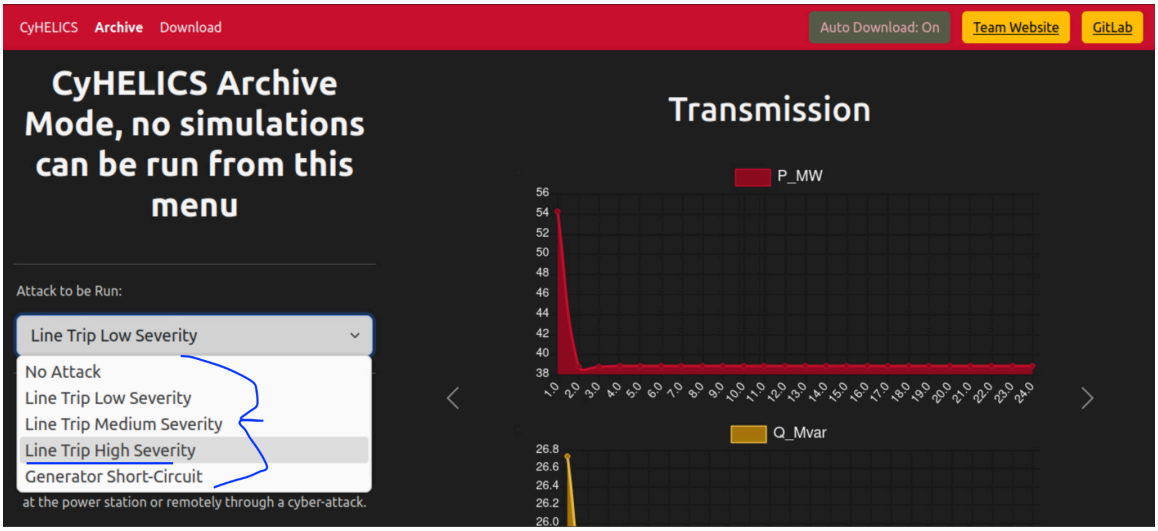


Figure A6

Archive Selection

Testing

To run the tests, use the “docker compose up –build” command and enter into either the frontend, dss, or pandapower docker image in a bash shell. This can typically be done from either in the VSCode Docker Extension or the command line. Once in the shell, run the command “python3 -m pytest” to run the test suite.

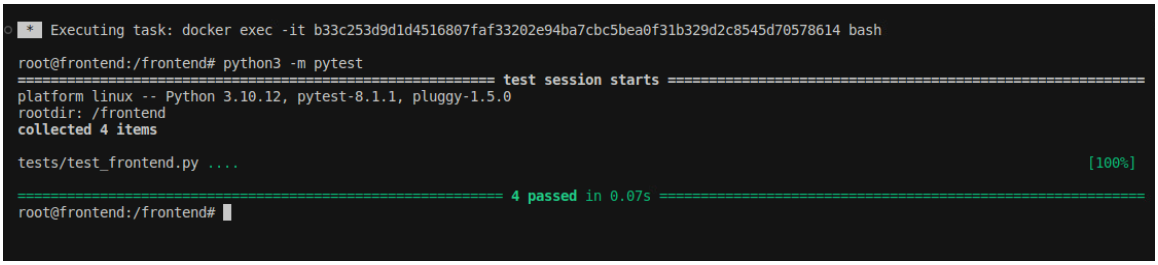


Figure A7

Frontend PyTests

These tests ensure that the endpoints used by the interconnecting docker instances are open and working properly. These tests are for system, regression, and end-to-end testing.

Appendix II - Learnings

Justin Templeton - I found this project really interesting, challenging, and frustrating at the same time due to the learning curve of working in a more DevOps role. I did really enjoy working on the web application side of the project, in which Computer Science 319 has been an invaluable class for that. In all, I learned Docker like it was the back of my hand, and it is now serving me well in personal projects or for other classes - so that at least is already paying off.

Kaya Zdan - This project was a very different project from anything else I had done at Iowa State. I hadn't worked much with simulation software, much less co-simulating multiple simulations together before. I was really glad for the first semester to have the time to learn all of this, since there was a very large learning curve for me. I learned a lot about transmission and distribution grids, using pandapower, dss-python, and HELICS simulation softwares.

Tyler Atkinson - While working on this project, I was also taking CprE539 to further understand cyber security in the electrical grid. Using both this project and that class, I learned the basics of how an electric grid works, possible attack vectors and outcomes of the attacks, and how companies develop defense strategies for the grid. Overall, this project was super interesting but also super frustrating at times as there were so many different problems that stalled progress at times.

Zach Hirst - This project was much larger and different than anything I had worked on before. I was able to draw some of the learning strategies I gathered from the Cyber core classes (230, 231, 331) I took at Iowa State but all of the frameworks were new to me. I enjoyed and am thankful for having the time in the first semester to learn basics for docker and how co-simulation programs function. I know that these programs are used across many industries and will have a great impact on my career now that I have a basic understanding of how they work. There were aspects of the project that I was able to deepen my understanding of certain technologies. Those technologies were mainly web development, git, and python. I was able to do smaller more focused tasks in each of these three technologies that I have already transferred the knowledge to my personal projects.

Matthew Nevin - This project allowed me to display the knowledge that I have gained throughout my years as an undergraduate student at Iowa State. I got to rope in skills such as power system analysis (EE 456, EE 457), coding (EE 285, COMS 207), problem solving, and teamwork in order to design a solution to this problem in a group setting. I learned a great deal about how electric vehicles can impact the power grid on large scales. I also learned about how charging infrastructure can be set up and the important variables that can play a role in designing this infrastructure. Through the use of softwares such as Pandapower and HELICS that can simulate different aspects of the power grid as a whole I was able to see how a power grid works on a large scale. Working on CyHELICS allowed me to further explore the transmission, distribution, and load aspects of the power grid which gave me a great understanding of the infrastructure as a whole.

Tommy Keeshan- I found that this project used many different learnings I had been taught throughout my power system classes. Such as knowing different equipment on a grid, how to solve for certain variables, and what values to expect. I had a learning curve with understanding the coding side of using these analysis programs, especially on a large scale. Using documentation with little examples made it challenging for me but did force me to work with others who had a stronger

skill set in coding. I also was given the chance to be a helping hand to others who did not have any understanding of power or electricity and hopefully taught them a thing or two about my major.

Appendix III - Code

For the code structure of our project, the Git Repository contains the following folders. The main four folders used in the “docker compose up –build” command are as follows:

- frontend
- helics
- pandapower
- dss

More folders can be added to Docker Compose by editing the docker-compose.yaml file.

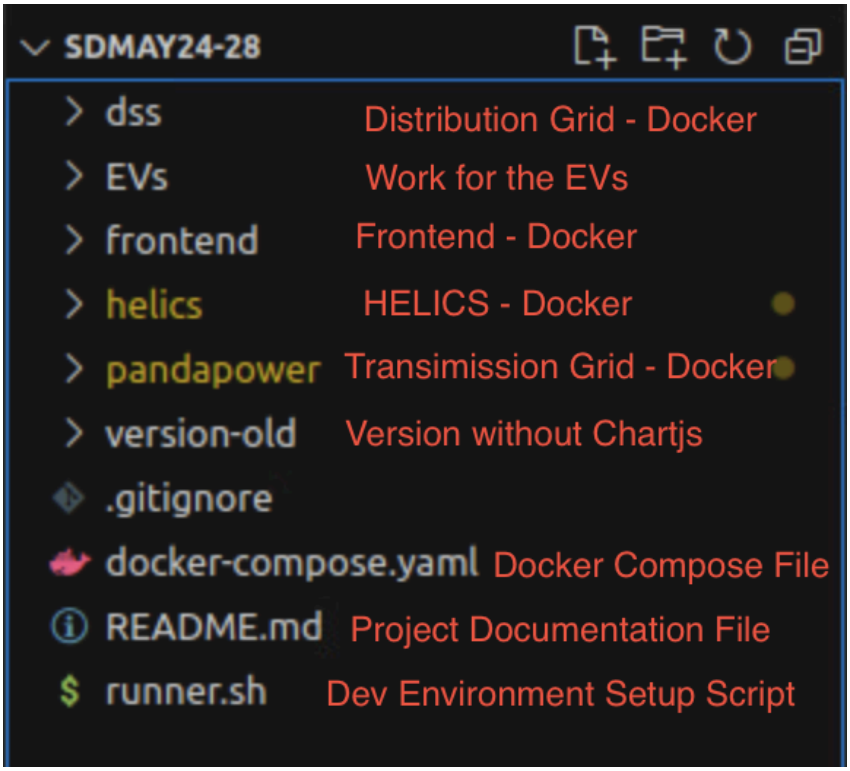


Figure A8

Root Directory Structure

The EVs directory contains all of the files that fulfill the electric vehicle component to our project. While it is not attached to the rest of the project currently, it is working independently - so this is the prime target for future development.

The version-old folder is for the previously working version in-case a rollback is needed during development.

The other files are used either for documentation, setting up a dev environment, or for docker purposes.

Appendix IV - Attack Research and Future Work

List of papers used in researching attack outcomes and historical attacks on electric grids: [13], [14], [15], [16], [17], [18], [19].

Based on the above research papers and the constraints that came with using a dockerized environment, as well as the libraries we were using for the simulation, we were able to come up with 3 different attack outcomes that we could feasibly implement. The attack outcomes are line tripping, generator short circuit, and load shedding.

Due to time constraints we weren't able to get everything implemented that we had initially planned on. We were only able to get the line tripping implemented and working. Our future work included implementing the other 2 attacks.

One of the attacks is simulating a generator short-circuit in Pandapower. According to Pandapower documentation [12], there is a function built into the library that allows for the simulation of a generator short-circuiting. Our implementation plan was to modify the pandapower code to have this function run when specified to run this specific scenario.

The other attack is to emulate a data integrity attack by causing load shedding. According to Pandapower documentation [12], there is a function built into the library that allows for State Estimation, which is an attack vector for data integrity attacks that was found using the research stated above. By trying to falsify State Estimation data, we could get the transmission model to cause load shedding leading to less power on the grid than is needed by the distribution model.

Appendix V - Docker Future Work

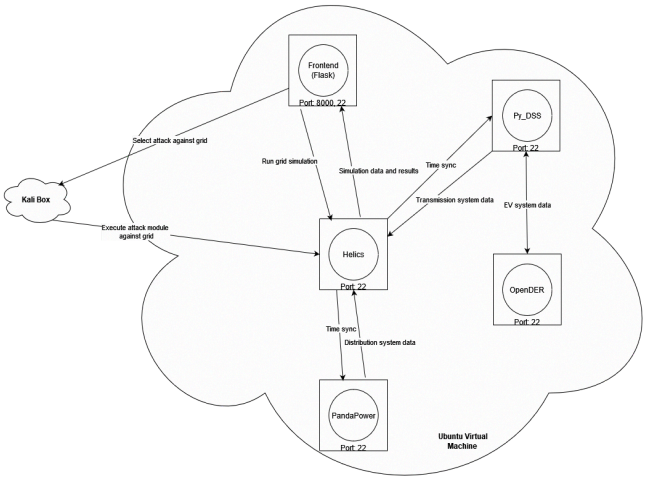


Figure A9

Initial Design

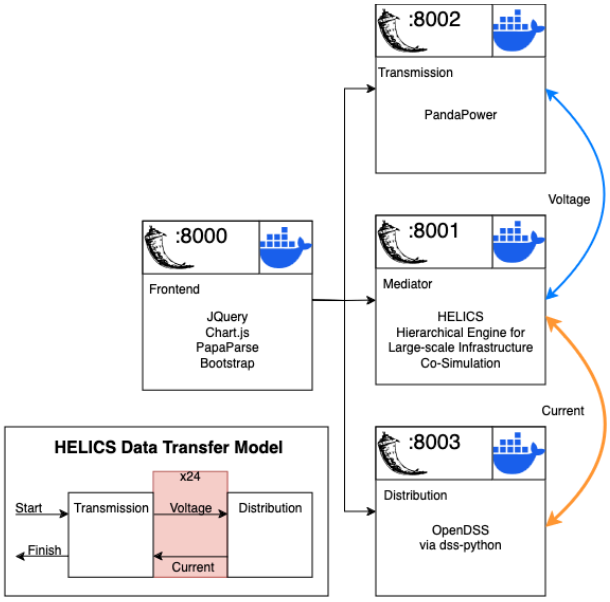


Figure A10

Current Design

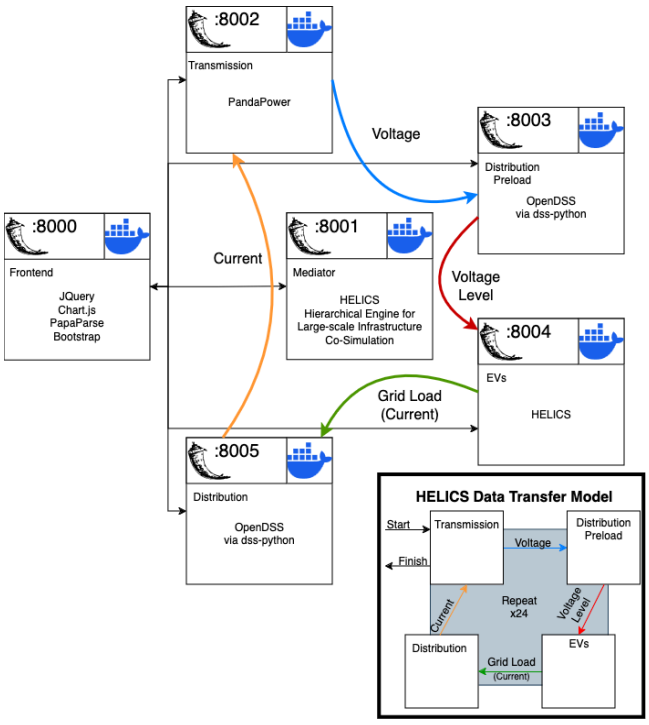


Figure A11

Future Design

As the Docker environments encapsulate any additional parts of the project, it is easy to expand upon in the docker compose file. Our current design is utilizing the design seen in Figure 2. However, the addition of the EV load can either be done using the same design with a complex encapsulation of the EVs into the Distribution stage or it can be split out into 2 extra Docker containers like in Figure 3. While this is a more complex design decision from the overall design standpoint, it makes the workflow much easier to follow and is more realistic when compared to a design like in Figure 2 due to electric vehicle charging stations thinking independently without regard to the outside world (i.e. the charging station stops when the car is done charging).

Appendix VI - References

- Docker:

[1] T. Donohue, "How To Communicate Between Docker Containers," *Tutorial Works*, Nov. 06, 2020. https://www.tutorialworks.com/container-networking/ (accessed Oct. 11, 2023).

[2] "How To Share Data between Docker Containers | DigitalOcean," *www.digitalocean.com*. https://www.digitalocean.com/community/tutorials/how-to-share-data-between-docker-containers (accessed Oct. 09, 2023).

- HELICS:

[3] "HELICS documentation — HELICS documentation," docs.helics.org. https://docs.helics.org/en/latest/index.html (accessed Sept. 18, 2023).

[4] "GMLC-TDC/HELICS," GitHub, Nov. 14, 2023. https://github.com/GMLC-TDC/HELICS (accessed Sept. 18, 2023).

[5] "HELICS-Examples," GitHub, Oct. 31, 2023. https://github.com/GMLC-TDC/HELICS-Examples (accessed Sept. 19, 2023).

[6] "Docker," hub.docker.com. https://hub.docker.com/r/HELICS/HELICS# (accessed Oct. 04, 2023).

- OpenDSS

[7] "DSS-Python's API reference — dss_python 0.14.0.dev documentation," dss-extensions.org. https://dss-extensions.org/dss_python/ (accessed Nov. 11, 2023).

[8] "DSS-Python: Extended bindings for an alternative implementation of EPRI's OpenDSS," GitHub, Nov. 27, 2023. https://github.com/dss-extensions/dss_python (accessed Nov. 11, 2023).

- OpenDER

[9] "epri-dev/OpenDER," GitHub, Oct. 29, 2023. https://github.com/epri-dev/opender (accessed Nov. 15, 2023).

[10] Y. M. Anandan Wei Ren, Paulo Radatz, Jithendar, "opender: Open-source Distributed Energy Resources (DER) Model that represents IEEE Standard 1547-2018 requirements for steady-state and dynamic analyses," PyPI. https://pypi.org/project/opender/ (accessed Nov. 15, 2023).

[11] "EPRI Home," www.epri.com. https://www.epri.com/opender (accessed Nov. 15, 2023).

- PandaPower

[12] "pandapower," pandapower. https://www.pandapower.org/ (accessed Sept. 15, 2023).

[13] L. T. Scheidler Alexander, "pandapower: An easy to use open source tool for power system modeling, analysis and optimization with a high degree of automation.," PyPI. https://pypi.org/project/pandapower/ (accessed Sept. 15, 2023).

[14] "e2nIEE/pandapower," GitHub, Dec. 02, 2023. https://github.com/e2nIEE/pandapower (accessed Sept. 15, 2023).

- Attacks

[13] Roman Bolgaryn, Gourab Banerjee, et. al, “Open Source Simulation Software pandapower and pandapipes: Recent Developments”. https://publica-rest.fraunhofer.de/server/api/core/bitstreams/72a583a8-7204-4f87-bc9d-10760e07701a/content

[14] Doney Abraham, Sule Yuldirim Yayilgan, et. al, “Cyber Attack Simulation and Detection in Digital Substation”. https://ieeexplore.ieee.org/document/10176955

[15] Vetrivel Subramaniam Rajkumar, Alexandru Stefanov, et. al, “Cyber Attacks on Power Grids: Causes and Propagation of Cascading Failures”. https://ieeexplore.ieee.org/document/10256104

[16] Ashutosh Dutta, Sumit Purohit, et. al, “Cyber Attack Sequences Generation for Electric Power Grid”. https://ieeexplore.ieee.org/document/9770105

[17] R. Liu, A. Srivastava, “Integrated simulation to analyze the impact of cyber-attacks on the power grid”. https://ieeexplore.ieee.org/document/7115395

[18] Siddharth Sridhar, Adam Hahn, Manimaran Govindarasu, “Cyber–Physical System Security for the Electric Power Grid”. https://ieeexplore.ieee.org/document/6032699

[19] Robert Lee, Michael Assante, Tim Conway, “Analysis of the Cyber Attack on the Ukrainian Power Grid”. https://nsarchive.gwu.edu/sites/default/files/documents/3891751/SANS-and-Electricity-Information-Sharing-and.pdf