6-13-2020

# CentriFi: A CentralizedWireless Access Point Management Platform

Andreas Anderhub

Zac Wilson

**SANTA CLARA UNIVERSITY**
**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**

Date: June 13, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

**Andreas Anderhub**
**Zac Wilson**

ENTITLED

# CentriFi: A Centralized Wireless Access Point Management Platform

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

B. Dezfouli

Thesis Advisor

DocuSigned by:
Nam Ling
96CE94A1A83A48A...

Department Chair

# CentriFi: A Centralized Wireless Access Point Management Platform

by

Andreas Anderhub
Zac Wilson

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 13, 2020

# CentriFi: A Centralized Wireless Access Point Management Platform

Andreas Anderhub
Zac Wilson

Computer Science and Engineering Department
Santa Clara University
June 13, 2020

ABSTRACT

With the ubiquity of wireless end-devices, more strain is placed on standard network deployment architectures. Mesh networks have started to rise in popularity in order to meet the needs of modern wireless networks. However, the existent solutions for deploying and centrally configuring mesh networks leave much to be desired, as most are either too expensive or too cumbersome. This paper showcases a solution to this problem, *CentriFi*—an open source platform, built to run on OpenWrt access points, providing a quick and easy way to set up and configure mesh networks in a central location using the 802.11s standard. CentriFi provides a web-based front-end for configuring the most crucial settings. Further, the system allows for greater expandability by providing a platform in which other configuration feature can be added by the open-source community in the future.

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation and Background

Internet connection is now widely considered a necessity for modern life. As more and more devices become integrated with internet connectivity, the strain on almost every network is increasing. There are already tens of billions of IoT devices connected to networks worldwide [1]. Often thousands of devices are expected to connect to the same wireless network, such as in large medical organizations where supporting tens of thousands of healthcare devices and computers on the same network is necessary [2]. Even smaller networks for homes or schools can expect to support hundreds of devices.

To simplify installation, improve coverage, and increase throughput, *multi access point networks* and, in particular, *mesh networks* have become useful in developing network structures capable of supporting these types of needs [3]. Finding a more efficient, customizable option to map out these mesh networks would benefit everyone greatly. Hospitals could maintain more connected healthcare devices; schools could give students access to connected modern learning materials; homes could be equipped with better connected security and an increased number of Internet of Things (IoT) devices. With more connected devices in the same space comes the need for more access points, and more dense network deployments require a higher level of monitoring and control [4]. There is current research being done on developing secure routing protocols for mesh networks, such as in [5] and [6]. However, not much research has been done on software used to set up and configure these networks.

## 1.2 Current Solutions

With the solutions that currently exist, building dense networks capable of supporting a large numbers of devices are either too expensive for most users, not customizable enough, or both. The first-party networking software included with networking appliances are either limited, in the case of home routers, or prohibitively expensive, in the case of enterprise-grade equipment. For example, Cisco Meraki [7] and HPE Aruba [8] are very popular enterprise-grade networking systems that provide mesh networking and extensive customization of the network. These systems are quite easy to set up, and even allow users to manage them via web interfaces. However, the initial cost of these systems is in the thousands of dollars, and require yearly subscription fees. Furthermore, networking equipment with first-party software is only interoperable with that same brand of networking equipment. Netgear routers only work with other Netgear routers, Cisco Meraki routers only work with Cisco Meraki routers; if you buy into a Netgear system and decide you want to expand your system, you cannot do so with Linksys hardware.

Some of the existing open-source solutions to the problem are OpenWrt [9] and OpenWISP [10]. While these systems are open-source software, they are difficult to understand and set up. Additionally, they lack easily understood documentation. In our experience with OpenWrt, the documentation on setting up a mesh network [11] was difficult to follow, and lacking in necessary information. To gather this information, scraping internet forums or trial and error were the only methods to get this system working.

## 1.3 Our Proposal

In this paper, we present a management platform—named *CentriFi*—that makes creating robust and customizable networks both intuitive and easy to set up. This solution is based on the OpenWrt platform which can be installed onto many existing router equipment for wide-scale compatibility. This solution allows users to intuitively customize and configure the network from a web browser. Users have access to a per-device configuration to be able to map connected devices to a desired access point, and build a network according to their needs. This allows for mapping of very dense network deployments that contain a high number of access points and a high number of devices. Additionally, CentriFi provides basic network traffic monitoring statistics.

CentriFi is not only open-source and free to use, but also cross-compatible with many type of hardware supported by OpenWrt, provided that this hardware supports 802.11s mesh networking. This cross-compatibility allows for users to expand their network with new hardware from different vendors without the need to replace existing hardware. The resulting reduced equipment cost is especially useful for small businesses and nonprofits where new networking equipment may be low on the priority list.

Furthermore, CentriFi is bundled with an extremely simple installation process. The installer is a Python 3 script that can be run on any computer. The access points are connected to the computer one at a time, and then the script automatically downloads all required software to the access points and configures them. After the installation, the web-based manager starts running on the designated master router, and the access points can be configured further.

Due to the easy setup, low initial cost, and ability to easily add new hardware if the network needs to be expanded, there are a wide variety of potential applications for CentriFi. It can be used by home users who are looking for a inexpensive solution to easily get a mesh network running for all of their home devices, including things like smart home security cameras, lights, voice assistants, etc. Additionally, it can be used in low-income or non-profit hospitals, schools, and small entrepreneurial businesses as a inexpensive way to get internet access throughout their building(s). CentriFi provides an inexpensive, easy, and extendable way to configure and manage the network for these applications. It can run on relatively inexpensive hardware for basic installations, or more expensive hardware for more expansive network solutions. In this way, CentriFi is extremely adaptable to the needs of the location and use case. As it is now an open-source project, available for download on GitHub, so it can also be customized for more specific needs, system development, and educational activities. As such, CentriFi has been tested in a home networking environment. Routers from both Netgear and Linksys were tested together along with end-devices including laptops, smart phones, and other connected devices.

# Chapter 2

# Requirements

We set out to make CentriFi an open-source access point management software for people in home and non-profit environments where enterprise solutions are too cost-prohibitive. As such, we developed a set of requirements, ranging from required features to desired qualities.

## 2.1 System Requirements

The first requirement is *mesh networking* capability. Given that mesh networking is such a crucial capability for the wide area deployments needed in the growing wireless world, this feature is at the top of the requirement list. Second, since we target users with cost-restrictive budgets, it aims to *lower the cost* of hardware necessary to run this system. This is accomplished through providing a system that works on a wide array of hardware from different manufacturers. Thus, users can upgrade existing networks with greater flexibility, not restricted to one hardware manufacturer, increasing the lifespan of their existing hardware. Third, CentriFi is *easy to install* and set up. The problem with current open-source solutions is the difficulty of setup, especially when it comes to mesh networking. This difficulty deters potential users who need a simpler solution. Fourth, in addition to being *easy to setup*, it is also *easy-to-use and configure*. Once the network is running, maintaining and customizing the network is easily performed. Fifth, it is *expandable*; the open-source community can easily add features as the need arises. Additionally, the features mentioned above are just a starting point.

CentriFi is feature-rich enough to provide users with customization for configuring their network while avoiding overwhelming complexity. The following features provide the balance: configuring whole network settings (WiFi SSID, password, and security protocol), configuring wireless channels on each access points, viewing network bandwidth statistics, and provisioning which access point end devices on the network can connect to.

Finally, all of the above features would be useless without security and reliability. Therefore, CentriFi supports the new WPA3 wireless security protocol on both the mesh backbone and main WiFi network. Further, communication between access points is performed securely over SSH.

# Chapter 3

# System Overview and Design Rationale

## 3.1 System Overview

In order to meet the ease of use requirements, CentriFi is bundled with a Python 3 installation script. The Python script installs all necessary components on the master and slave access points, configures the mesh, and starts the web server. Python is used because of its compatibility across computer operating systems and ease of use.

For feasibility reasons, this setup script has only been tested on routers with a Qualcomm chipsets supported by the ”ath10k” wireless driver. However, even with this restriction, CentriFi is capable of running on a wide range of modern, consumer routers from multiple manufactures. In practice, we will use Netgear and Linksys routers for testing, as shown in Section 5.1. In theory, this support covers routers from brands like Netgear, Linksys, TP-Link, D-Link, Zyxel, and more. Given the survey of our testing, we can say with a relatively high degree of certainty that CentriFi will run on every router with a chipset that supports the 802.11s mesh protocol.

Before beginning the setup, each router is first flashed with OpenWrt. After this, each router is connected to a modem via Ethernet and then connected to the computer running the script, one at a time, following the prompts in the script until all routers are configured. This process is much simpler than the individual configuration of each router that is necessary for other open-source solutions. During this process, the install script configures the hardware on the network with the following software architecture as can be seen in Figure 3.1.

Each router runs the OpenWrt operating system. OpenWrt is a free, open-source, operating system for routers that allows for users to customize their routers beyond what the manufacture allows. It also provides a base for additional software. Although this operating system alone is too cumbersome and lacks proper documentation for mesh networking, it nonetheless provides the layer of abstraction needed for software to work across the firmware of many different routers from many different manufacturers. If a user wants to perform some network configuration outside of CentriFi’s scope, OpenWrt allows them to do so through the Luci (web interface) or Command Line Interfaces (CLIs).

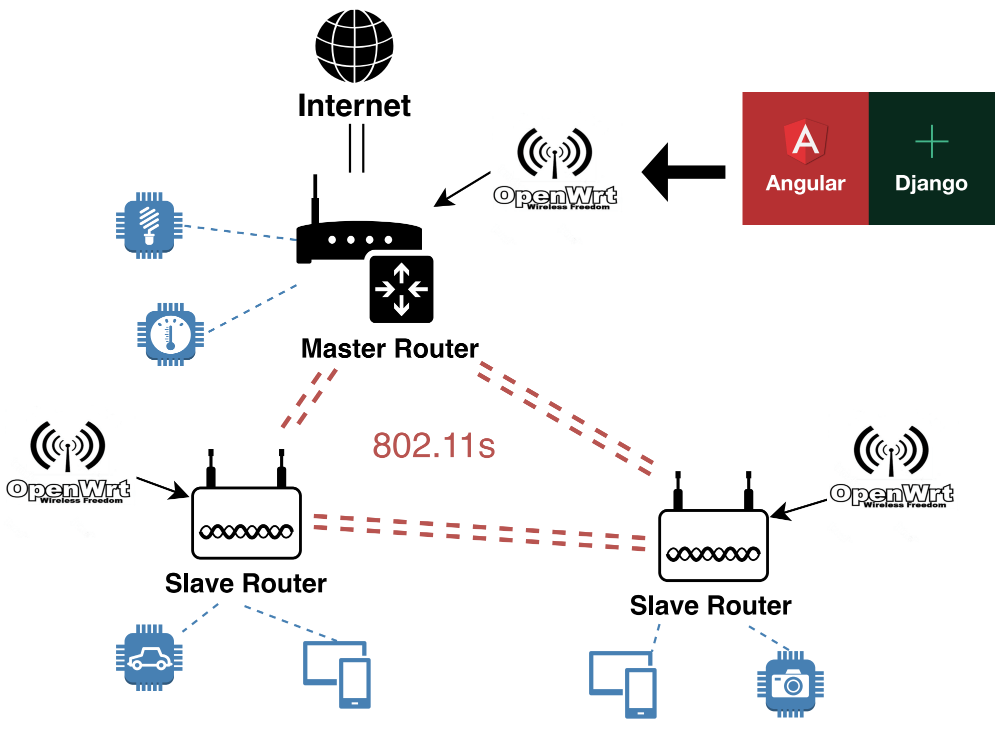


Figure 3.1: The Architecture of a CentriFi System

## 3.2 Design Rationale

We chose OpenWrt primarily because of its wide hardware support. It also supports the range of software necessary for CentriFi's operation. And, as of version 19.07, OpenWrt provides WPA3 security as well. We chose OpenWrt over competitors like Tomato [12] and DD-WRT [13] for a few reasons. The breadth of router support and community updates that OpenWrt provides greatly eclipses Tomato's offerings. While DD-WRT does provide somewhat comparable router support to OpenWrt, DD-WRT does not provide the base on which to build out the rest of the system. Additionally, it is less frequently updated than OpenWrt. Furthermore, neither DD-WRT nor Tomato support WPA3 at the time of writing.

On top of that operating system, CentriFi configures the mesh network with one master router connected to the Wide Area Network (WAN) via Ethernet, and all the slave routers connected via a wireless mesh to the master. One transceiver on each router is used solely for the purpose of providing the mesh backbone, linking the network, while another transceiver serves to create the wireless network available for client devices.

For the mesh network, we chose to use the IEEE standard, 802.11s [14]. This protocol allows for the creation of a seamless wireless internet connection where the routing of packets between routers in the mesh can be automatically handled, making the creation of a wireless WLAN simple and standardized. We chose 802.11s because it has already been implemented on the OpenWrt platform and is the most widely compatible protocol with different hardware devices [11].

Once the mesh and WiFi network is setup, configuration is handled via a master-slave architecture. The master router runs a Django server that provides a web application for network configuration. We chose Django because it is supported by OpenWrt's native package system, and it is based on Python 3. Given OpenWrt's support of Python 3, this server does not need to be recompiled for different processor architectures, since the Python interpreter is already compiled for each processor architecture supported by OpenWrt. This removes the need for a resource demanding compiler, and allows CentriFi to work on a variety of hardware as it is intended to do. We utilized Django over Flask (the other major Python 3 web server) primarily because of its large built-in feature set when compared to Flask. Not only did this speed up development, but it allowed us to test a worst-case scenario server install size and also leaves the door open for future developments.

The Django server executes settings or requests made by the user via the web application. From there, any OpenWrt setting can be changed via the server if desired. If these settings affect the other slave routers, the server then propagates these settings to other routers securely through the SSH protocol. We chose SSH in order to keep the footprint on client devices as small as possible. Since SSH is built into OpenWrt, no additional software is necessary on the slave routers.

The served web application itself was created using the Angular 9 framework. We primarily utilized Angular because our team had experience with it, and it facilitates quicker development, easy expandability, and clean coding practices. It also provides access to Google's material design form control, which is particularly useful for creating the clean and modern form UI that CentriFi presents.

The web application is accessible by any device on the network with a web browser. It resizes UI elements according to mobile or desktop screen real estate needs. This interface provides the user with an easy-to-use interface for configuring the whole network (WiFi SSID, password, and security protocol), configuring wireless channels on each access points, viewing network bandwidth statistics, and provisioning which access point end devices on the network can connect to.

Furthermore, the built-in features of the web interface can be easily expanded. Since CentriFi is being released as an open-source project, it can be expanded to include new configuration options as needed.

# Chapter 4

# Design and Implementation

The platform is comprised of three main parts: the installation script, the web interface front-end, and the web interface back-end.

## 4.1 Installation Script

The installation script was created using Python and the Paramiko module to work as a simple installer, requiring only that the devices be plugged into the computer running the installer and the modem. The installer then uses Paramiko to run commands on the connected device for all the necessary setup of CentriFi.

The setup script consists of multiple functions that implement the necessary features of the installation process. First, it updates the OpenWrt package installer, opkg, and then handles any firmware changes. Then the install script installs necessary packages, configures the mesh network, and reboots the access point. This process first occurs for the master router, and then the slave routers as they are sequentially plugged in .

Before the installation script is run, each router's firmware must be flashed to at least version 19.07 of OpenWrt, which provides important support for WPA3 security. Upon completion of this flash, the WAN port of each router must be connected to either a modem or another router that is not to be used in the mesh.

Then, the installation process can begin with the install script. First, the master router is connected to the computer via Ethernet, as well as being connected to the WAN. After completion, the master router is unplugged, and the slave routers follow the same process one by one. The install script then uses the Paramiko module to connect to the access point over SSH, to the default IP address ”192.168.1.1”. The user name for this connection is ”root” with no corresponding password, and this module automatically logs in. The installer then prompts for an admin password. The password is set on the router through the `passwd` command. After the password is set, the basic wireless network software, `wpad-basic`, is replaced with the mesh capable one, `wpad-mesh` (note: the full version of `wpad` is stated to work, but in our testing only `wpad-mesh` was successful). In order to do this, these three commands are executed on the router:

```
opkg update
opkg remove wpad-basic
opkg install wpad-mesh
```

In testing router support of 802.11s, we ran the `iw list` command to see a router's reported hardware operating modes and checked for the presence of `mesh_point`. However, even on routers that listed that support, it was not always supported in reality. For example, Marvell chipsets, like that on the Linksys WRT3200ACM, do not work with 802.11s. The OpenWrt community has identified this issue as well [15]. For this reason, we will perform all our testing on Qualcommm QCA9xxx chipsets that use ”ath10K” firmware.

Furthermore, we found that ath10k Qualcomm chipsets do not work with the default drivers installed on OpenWrt 19.07, as OpenWrt now uses an open-source variant of the drivers managed by Candela Technologies and denoted by a ”-ct” at the end of the driver name. To fix this problem the current ath10k ”ct” drivers are replaced with the original, non-Candela Technologies, ones by running the following commands (where `xx` corresponds to the firmware installed, listed by the first command):

```
opkg list-installed
opkg remove ath10k-firmware-qca98xx-ct
opkg install ath10k-firmware-qca98xx
```

The original chipset firmware is now installed. Based on our minimal testing, the stability and feature set of the original firmware is comparable to that of the Candela technologies versions.

After the firmware has been swapped, the necessary packages are installed. On the master router, the packages installed are: python3, python3-pip, vnstat, git-http, nmap, and hostapd-utils. These packages are necessary for various features of CentriFi. Pexpect [16] and Django are also installed on the master router, though these are installed using pip3 instead of opkg, to get more up-to-date versions. On the slave routers, only vnstat and hostapd-utils are installed. On the master router, the CentriFi repository is pulled from GitHub.

Next, the IP address of each slave router is changed to a unique one. The master router is assigned ”192.168.1.1”, but the slave routers are assigned sequential IPs: ”192.168.1.2”, ”192.168.1.3”, through ”192.168.1.31”. Further configuration changes are necessary in order for the slave routers to redirect their traffic to the master router, via setting the master router as the IPv4 gateway and DNS. To do so, the network config file is edited by the following five commands:

```
uci set network.lan.ipaddr='192.168.1.x'
uci set dhcp.lan.ra_management='1'
uci set network.lan.gateway='192.168.1.1'
uci add_list network.lan.dns='192.168.1.1'
uci set network.wan.ifname='eth1'
```

On the slave routers only, firewall, IPv6, and DNS are disabled. This prevents the network from having more than one router doing the same thing. It also ensures that the router with connection to the WAN is directing domains to the DNS server and handling IPv6 conversion instead of routers with unconnected WAN ports. To do so, the following six commands are issued to disable those features on the slave routers during the startup process:

```
/etc/init.d/dnsmasq disable
/etc/init.d/dnsmasq stop
/etc/init.d/odhcpd disable
/etc/init.d/odhcpd stop
/etc/init.d/firewall disable
/etc/init.d/firewall stop
```

In order to configure the mesh network, we use 802.11s which is built into the `wpad-mesh` package for OpenWrt. In the `etc/config` directory, there is the wireless configuration file `wireless`. This contains the configuration for the wireless devices on the access point. To set up the mesh network, one radio antenna is dedicated as a mesh ”backbone.”

For our implementation, we use any radio0 which supports 802.11ac on all tested routers. The following UCI commands configure the hardware radio for the mesh ”backbone”:

```
uci set wireless.radio0.type='mac80211'
uci set wireless.radio0.channel='161'
uci set wireless.radio0.htmode='VHT80'
uci set wireless.radio0.hwmode='11a'
uci set wireless.radio0.disabled='0'
```

As can be seen in the channel option of the wifi-device for radio0, the wifi hwmode which corresponds to wifi channel is set to 161. Based on our testing, channel 100, 165, and 163 do not work for mesh networking on one access point or another. Channel 161 is used as it is the first one to work for all our testing routers. Further testing needs to be done to determine a method for determining which channels are compatible with mesh networking on all devices in the network. Next, the channel option is set to 11a so that routers use 802.11ac as their ”backbone” and the option type is set to ’VHT80’ to ensure the channel width is 80MHz.

Next, the software settings for radio0 are configured on default_radio0 via the following UCI commands:

```
uci set wireless.default_radio0.device='radio0'
uci set wireless.default_radio0.network='lan'
uci set wireless.default_radio0.mode='mesh'
uci set wireless.default_radio0.mesh_id='centrifi'
uci set wireless.default_radio0.encryption='sae'
uci set wireless.default_radio0.key='randomKey'
uci set wireless.default_radio0.mesh_fwding='1'
uci set wireless.default_radio0.mesh_rssi_threshold='0'
```

Radio mode mesh sets the software mode to mesh. Mesh_id indicates the name of the mesh network that all mesh points connect to. Encryption is set to 'sae' in order for the mesh to be connected via WPA3 encryption. The key option is randomly generated at the start of the install script, and kept the same for all routers. `Mesh_fwding` set to 1 selects 802.11s to provide the routing protocol for network packets instead of another routing protocol. `Mesh_rssi_threshold` configures the mesh so that there is no minimum threshold for the mesh point connection strength.

Next, the radio1 hardware is set up for the WiFi network for the end devices. First the WiFi antenna is enabled on radio1 via the command `uci set wireless.radio1.disabled='0'` and the channel is set to a corresponding channel given the number of routers in the system via the command `uci set wireless.radio1.channel='x'`. The channel selected (replacing the `x`) cycles between 11, 6, and 1, as those are the channels that do not overlap. Radio1 runs on the longer range, 802.11n protocol because it is the only one supported by most device's second radio.

The software settings for radio1 are then changed on the default_radio1 interface via the following UCI commands:

```
uci set wireless.default_radio1.device='radio1'
uci set wireless.default_radio1.network='lan wan wan6'
uci set wireless.default_radio1.mode='ap'
uci set wireless.default_radio1.ssid='CentriFi'
uci set wireless.default_radio1.encryption='psk2'
uci set wireless.default_radio1.key='mymesh'
uci set wireless.default_radio1.ft_over_ds='1'
uci set wireless.default_radio1.ft_psk_generate_local='1'
uci set wireless.default_radio1.ieee80211r='1'
uci set wireless.default_radio1.disassoc_low_ack='0'
```

The option SSID, key, and encryption determine the network name and security measure. These are simply default settings and can be changed later. We chose WPA2 so that a wider range of devices are supported in the case that a user does not have any WPA3 capable devices. But that can easily be changed to WPA3 through the web interface. The three options `ft_over_ds`, `ft_psk_generate_local`, and `ieee80211r` add support for 802.11r which provides faster transitioning for clients between different mesh points. Finally, `disassoc_low_ack` is set to 0 to allow mobile devices to stay connected without timing out as often.

The `uci commit` command is then run to commit all the changes made to network and wireless settings. The file `/etc/rc.local` is then edited through the `sed` command, in order to run the necessary processes at the startup of the router. Once added, the commands are then initiated upon startup. The following commands are added to the files:

On the master router:

```
/root/CentriFi/manage.py runserver 0.0.0.0:8000 &
ntpd -q -p 1.openwrt.pool.ntp.org &
```

```
sleep 10 && vnstat -u -i wlan1 &
sleep 10 && vnstat -u -i br-lan &
```

On the slave routers:

```
ntpd -q -p 1.openwrt.pool.ntp.org &
sleep 10 && vnstat -u -i wlan1 &
sleep 10 && vnstat -u -i br-lan &
```

These commands initialize the Django web server on the master router, sync the system time to UTC, and start vnstat monitoring of the used network interfaces. The sleep commands are done to ensure vnstat is up and running before the commands are issued. The final step in the installation process is the rebooting of the router. If the routers were not rebooted at the end of this process, the command `wifi up` would be needed to bring the WiFi network up.

After the master router reboots, the installation script prompts for the slave routers to be connected, to run the installation process. Once all the routers are configured and rebooted, the routers are reachable at the assigned IP address and automatically begin peering to other meshes in the network. To test connection between two router, the command `iw dev wlan0 station dump` can be run. Upon success, the other routers should appear in a list following the command. Additionally, the web server is now up and running, and the network can be verified there, by navigating to 192.168.1.1:8000 in any web browser on a device connected to the network.

## 4.2 Back-End

The back-end of the CentriFi application was created in Django, in order to run the web server hosted on the master router. Django handles all of the processing of requests sent by the angular front-end by updating the configuration of the routers and sending a respone back to the web interface upon completion. The main functionality is written as view functions, which take an HTTP Request as input, and respond with a JSON response object. There are eight functions to implement the main api responses.

The `login` view utilizes the Pexpect module to verify the admin login, by checking the submitted password against the root password of the master router.

The `list_network_info` view uses Pexpect to run OpenWrt Unified Configuration Interface (UCI) commands on the master router to get the WiFi network information, as well as running nmap [17] to get a list of all routers connected to the mesh and their corresponding IP address, MAC address, and manufacturer (if known, based on the MAC address). Since the routers are assigned the IPs 192.168.1.1-31, nmap only scans those ports to determine the information for those routers.

The `set_router_passwords` view uses Pexpect to SSH into the connected routers, as determined by `list_network_info`. It then uses the `passwd` command to change the passwords for all connected routers.

The `get_wifi_settings` view uses Pexpect to run UCI commands to get the WiFi SSID, password, and security scheme (WPA2 or WPA3), as well as the channel each connected access point is running its connections on. The `set_wifi_settings` is largely the same, except it is given the information as a JSON object from the front-end, and then changes the settings instead of getting them, once again using Pexpect to do so.

The `network_statistics` view uses Pexpect and vnstat [18] to gather lightweight statistics regarding network traffic on each interface for each router on the network.

The `list_end_devices` and `set_end_devices` views use Pexpect to look at the DHCP leases to get information regarding connected devices. The MAC address of each connected device, as well as the setting for restricted roaming are stored in a data file. Then, Pexpect is used to run UCI commands to add the device to the macfilter list for all of the access points other than the one it is specified to connect to.

## 4.3 Front-End

Our angular web app was created using the Angular CLI making it easier to generate the components necessary for our web application. Following Angular's design paradigm ensures that the components are re-usuable and permit low coupling of the web application's code. Further, Google's material design library and icons are used for buttons, text boxes, and other UI elements in order to maintain UI quality while maintaining development speed. The angular project is then compiled ahead of time into static files of Javascript, CSS, and HTML making the front-end web bundle both smaller and faster. This also allows the Django server to serve the static website in a production environment.

The web application consists of six pages. Each of these pages is split into its own Angular component in order to improve code readability and allow for further development and expansion. If a new page needs to be added in the future, another component can easily be added and linked to via a button on the homepage.

First, it contains a ”Login” page that allows users to login and authenticate themselves (Figure 4.1). Second, it contains the ”Homepage” (Figure 4.2), which allows users to navigate to the remaining four pages, and also shows users info about the client device network and the routers that make up the network. Third, it contains the ”Change CentriFi Password” page (Figure 4.3) that allows users to change the password that allows them to login to the web application in the first place. Fourth, it contains a ”WiFi Settings” Page (Figure 4.4) that allows users to change the SSID, set the password, change security settings of the client network, as well as the radio channel of each router on the network. Fifth, it contains the ”Network Statistics” page (Figure 4.5) that allows users to view statistics about the network bandwidth being used by clients on each router in the network. Sixth, it contains the ”Connected Device Configuration” page (Figure 4.6) which allows users to configure whether client devices should be allowed to roam from router to router or be restricted to connect to one specific router on the network.

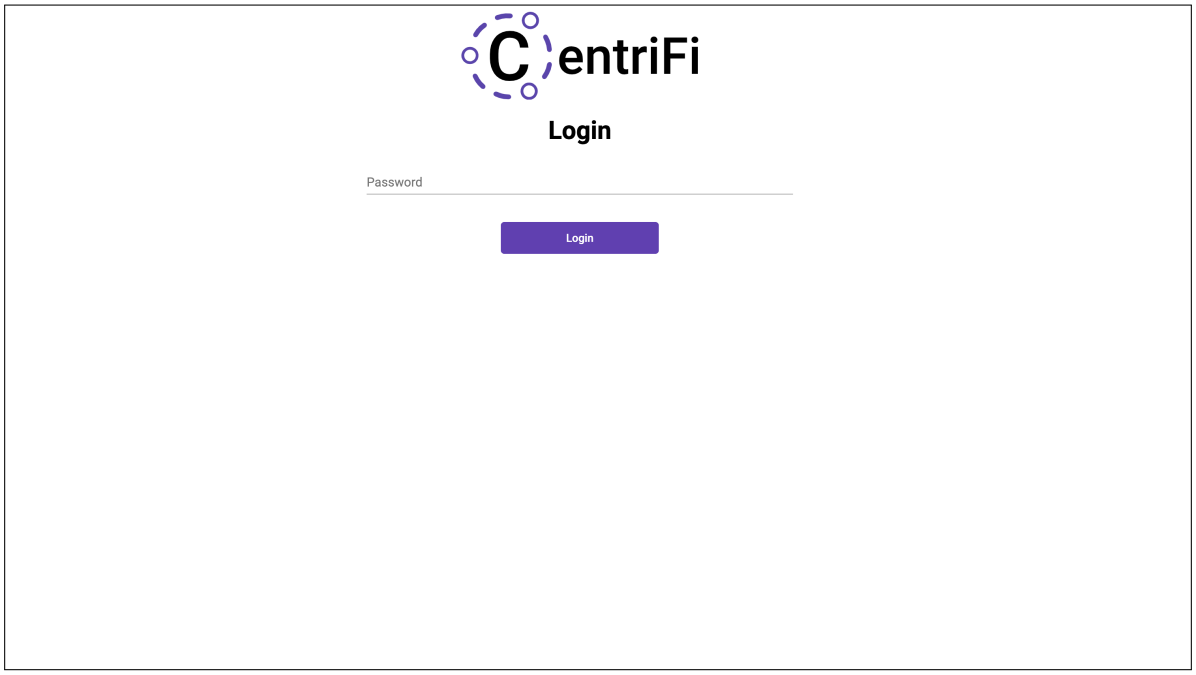
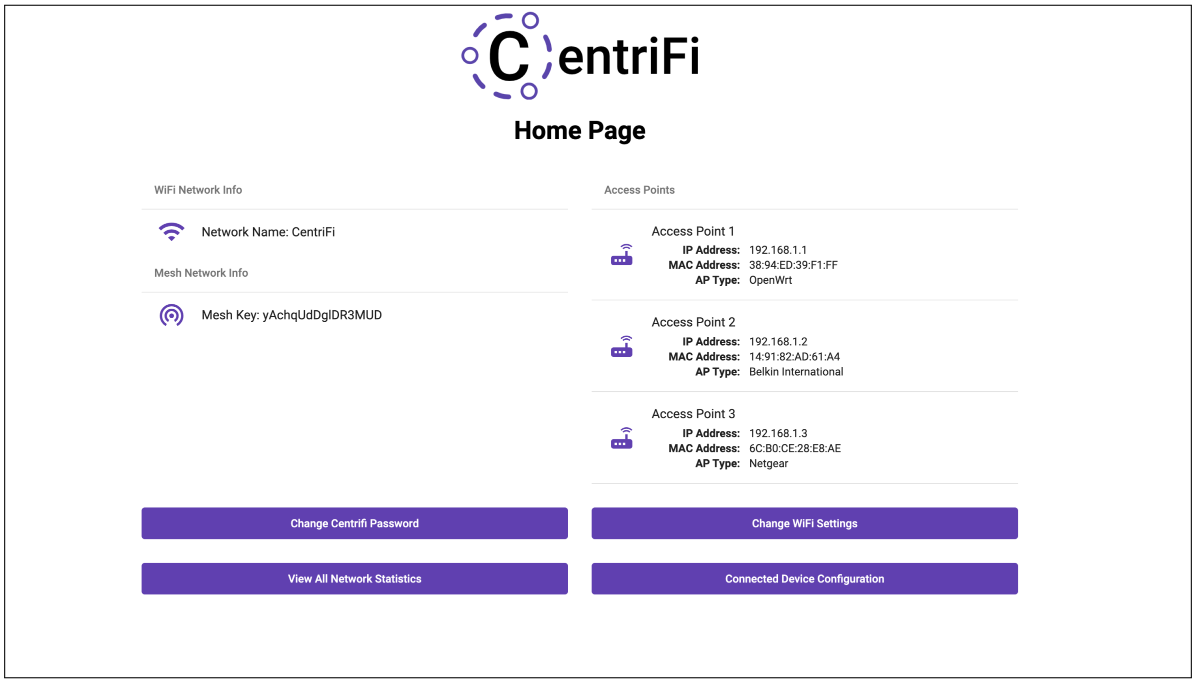


Figure 4.1: Login Page



Network Name: CentriFi

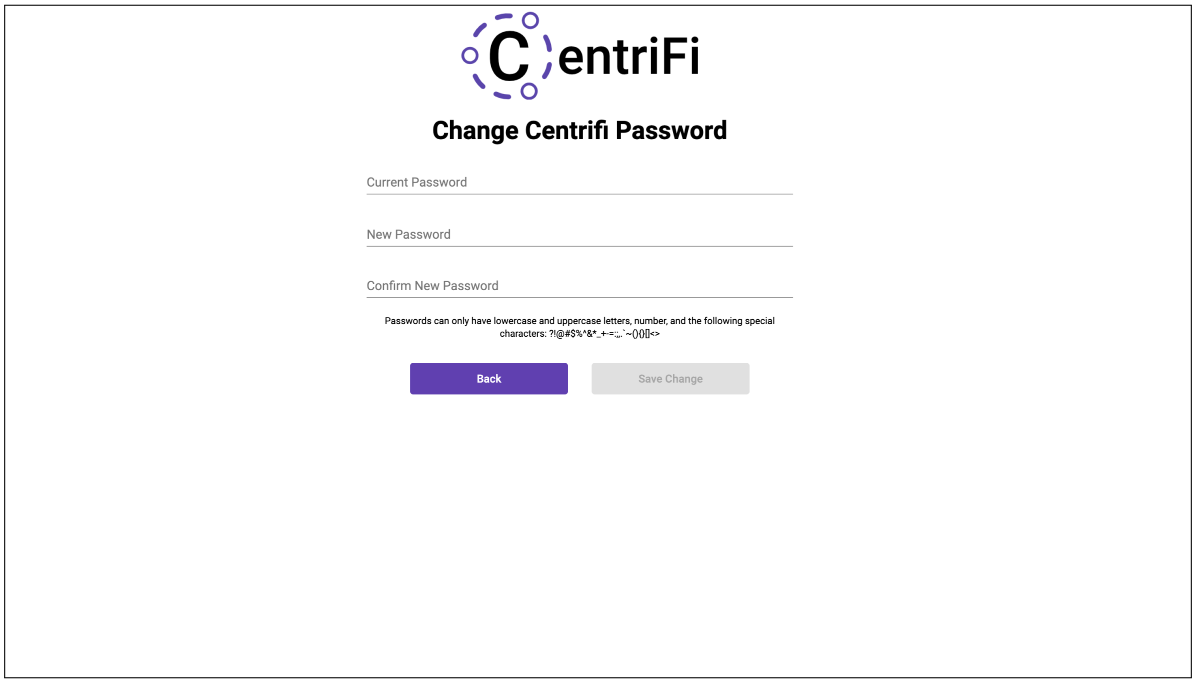
Mesh Key: yAchqUdDglDR3MUD

Access Point 1
IP Address: 192.168.1.1
MAC Address: 38:94:ED:39:F1:FF
AP Type: OpenWrt

Access Point 2
IP Address: 192.168.1.2
MAC Address: 14:91:82:AD:61:A4
AP Type: Belkin International

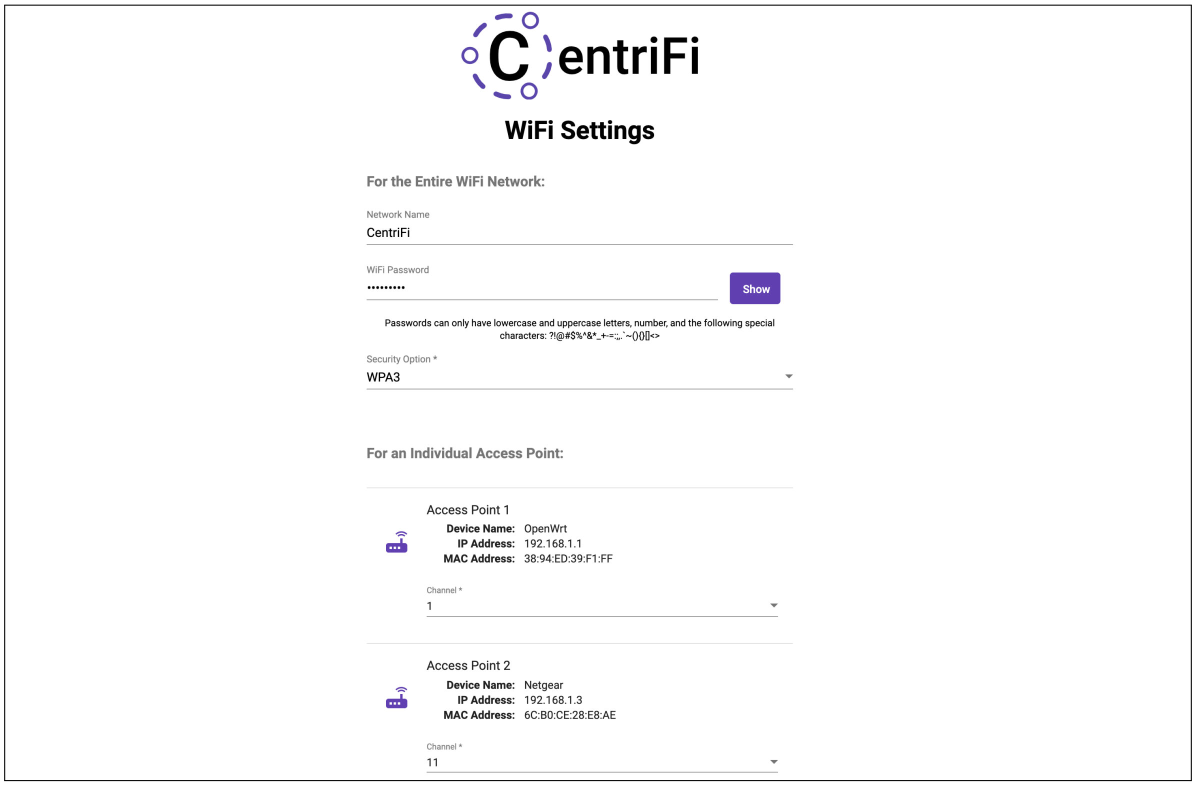
Access Point 3
IP Address: 192.168.1.3
MAC Address: 6C:B0:CE:28:E8:AE
AP Type: Netgear

Figure 4.2: CentriFi Home Page



Passwords can only have lowercase and uppercase letters, number, and the following special characters: ?!@#$%^&*_+-=:;,.`~(){}[]<>

Figure 4.3: CentriFi System Password Page



For the Entire WiFi Network:

Network Name: CentriFi

WiFi Password: •••••••••

Passwords can only have lowercase and uppercase letters, number, and the following special characters: ?!@#$%^&*_+-=:;,.`~(){}[]<>

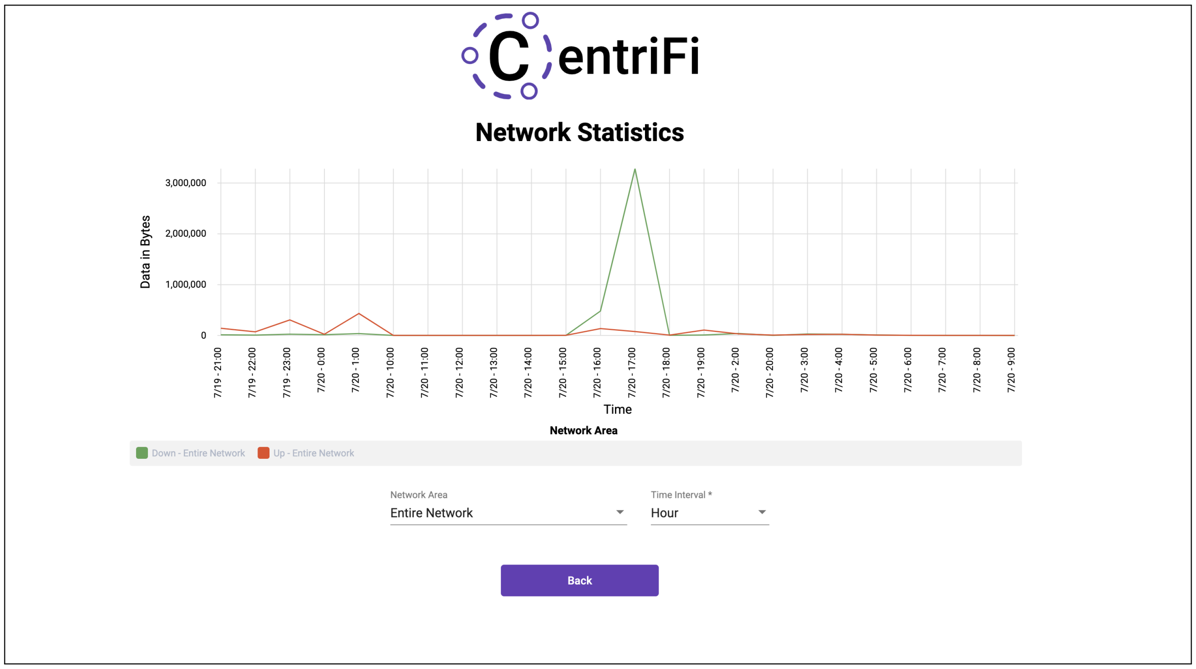
Security Option *: WPA3

For an Individual Access Point:

Access Point 1
Device Name: OpenWrt
IP Address: 192.168.1.1
MAC Address: 38:94:ED:39:F1:FF
Channel *: 1

Access Point 2
Device Name: Netgear
IP Address: 192.168.1.3
MAC Address: 6C:B0:CE:28:E8:AE
Channel *: 11

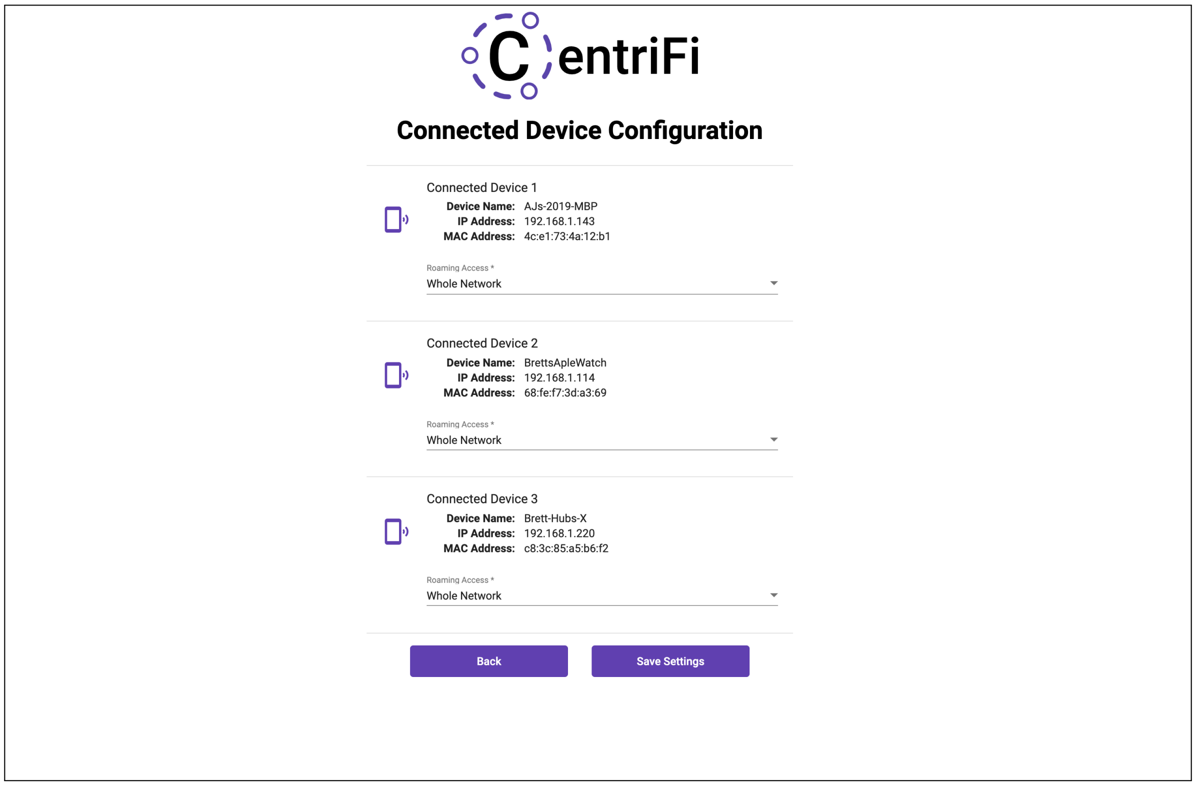
Figure 4.4: WiFi Settings Page



Network Area: Entire Network

Time Interval *: Hour

Figure 4.5: Network Statistics Page



Connected Device 1
**Device Name:** AJs-2019-MBP
**IP Address:** 192.168.1.143
**MAC Address:** 4c:e1:73:4a:12:b1
Roaming Access *: Whole Network

Connected Device 2
**Device Name:** BrettsApleWatch
**IP Address:** 192.168.1.114
**MAC Address:** 68:fe:f7:3d:a3:69
Roaming Access *: Whole Network

Connected Device 3
**Device Name:** Brett-Hubs-X
**IP Address:** 192.168.1.220
**MAC Address:** c8:3c:85:a5:b6:f2
Roaming Access *: Whole Network

Figure 4.6: Connected Device Configuration Page

# Chapter 5

# Evaluation and Testing

For our evaluation and testing we chose to test the routers in the home environment, since it is was the only environment we had ready access to during quarantine due to COVID-19. Nonetheless, given that the home environment should be relatively comparable physically to most small business or medical environments, these tests should be representative of CentriFi's performance in those environments as well. These tests aim to show the benefits of a mesh network that CentriFi is able to setup and configure, while also showing the rather minimal router resource usage required.

## 5.1 Methodology

For our testing hardware in this paper, we use the three following routers: Linksys EA8300 (Qualcomm QCA9886) released in 2017, Netgear R7800 Nighthawk X4S (Qualcomm QCA9984) released in 2016, Netgear R6100 (Qualcomm QCA9880) released in 2013.

We select these access points based on their chipsets with each chipset being supported by the ”ath10k” wireless driver. We also use these routers because they come from two different manufacturers. With this brand diversity, our test can show the interoperability between router brands that CentriFi provides. Also, the Netgear R7800 and Linksys EA3000 provide examples of high- to mid-range routers from three to four years ago with powerful antennae setups, while the Netgear R6100 provides an example of a budget router with less powerful hardware that is nearing 7 years old.

The placement of the routers is presented in Figure 5.1. The two slave routers are set up in two testing locations: Router 2 is 42 feet away from Router 1, and Router 3 is 70 feet from Router 1 and 30 feet away from Router 2. These locations span a house with 2 to 3 walls in between each. Our client device is used to run all tests, and is positioned a few feet away from Router 2.

For the hardware configuration, the Netgear R7800 Nighthawk X4S serves as Router 1. Router 1 is the master router that is connected to the internet port on a AT&T modem for a home network. The other two routers are configured as slave routers. The Linksys EA8300 serves as Router 2, and the Netgear R6100 serves as Router 3.

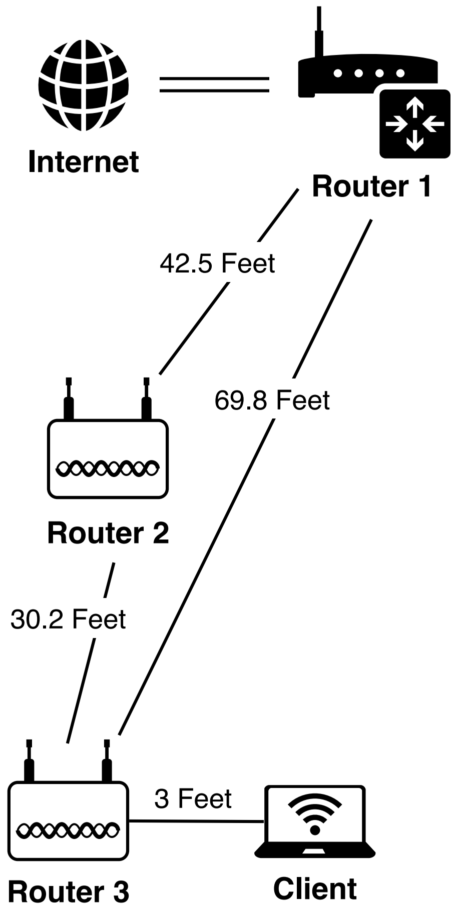


Figure 5.1: Hardware Placement

In order to evaluate network performance, three tests are performed. The first test is a Packet Loss Test, which determines the consistency of the connection between the network and the client device. Using the command `ping 8.8.8.8 -c 256`, we are able to ping Google's public DNS server at IP address 8.8.8.8 with a 64 byte packet 256 times in a row. This utility records whether or not a packet is lost in the process, allowing us to monitor network connection consistency by calculating the packet loss percentage. The test is performed with the client connected to a different router each time, for a total of three trials.

The second test is a Network Bandwidth Test used to determine the quality of the connection. To do so, the network bandwidth is tested using *speedtest.net* which aggregates download and upload speed by streaming sample data to a selected server. Further, for each test the test is run three separate times so as to take an average of the three results. To ensure consistency, every test is connected to the same server. Similar to the first test, this bandwidth test is performed with the client connected to a different router each time.

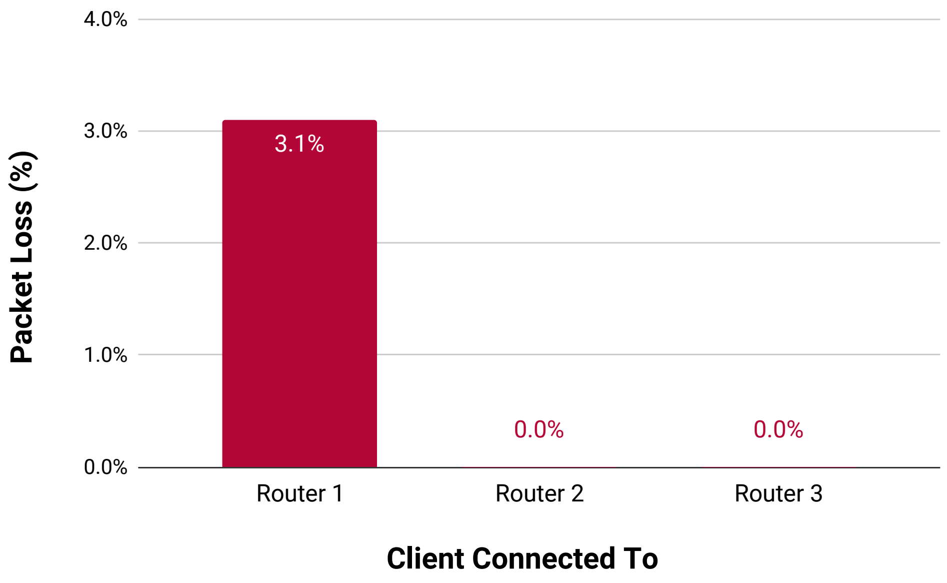


Figure 5.2: Packet loss in a 256 packet sample for a client connected to corresponding router

The third and final network performance test is a Processor Utilization Test. For this test, data is streamed from the client device connected to Router 3 to a public server using the program iPerf3 via the command `./iperf3 -c iperf.worldstream.nl -p 5201 -i 1 -t 120`. While this stream is being sent, the processor utilization for each router on the three router network is recorded using the `top` software utility built into OpenWrt. That processor utilization is recorded over 10 short time intervals during the stream. On Router 1, the percentage used by the web server is removed in order to focus our analysis on the other processes that CentriFi employs.

We also take measurements of the resource usage by CentriFi's server and other accompanying software on the master router. This is done to determine the minimum RAM and storage requirements for the master router.

## 5.2 Results and Discussion

The results for the Packet Loss Test can be seen in Figure 5.2. In these results, we see our client device struggle to send and receive packets when connected to Router 1, with 3.7% packet loss. However, given a connection to Router 2 or 3, which provides a mesh backbone to extend this network, we see the packet loss drop to 0%.

The results for the Bandwidth Test shown in Figure 5.3 indicate a similar result. In these results, we see our client device struggles to acquire bandwidth when connected to Router 1. Again, connecting to Router 2 or Router 3 instead yields a much higher result, giving roughly 3x improvements in download bandwidth and 10x improvements in upload bandwidth. This higher bandwidth also translates into a shorter end-to-end communication delay, which highly affects the energy efficiency of IoT devices [19].

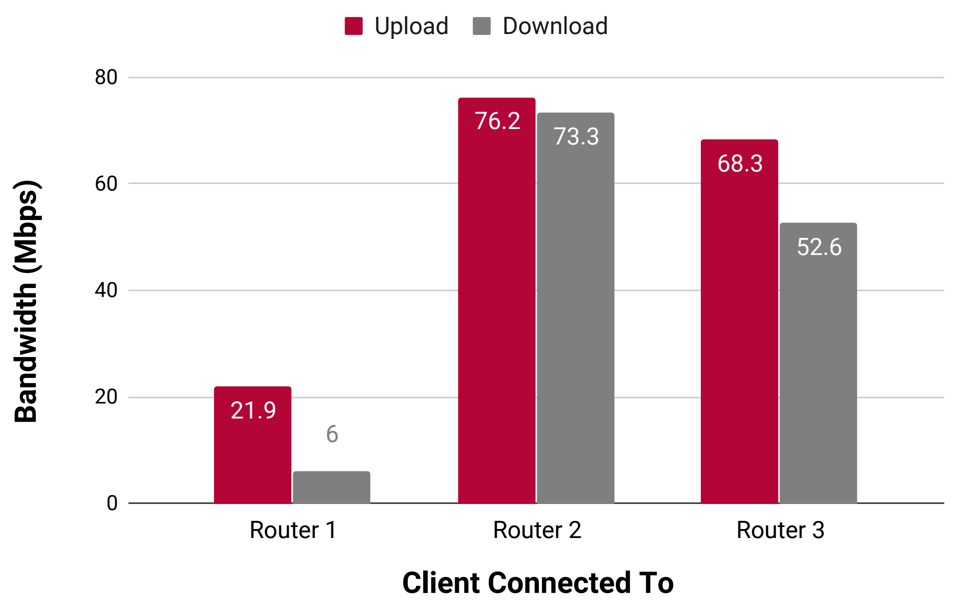


Figure 5.3: Internet Connectivity Bandwidth Test

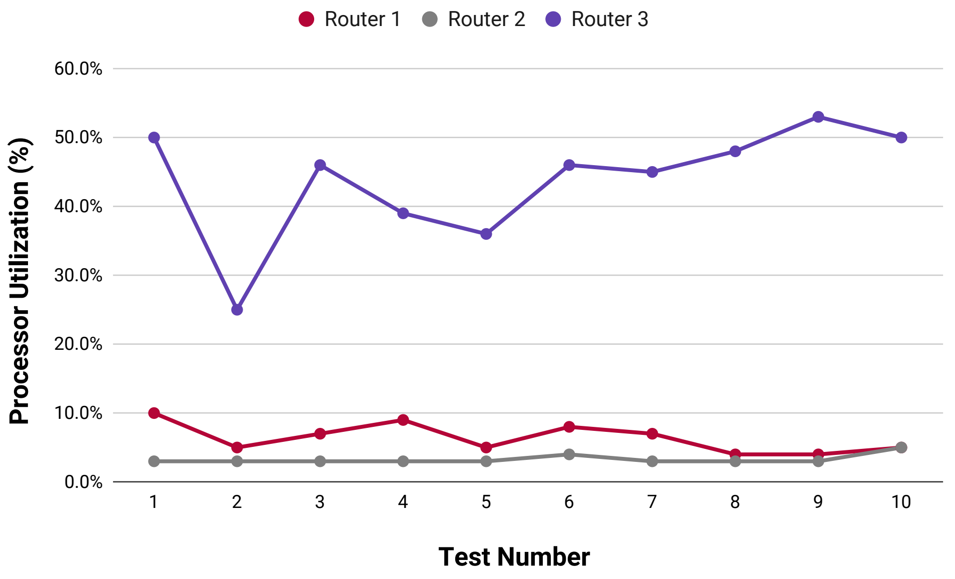


Figure 5.4: Processor utilization during a data stream from Router 3 via iPerf3

Connecting to Router 3 decreases bandwidth when compared to connecting to Router 2. This is due to this connection being a multiple-hop communication across the network, and the fact that Router 3 has a single core processor from 2013. However, it may simply be due to the overhead of forwarding packets wirelessly across multiple hops. Fortunately, this bandwidth penalty is quite low, and enables our network to reach even further.

Finally, we have the Processor Utilization Test presented in Figure 5.4. Even during a data stream test, where

Router 2 and Router 1 must process packets coming from Router 3, the processor utilization of those two devices is quite low, hovering around or below 10% for the duration of the test. Again, the somewhat higher processor usage of Router 3 is likely due to its aging hardware and the significantly slower 560 Mhz single-core CPU. Nonetheless, the processor utilization is quite low in general, leaving room for the system to be expanded with more processor-intensive processes in the future.

In the configuration of our setup, the Linksys EA8300 serves as Router 2. We also perform the Packet Loss Tests and Bandwidth Test where the positions of the Linksys EA8300 and the Netgear R6100 are swapped. That is, the Linksys EA8300 occupies Router 3's position, and the Netgear R6100 occupies Router 2's position. This alternate configuration results in a considerable drop of performance when connected to the Linksys EA8300 as Router 3. The packet loss test spikes to 2.7% and the bandwidth drops to about 20 Mbps for both upload and download. So, the hardware in the Netgear R6100 is not capable of relaying packets as consistently as the Linksys EA8300.

Therefore, placing less powerful routers on the edge of the network improves network performance and is something that users will be made aware of. Nonetheless, these tests show that a CentriFi network greatly benefits users who need to cover large areas with wireless connection at a lower cost and adequate performance level. Improvement can be seen in the realms of both bandwidth and packet loss. In the future, more testing could be performed in order to determine how dense and congested networks are managed. Most of these would simply be tests of the 802.11s, and, specifically, its implementation on OpenWrt. Nonetheless, further testing would still be valuable in holistically assessing CentriFi's usefulness.

The resource usage of the Master router is rather minimal. The amount of RAM that CentriFi uses on top of standard OpenWrt operating system processed was recorded at 99.9 MB. Given that, routers used need at least 256 MB of RAM. Similarly, CentriFi and its dependencies require 56.3 MB of RAM on top of the standard OpenWrt install; thereby, the master router requires at least 128 MB of flash storage as well.

Fortunately, these measurements signify that CentriFi is light enough for most modern routers on the 2020 market. At around the $120 price range, there are many options from different manufacturers that meet our requirements for the master router of 256 MB of RAM and 128 MB of flash storage. For example, both the Linksys EA8300 and he Netgear R7800 meet those requirements. The slave routers require even less resources, as shown with the Netgear R6100.

# Chapter 6

# Societal Issues

## 6.1 Ethical

CentriFi could be used to configure mesh networks in shelters for the unhoused, public libraries, and other resources that help marginalized and under-served communities. These networks would allow for reliable connections to the internet for any of the needs of the people in those communities. Additionally, by making CentriFi open-source, it allows for those interested in technology to look ”under the hood” to see how it works as well as make their own additions to the system if desired.

## 6.2 Social

The affordability of CentriFi allows for its use in areas that reach under-served populations and can help connect them to online resources. Mesh networking can also be used to facilitate basic infrastructure connectivity in areas without reliable, wide-range network access [20, 21], and CentriFi could be used to reliably configure these networks.

## 6.3 Political

With CentriFi, the user’s networking data is not stored in the cloud, it is theirs to do with what they want. Their configurations are stored locally thus offering greater privacy.

## 6.4 Economic

CentriFi can be used on mid and low-price range routers and therefore does not require purchasing new or expensive hardware. Additionally, the network can be expanded fairly easily, without needing to replace the existing network infrastructure. Therefore, CentriFi works well for low-budget applications.

## 6.5 Health and Safety

As it provides an easily-configured mesh network, CentriFi can provide more reliable connections for medical devices or home smart alarms compared to normal networks. Because of the customization CentriFi offers in configuring which access point a designated endpoint communicates with, the user can modify the connections to make sure that the smart alarms are communicating with the most available access point, ensuring a stable connection.

## 6.6 Manufacturability

CentriFi does not have any manufacturability concerns. Since CentriFi is open-source software that runs on already manufactured routers, it does not require anything new to be manufactured. Additionally, software does not require anything besides a hosting website, and this project's source will be hosted in an SCU IoT Lab's GitHub repository.

## 6.7 Sustainability

Since CentriFi is open-source and works on a wide variety of devices, it is more sustainable than the existing, proprietary options as it can be used to upgrade an existing network without requiring the purchase of new equipment.

## 6.8 Environmental Impact

CentriFi increases the lifespan of routers by providing features that are typically locked to newer hardware, which reduces electronic waste created by having to replace older hardware with newly manufactured units.

## 6.9 Usability

CentriFi is extremely user-friendly. CentriFi uses Google's material design form control, for a clean, modern, and easily understood user interface. Additionally, the installation process is simple, it just requires running a single python executable and following on-screen prompts.

## 6.10 Lifelong Learning

This project contributed to our understanding of web based applications, and utilizing existing frameworks. Furthermore, this project helped us learn how to read and understand sparse documentation, research debugging methods, and engineer solutions when we could not find them. It also furthered our understanding of the importance of clean coding practices.

## 6.11 Compassion

Part of the motivation for creating CentriFi was to contribute to internet availability for under-served populations. As CentriFi can run on existing, older hardware, we hope for it to be used in non-profit organizations, potentially on donated hardware, to provide internet access for populations served by those non-profits. Examples of potential uses include providing stable internet access for shelters for the unhoused, community service centers, and more.

# Chapter 7

# Conclusion

In this paper, we demonstrated the need for a software management system that provides centralized configuration of mesh networks while remaining open-source, easy to setup, and expandable, since current solutions do not possess all three characteristics. We then outlined a system that fulfills those three criteria, CentriFi.

We built this system on top of the OpenWrt operating system to provide widespread consumer router support. Further, by building a web server into the system, we were able to centralize configuration and allow for easy user control over a web interface. Next, we tested the efficacy of the system through performing multiple network tests and monitoring resource usage. Through these tests, we were able to demonstrate the measurable benefits CentriFi has in improving network range on budget consumer hardware. Finally, we discussed the positive impacts that a system like this could have on populations typically under-served by current networking solutions.

Of course, our solution is not perfect. We would have liked to add more configuration and monitoring features if given more time. Further, testing the stability and performance of our system across many more routers would most likely be necessary to ensure users can use the system with on a wide variety of hardware with good results.

We hope that future work will be done to CentriFi either by a proceeding senior design team or the open-source community. We believe that the system could further benefit from more in-depth bandwidth usage statistics so users can analyze the network usage and connection stability on a per-client-device basis. Additionally, the system could be extended to use a custom packet routing algorithm like OSLR [22] so that a user can further configure the topology of their network, instead of allowing the automatic configuration algorithm used by 802.11s. These and other features could easily be integrated to the web interface, given the proper time to do so. Also, creating a database of supported routers and cataloging their performance could be helpful in aiding users as to what hardware they should buy. Finally, a standard like JavaScript Web Tokens [23] could be used to encrypt the communication of the web interface so that HTTP requests are not sent in plain text.

# Appendices

# Appendix A - Supporting Materials

## A.1 Source Code and Demo Video

**Full implementation is accessible via the following link:**
`https://github.com/SIOTLAB/CentriFi`.

**A short 5 minute demo of the system can be seen at the following link:**
`https://www.youtube.com/watch?v=GI7cOSr47i0`

## A.2 Activity Diagram

Pictured below is an activity diagram detailing the possible activities that a user could perform on a CentriFi system.

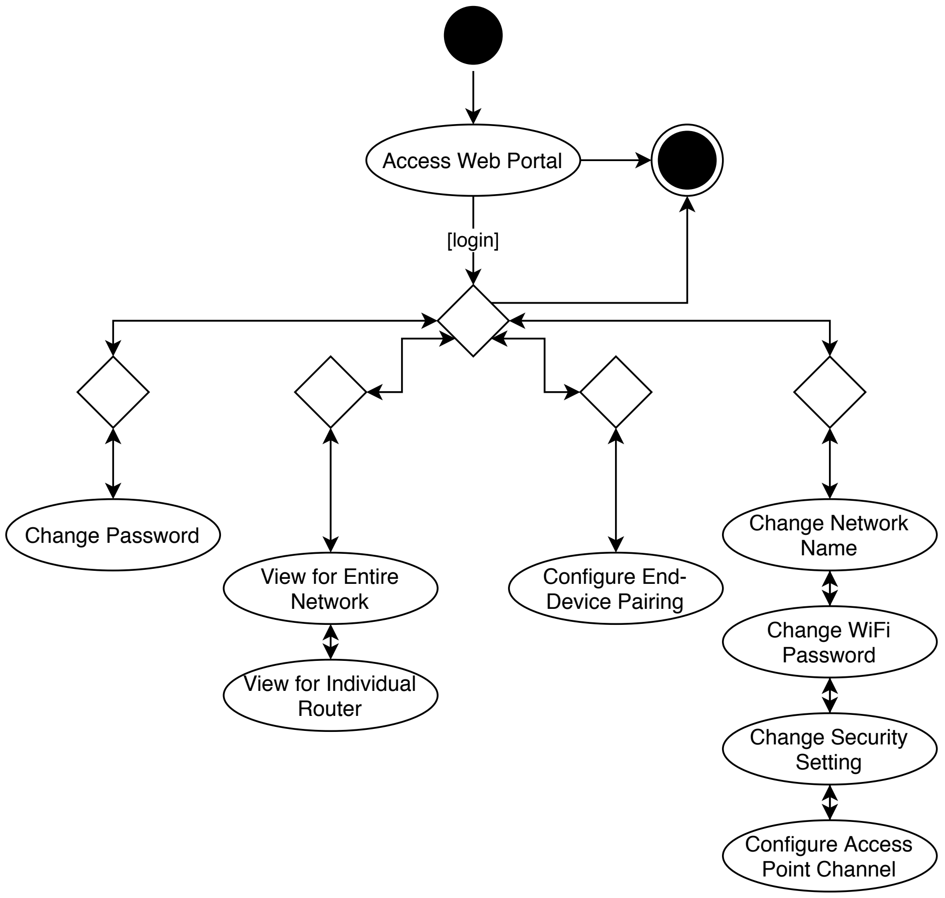


Figure A.1: Activity Diagram for the CentriFi System

# Appendix B - User Guide

## B.1 Overview

CentriFi is an access point management system that makes creating robust and customizable networks realistic and affordable. Built on the free OpenWRT platform, CentriFi allows for local area networks to be exactly configured and maintained on low cost hardware, intuitively and easily.

Currently CentriFi sets up a mesh network automatically via an install script.

From there, CentriFi can be further configured via a web inteface, where the following settings can currently be configured:

- The CentriFi admin password that gives access to configuration
- The SSID, set the password, and the set the security settings (WPA2/WPA3) of the client wifi network
- The radio channel for the client wifi network of each router on the network.
- Whether a specific client device should be allowed to roam from router to router or be restricted to connect to one specific router on the network
- This web interface also allows users to view statistics about the network bandwidth being used by the network as a whole or by each router in the network.

## B.2 Installation Guide

1. First, make sure you have Python 3.7 or newer installed. Open the command line (powershell, command prompt, terminal, or the equivalent). You can check your python version by running the command:
   ```
   python -V or python --version
   ```
2. Second, check to see if you have paramiko version 2.7 or newer installed. You can check this by running the command:
   ```
   pip show paramiko
   ```
3. If paramiko is not found, install it by running the command:
   ```
   pip install paramiko
   ```
4. Then, download the file 'install_script_v2.py' from the setup folder of this repository, navigate to its location within the terminal window, and run it using the command:
   ```
   python install_script_v2.py
   ```
5. First, power on the master access point, and connect it to the internet using an ethernet cable and the WAN port. Then, connect the access point to your computer (running the install script) using an additional ethernet cable and any of the LAN ports on the device. Wait around 10-15 seconds, or until the lights on the port show a stable connection (if there are lights for the port).
6. Then, follow the prompts in the terminal window to configure the first (master) access point.

7. If you have additional access points to configure: After the first access point has been configured, it will reboot. Once it is finished rebooting, you can disconnect it and connect the next access point to be configured. Once again, connect the access points WAN port via ethernet to the internet, and then connect it your computer (running the install script) using the additional ethernet cable and any of the LAN ports on the device. Again, wait around 10-15 seconds, or until the lights on the port show a stable connection (if there are lights for the port).
8. Then, continue following the prompts to configure the additional access points.
9. Once you have configured all of the access points, you can then move them around to set up the mesh network within the location you want to set up the mesh. The master access point needs to be connected to ethernet (via the WAN port) and power, the slave access points just need to be connected to power.
10. Once they are set up and powered on, wait 1-2 minutes and they should be up and running the mesh. Each router will be running CentriFi on top of OpenWrt 19.07 and configured with an new IP address starting with the master router at 192.168.1.1, with the following routers having the sequential IP addresses 192.1.168.2, 192.1.168.3,...
11. To configure the devices further, and look at the settings, connect to the network 'CentriFi' with password 'centrifi'.
12. Then, on that device, in a web browser's address bar, visit 192.168.1.1:8000 and you should see the CentriFi homepage. If you still need to access OpenWrt's Luci web interface you can do so at the corresponding IP address for each router (e.g. 192.168.1.1 or 192.168.1.3).

# Bibliography

[1] P. Newman, "IoT Report: How Internet of Things technology growth is reaching mainstream companies and consumers," Jan 2019. [Online]. Available: https://www.businessinsider.com/internet-of-things-report

[2] E. O'Dowd, "Considerations for Deploying Healthcare Wireless Networks," Feb 2019. [Online]. Available: https://hitinfrastructure.com/features/considerations-for-deploying-healthcare-wireless-networks

[3] Yu Liu, Kin-Fai Tong, Xiangdong Qiu, Ying Liu, and Xuyang Ding, "Wireless Mesh Networks in IoT networks," pp. 183–185, 2017.

[4] B. Dezfouli, V. Esmaeelzadeh, J. Sheth, and M. Radi, "A review of software-defined WLANs: Architectures and central control mechanisms," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 431–463, 2018.

[5] R. Matam and T. Somanath, "Provably Secure Routing Protocol for Wireless Mesh Networks," *International Journal of Network Security*, vol. 16, pp. 182–192, 05 2014.

[6] S. Alanazi, J. Al-Muhtadi, A. Derhab, K. Saleem, A. N. AlRomi, H. S. Alholaibah, and J. J. P. C. Rodrigues, "On resilience of Wireless Mesh routing protocol against DoS attacks in IoT-based ambient assisted living applications," in *2015 17th International Conference on E-health Networking, Application Services (HealthCom)*, 2015, pp. 205–210.

[7] "Cisco Meraki." [Online]. Available: https://meraki.cisco.com/products/wireless

[8] "Aruba." [Online]. Available: https://www.arubanetworks.com/products/networking/

[9] "OpenWrt." [Online]. Available: https://openwrt.org/

[10] "OpenWISP." [Online]. Available: http://openwisp.org/

[11] "OpenWrt 802.11s based wireless mesh network." [Online]. Available: https://openwrt.org/docs/guide-user/network/wifi/mesh/80211s

[12] "Tomato." [Online]. Available: https://advancedtomato.com/

[13] "DD-WRT." [Online]. Available: https://dd-wrt.com/

[14] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, "IEEE 802.11s: The WLAN Mesh Standard," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 104–111, 2010.

[15] "Recommended Routers." [Online]. Available: https://openwrt.org/toh/recommended_routers

[16] "Pexpect Documentation." [Online]. Available: https://pexpect.readthedocs.io/en/stable/

[17] "nmap." [Online]. Available: https://nmap.org/

[18] "vnstat." [Online]. Available: https://humdi.net/vnstat/

[19] J. Sheth and B. Dezfouli, "Enhancing the Energy-Efficiency and Timeliness of IoT Communication in WiFi Networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9085–9097, 2019.

[20] B. Dezfouli, M. Radi, and O. Chipara, “REWIMO: A real-time and reliable low-power wireless mobile network,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 13, no. 3, pp. 1–42, 2017.

[21] J. A. Greig, “Wireless Mesh Networks as Community Hubs: Analysis of Small-Scale Wireless Mesh Networks and Community-Centered Technology Training,” *Journal of Information Policy*, vol. 8, pp. 232–266, 2018. [Online]. Available: https://www.jstor.org/stable/10.5325/jinfopoli.8.2018.0232

[22] “OSLR Mesh.” [Online]. Available: https://openwrt.org/docs/guide-user/network/wifi/mesh/olsr

[23] “JSON Web Tokens.” [Online]. Available: https://jwt.io