

6-11-2019

Top Level Mesh

Matt Jasaitis

Tristen Islam

Follow this and additional works at: https://scholarcommons.scu.edu/cseng_senior



Part of the [Computer Engineering Commons](#)

Recommended Citation

Jasaitis, Matt and Islam, Tristen, "Top Level Mesh" (2019). *Computer Engineering Senior Theses*. 154.
https://scholarcommons.scu.edu/cseng_senior/154

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Computer Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

Date: June 12, 2019

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY


Matt Jasaitis
Tristen Islam

ENTITLED

Top Level Mesh

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

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



Thesis Advisor



Department Chair

Top Level Mesh

by

Matt Jasaitis
Tristen Islam

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

Santa Clara, California
June 11, 2019

Top Level Mesh

Matt Jasaitis
Tristen Islam

Department of Computer Engineering
Santa Clara University
June 11, 2019

ABSTRACT

This paper will cover our report on the Top Level Mesh. We have built a web-based system for mesh network management. This system allows network utilities to be used from a web-based interface to monitor and manage the transfer of data. The system runs primarily on Raspberry Pis using Raspbian Linux. Users can access the system through web browsers to both configure the system and interact with the data on the network. We discuss our motivation for the project, design decisions made, technologies used and more throughout this report. We conclude with some lessons learned and future work to be done.

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Solution	2
2	System Design	3
2.1	Functional	3
2.1.1	Critical	3
2.1.2	Recommended	3
2.1.3	Suggested	3
2.2	Non-Functional	4
2.2.1	Critical	4
2.2.2	Recommended	4
2.2.3	Suggested	4
2.3	Constraints	4
2.4	Use Cases	5
2.4.1	Forward Traffic	5
2.4.2	Manage Traffic	5
2.4.3	Take Data from Device	6
2.4.4	Send Network Data	6
2.5	Activity Diagram	7
2.5.1	Mesh Manager	7
2.5.2	Mesh User	7
2.6	High Level View Of System	9
2.6.1	Network Status	9
2.6.2	Network Data	9
2.6.3	ARP Table Graph	10
2.6.4	Traffic Control	10
2.6.5	Auto Shaping	11
2.7	Technologies	12
2.8	Component State Diagrams	14
2.8.1	Setup of Individual Node	14
2.8.2	Admin State Flow	14
3	Implementation	16
3.1	Architectural Diagram	16
3.2	Design Rationale	17
3.2.1	Technologies	17
3.2.2	Layout	17
3.3	Description of System Implemented	18
3.3.1	Nodes	18
3.3.2	Frontend	18
3.4	Node Connectivity	20

3.5	Networking Protocols	20
3.6	Traffic data	21
3.7	ARP Data	22
3.8	Traffic Control Result	22
3.9	Algorithms and Techniques	23
3.9.1	Decision Trees	23
3.9.2	Statistics and Routing	23
3.9.3	Classification	23
3.9.4	Heuristics	24
3.10	Frontend Information	25
3.10.1	Sent Traffic	25
3.10.2	Frontend Networking	25
3.11	Node Functionality	26
3.11.1	Graphical Images	26
3.11.2	Data	27
3.12	Example Work Flow of System	29
3.13	Test Plan	30
3.13.1	Verification	30
3.13.2	Validation	30
4	Concluding Remarks	31
4.1	Challenges Encountered	31
4.1.1	Technology	31
4.2	Technology	31
4.3	Suggested Changes	32
4.3.1	Node Functionality	32
4.3.2	Frontend Functionality	32
4.3.3	Core System Changes	32
4.4	Lessons Learned	33
4.4.1	Implementation	33
4.4.2	Team Coordination	33
4.5	Ethics and Societal Impact	34
4.5.1	The Politics of Ownership	34
4.5.2	Privacy	34
4.5.3	Throttling and Fairness	34
4.5.4	Environmental Sustainability	35
4.5.5	Economic Sustainability	35
4.5.6	Societal Impact	35
4.5.7	Usability	35
4.5.8	Lifelong Learning	36
4.5.9	Compassion	36
4.6	Conclusion	37
A	User Manual	39
A.1	Setting Up Nodes	39
A.2	API	40
A.3	Frontend	40
B	code	41
B.1	routes.js	41
B.2	filecontroller.js	44
B.3	nodecap9.py	52
B.4	autoshape9.py	61
B.5	genarp5.py	77

B.6	comparearpgraphs3.py	79
B.7	pytc5.py	81
B.8	trafficplot7.py	82
B.9	generatefullarp.py	84
B.10	protocolplot.py	86
B.11	Frontend Angular Build	87

List of Figures

2.1	Use Cases for Internal & External Users	5
2.2	Manager Activity Diagram	7
2.3	User Activity Diagram	8
2.4	Network Status Mock-up Diagram	9
2.5	Network Status Mockup Diagram	10
2.6	Network Status Mockup Diagram	10
2.7	Traffic control	11
2.8	Traffic control	11
2.9	State diagram of node	14
2.10	State diagram of admin console	15
3.1	Architecture of Platform	16
3.2	Network Status Mockup Diagram	25
3.3	Network Status Mockup Diagram	26
3.4	Network Status Mockup Diagram	27

Chapter 1

Introduction

1.1 Motivation

In today's world, access to information is more important than ever for timely decision-making. For example, networked medical devices need to send large amounts of data to let doctors know their patients' status, firefighters need to get satellite images of brush fires, and foreign aid workers need to get information on food supplies. All of these tasks are relatively simple when there is an established wired or wireless infrastructure that can connect people to the applications they need. However, in many places in the world, a wireless network is not available. For example, there may be medical camps in remote areas that have to transfer data between themselves to the Internet. In addition, a hospital may have a large number of sensors to monitor a patient's condition that require guaranteed access. All of these scenarios require a robust infrastructure to manage the sending and receiving of data. We are trying to create a high performance wireless network that supports the delivery of high-priority traffic and is easy to deploy in any situation.

Currently, existing mesh software packages do not provide easy to deploy solutions that meet all the needs of ad hoc users such as emergency responders. For example, GoTenna provides a piece of hardware that connects to phones and enables data to be relayed between other phones with the same device. While this shows potential, it cannot be used to build large mesh networks (1). Another contender is Althea, a company which offers paid access to the Internet. However, it does not solve many of the problems that we are attempting to address with our bandwidth-aware routing. Althea has a payment system that is focused on setting up ISPs run by the community. It creates nodes in the mesh network and makes participation profitable for them (2). In short, both of these technologies, while similar, do not fully solve the issue of managing a complex network in the absence of infrastructure. Discussing every mesh product would take too long, but suffice it to say that solutions involving closed hardware, such as Google and Orbi, have their limitations and cannot accurately attune themselves to different environments. The software that comes the closest to what we are trying to accomplish is OpenWISP. However, this company lacks features that exploit the fact that mesh nodes are smart and have both edge and core functionality. OpenWISP offers some functionality to do edge tasks -

like quickly set up VPNs - but it does not offer the traffic management features that we want (3). One monitoring package that comes close to what we are attempting to do is Gotham - a visualization tool for batman-adv. This tool uses Hazelcast to help users visualize network topology. It also allows nodes to connect to the mesh automatically. While these features are useful, Gotham is built for one specific protocol (batman-adv.) and does not exploit all of the edge features of mesh architecture (4).

1.2 Solution

We have built a web-based system for the setup and configuration of wireless networks. Through a variety of techniques, our system provides a robust method of merging multiple wireless nodes into one network. The network is also able to manage up-and-down traffic to ensure priority data gets through. Additionally, the system provides near real-time data and graphics on the state of the network. This system differentiates itself from existing solutions by providing an integrated view of the network. We have developed software to live on the mesh and more efficiently move packets. From the web-based interface, a user is able to configure various properties of the mesh, as well as set various nodes as access points. The mesh network runs primarily on inexpensive hardware that is versatile. The nodes require minimal setup to be connected to the network and the management software. The initial connection to the main server where the web application is hosted could be through Ethernet or a wireless protocol that is pre-configured at start-up. The software is able to assign gateway nodes as well as configure pieces of the mesh that are specific for each protocol. Most of the protocols have routing algorithms built in, but this software is able to prioritize traffic.

We have implemented simple machine learning into the system for smarter traffic optimization. The web interface allows users to see which nodes are using a disproportionate amount of bandwidth and react accordingly.

In conclusion, our solution allows for the faster deployment and improved performance of wireless mesh networks. With this software, users have an easier time managing the complex protocols needed to set up a network and can do so from a unified web-based interface. The software also allows users to more readily diagnose and resolve problems in their network using performance metrics on the web interface.

Chapter 2

System Design

Functional requirements, nonfunctional requirements, and constraints are specified for the platform. Each requirement is categorized into 'Critical', 'Recommended', and 'Suggested' to specify how essential each component is to the platform.

2.1 Functional

2.1.1 Critical

- Web-based manager must allow users to configure how traffic is distributed
- Web-based manager must facilitate file transfer of important files and mission critical data
- System must display metrics to users

2.1.2 Recommended

- Implementation of machine learning to improve traffic distribution
- Keep traffic distribution equitable
- Device security

2.1.3 Suggested

- Implementation of traffic storing and forwarding
- Swapping of base routing protocols
- Base security features

2.2 Non-Functional

2.2.1 Critical

- Web interface must look clear and intuitive

2.2.2 Recommended

- Minimal bugs

2.2.3 Suggested

- Configurations will be pushed out quickly

2.3 Constraints

- Must run on web browsers
- Must work from within the mesh
- Mesh must run on Raspberry Pis with Alfa wireless cards
- Web servers must be accessible through WiFi

2.4 Use Cases

Use cases for the two stakeholder groups - namely, network users and network administrators - are detailed below. The network administrator has all of the functionality for managing traffic, while the clients have the ability to access the rest of the network as well as push form data up.

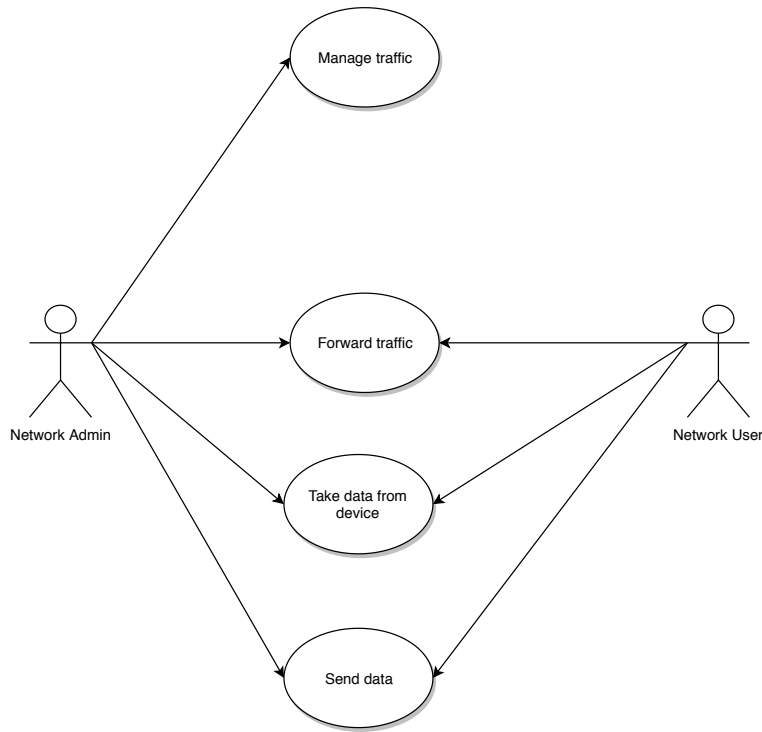


Figure 2.1: Use Cases for Internal & External Users

2.4.1 Forward Traffic

All mesh nodes can transfer traffic between each other. This will be a default action performed without user intervention. Traffic is forwarded through the 802.11s protocol and is handled by the Alfa WiFi cards on the Raspberry Pis.

Pre-conditions: The network is set up and nodes are connected to each other on the same ESSID.

Post-conditions: Traffic will be transferred between the nodes.

2.4.2 Manage Traffic

Node administrators are able to modify how the devices behave in a networked setting through the web interface. This involves choosing which IP addresses to prioritize and which ones to throttle. Administrators also have information on the current state of various devices. The throttling action is managed by the server that runs on the nodes. It is performed with the traffic control Linux utility.

Pre-conditions: The network is set up and nodes are connected to each other on the same ESSID.

Post-conditions: Nodes can prioritize different ports for the transmission and receiving of traffic. IP addresses can be blocked to improve performance if necessary. In short, basic networking commands are specified by the administrator and deployed to the network.

2.4.3 Take Data from Device

All users have access to the files on their current node and other nodes that they can contact. They can view these files to gain insight on how their network is behaving. The collection of data is managed through API endpoints on the nodes and is handled through the express utility. Most of the files that users can access with this action pertain to files containing various statistics or data on the network.

Pre-conditions: The network device is active and has a working WiFi connection.

Post-conditions: Users can view files on their device.

2.4.4 Send Network Data

Nodes have a "chron" task setup to send data to each other intermittently. This gives every node an up-to-date record of what the network is doing. The chron tasks are primarily run through the HTTP protocol.

Pre-conditions: The network device is active and has a working WiFi connection.

Post-conditions: The form data is stored in a file and pushed to the admin system when applicable.

2.5 Activity Diagram

The network manager is able to send files to and get information from clients.

2.5.1 Mesh Manager

Figure 4.1 highlights the network manager's ability to manage traffic and get information from devices. The web manager can be run on almost any device and has the ability to view many important characteristics of connected nodes. The manager itself is run through a simple web user interface that allows users to easily navigate the various pages. Network managers have three main functions:

1. They can view information on connected mesh nodes.
2. They can edit traffic configuration rules.
3. They can push traffic configuration rules to the rest of the nodes.

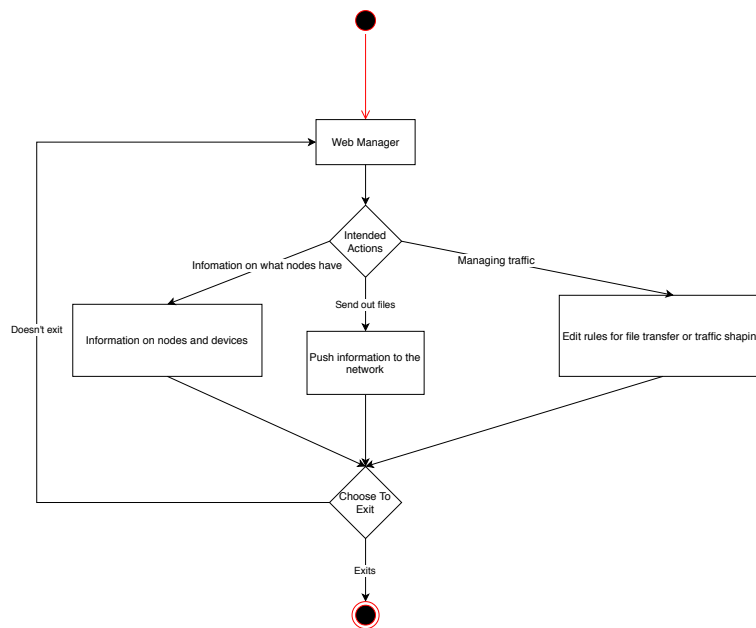


Figure 2.2: Manager Activity Diagram

2.5.2 Mesh User

Users are able to access locally stored resources and make network requests. Users on the mesh can also access the wider Internet through gateway nodes if they are available. That being said, the system may be configured in ways that do not permit Internet access, so it is uncertain whether they will succeed. Users will also be able to take data from the nodes themselves through their API endpoints.

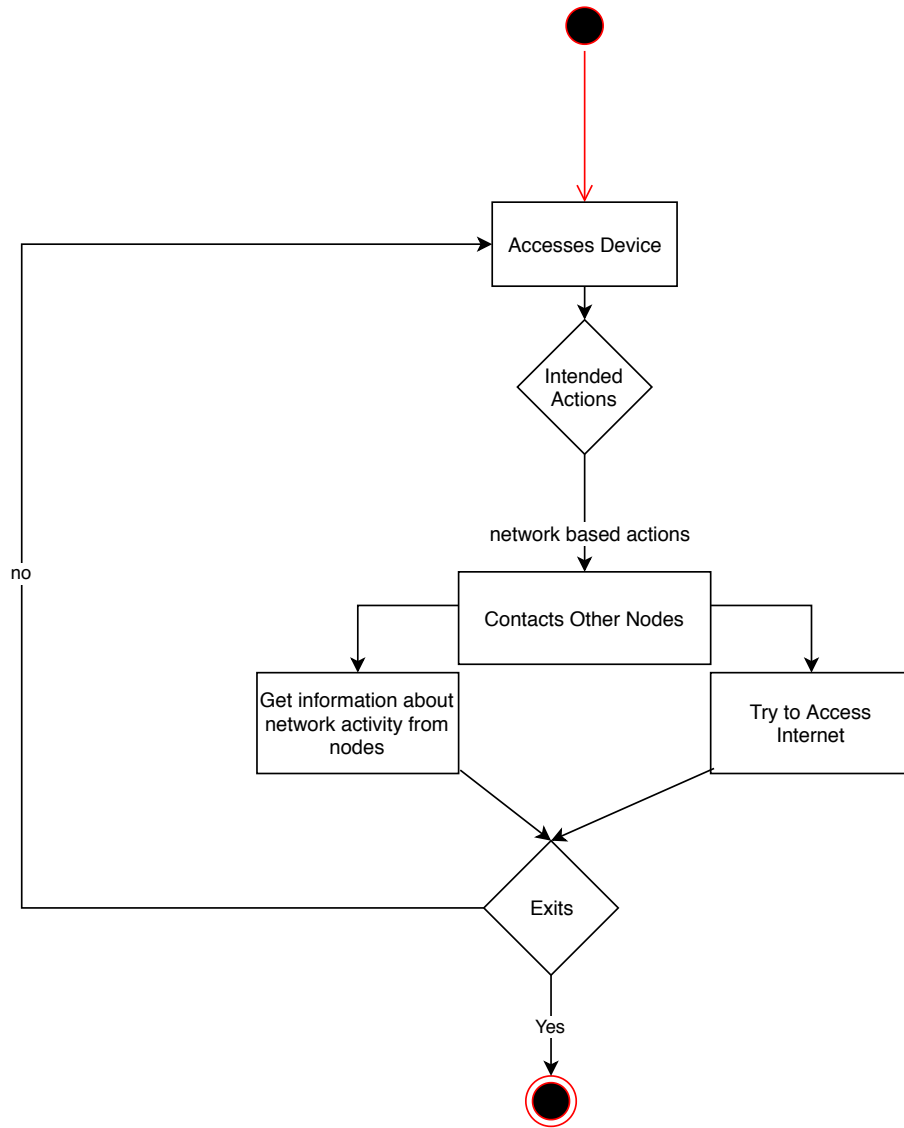


Figure 2.3: User Activity Diagram

2.6 High Level View Of System

Below, you can see the various views that we have in our frontend web application. These views include the main network status page, the network data page, the traffic control page and more.

2.6.1 Network Status

From this page, administrators can view network status. There is a visual display of the status of the nodes, as well as the traffic moving between them. The nodes are labeled with their IP addresses and the edges are dynamically configured based on data from the network. Elements on this page can be clicked on to get more detailed information on each node. The cloud at the top of this view represents the Internet.

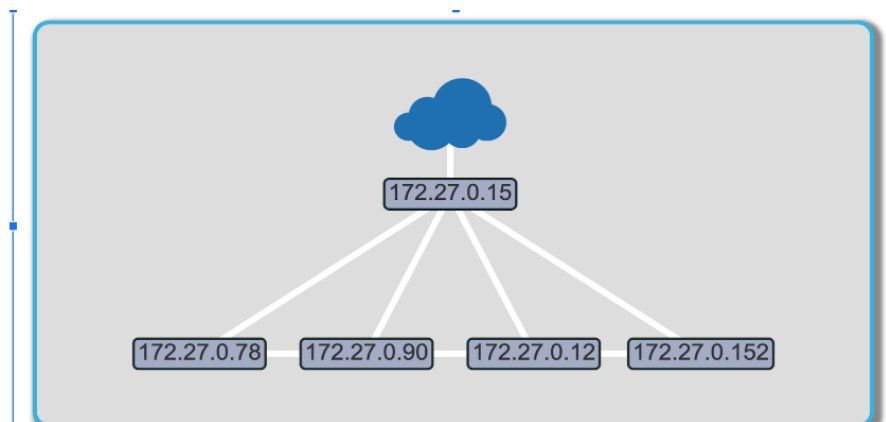


Figure 2.4: Network Status Mock-up Diagram

2.6.2 Network Data

The network status web page (based on the data collected from our software) provides a connectivity map of the network. From this page, users can see the traffic that each node sent and get an idea how the bandwidth is being used. This web page can give a large amount of relevant information, such as how much traffic each node has sent, as well as what protocols the nodes are using. Since protocols can give an indication of what sort of services are being utilized, this is valuable information.

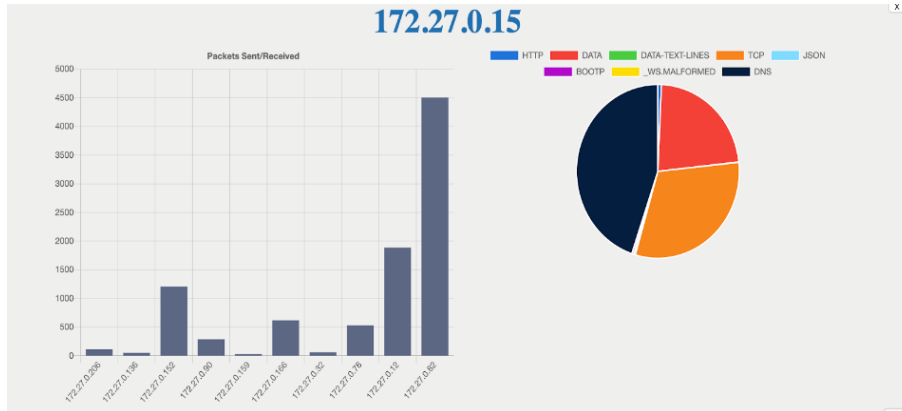


Figure 2.5: Network Status Mockup Diagram

2.6.3 ARP Table Graph

This is a graphic that appears when the individual nodes are queried directly from the API as opposed to viewed via the frontend. It shows how nodes are connected and gives users a cohesive view network topology from the perspective of that node. Many errors in networks (especially mesh networks) relate to ARP tables. If a node cannot find another node, its ARP table will likely reflect that. In an ideal scenario, every node can at least see the gateway node.

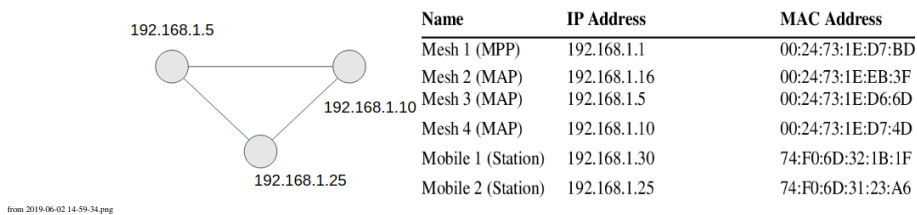


Figure 2.6: Network Status Mockup Diagram

2.6.4 Traffic Control

From this page, the administrators can prioritize IP addresses. Functionally, this system will call utilities like traffic control on the backend of the system. Once the request gets to one node, it is propagated to other nodes in the network. The goal of our system is that users could use this page to determine which nodes are misbehaving and then use the prioritization function to throttle or help those nodes.

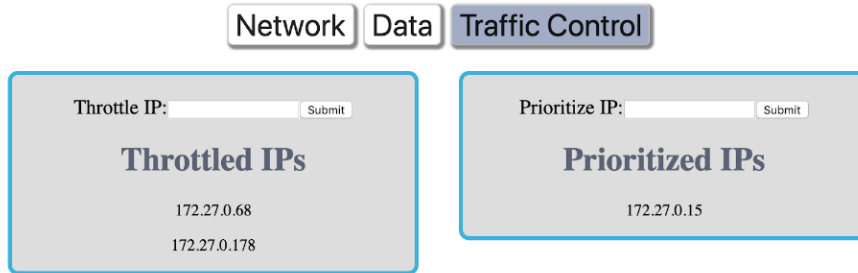


Figure 2.7: Traffic control

2.6.5 Auto Shaping

Our system can automatically shape traffic based on certain heuristics to keep the network running smoothly. This is done through data that we obtained experimentally, as well as with some basic machine learning.

```

{"172.27.0.80": {"comm": ["tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip dst 172.27.0.80 flowid 1:10", "tc filter
mesh0 parent 1: protocol ip prio 2 u32 match ip src 172.27.0.80 flowid 1:10"], "class": "prio"}, "172.27.0.56": {"comm": [], "class":
"}, "172.27.0.82": {"comm": ["tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip dst 172.27.0.82 flowid 1:30", "tc filter
mesh0 parent 1: protocol ip prio 2 u32 match ip src 172.27.0.82 flowid 1:30"], "class": "slow"}, "172.27.0.115": {"comm": [], "class":
"}, "172.27.0.218": {"comm": [], "class": "med"}, "172.27.0.96": {"comm": ["tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match
172.27.0.96 flowid 1:10", "tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip src 172.27.0.96 flowid 1:10"], "class":
"}, "172.27.0.78": {"comm": ["tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip dst 172.27.0.78 flowid 1:10", "tc filter
mesh0 parent 1: protocol ip prio 2 u32 match ip src 172.27.0.78 flowid 1:10"], "class": "prio"}, "8.8.8.8": {"comm": ["tc filter add d
parent 1: protocol ip prio 2 u32 match ip dst 8.8.8.8 flowid 1:30", "tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip
.8.8.8 flowid 1:30"], "class": "slow"}, "172.27.0.90": {"comm": ["tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip dst
.90 flowid 1:10"], "class": "prio"}, "74.125.166.200": {"comm": ["tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip d
:166.200 flowid 1:30", "tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip src 74.125.166.200 flowid 1:30"], "172.27.0
comm": [], "class": "med"}, "172.27.0.116": {"comm": ["tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip dst 172.27.0
d 1:30", "tc filter add dev mesh0 parent 1: protocol ip prio 2 u32 match ip src 172.27.0.116 flowid 1:30"], "class": "slow"}

```

Figure 2.8: Traffic control

2.7 Technologies

To create this platform, we chose the following technologies for their effectiveness, practicality and ease of integration.

- 802.11s

This is our main routing system, though we utilize other wireless standards in the future. Research has shown that 802.11s is a reliable protocol and it serves all of our needs. Another advantage of this protocol is that it is supported by many Linux distributions (for our purposes, it works with Raspbian) and gives mesh nodes access to the full protocol stack with relative ease. (5)

- Raspberry Pis

Raspberry Pis are our main network backbone. They can do many tasks by installing new software and are relatively easy to configure. Their ubiquitous nature has allowed them to become a staple of many fields of computing in recent years. These devices are very adaptable and there is a large amount of online information that helps with problem-solving.

- Alfa WiFi Cards

Alfa WiFi cards can handle a wide range of protocols and are a very reliable choice for networking. They can run 802.11s, as well as batman-adv which makes switching protocols possible if needed.

- Raspbian

This is the operating system on the Raspberry Pis. It is Debian-based and thus easy to work with and to configure. The wide array of support that this distribution has received makes it an appealing choice, as most issues can be easily solved through a simple web search.

- Ansible

It works through SSH and it is used to configure the nodes. This is more of an internal tool rather than a tool for deployment, but it is vital for testing and configuring the nodes quickly.

- Node.js

This is the backend JavaScript run-time environment for our server. Node.js's ability to combine client and server technologies helps us set up apps and features more quickly. It also gives us access to a more extensive library of software in the event that we need to add supplemental functionality.

- Angular

This is the frontend framework for our user interfaces. It works well with Node.js and is fairly easy to use. Angular provides a wide variety of JavaScript packages that allow our application to be easily extended.

- Traffic Control Linux utility (TC)

We use the TC utility for shaping traffic on the nodes. This utility is flexible and provides many different ways to shape traffic. It is particularly useful because it can split traffic from different IP addresses and ports into groups called qdiscs and then manage those qdiscs individually.

- Python

This is the language that we use for data analysis on the network. Its wide variety of functionality is very helpful. Python also has the pip (package manager) utility that makes the installation of packages very simple. Python also has many utilities for data science and traffic capturing that makes setting up the system much easier. Notable utilities are PyShark for capturing traffic, NetInterfaces for dynamically configuring interfaces, Python ARP tables for getting a system's ARP table and scikit-learn.

- scikit-learn

This is used for machine learning on traffic data. Most of the learning is done on data extracted from PCAP files. This utility has the ability to quickly make decision trees on data and can be used to determine what sort of entities the users want to throttle in a network.

2.8 Component State Diagrams

2.8.1 Setup of Individual Node

This is the state flow for an individual node. The node boots up and then receives its configuration from the main node. The main feature in this is that it will either be a gateway or not be a gateway. If a gateway node's external Internet goes down, then it becomes a non-gateway. To change its configuration, it must reboot.

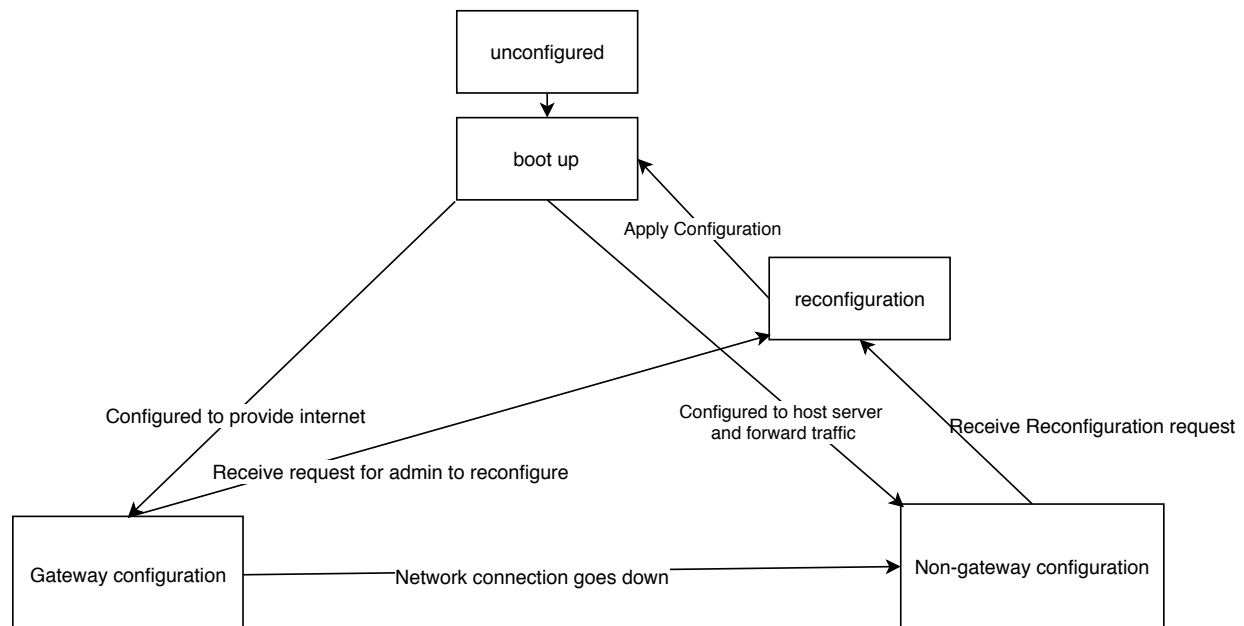


Figure 2.9: State diagram of node

2.8.2 Admin State Flow

The first step of the admin console is that it boots up. From there, it tries to ping other nodes. The console then obtains information on other nodes and sets up the console for configuring them. In the event of a network outage, it tries to ping the nodes again.

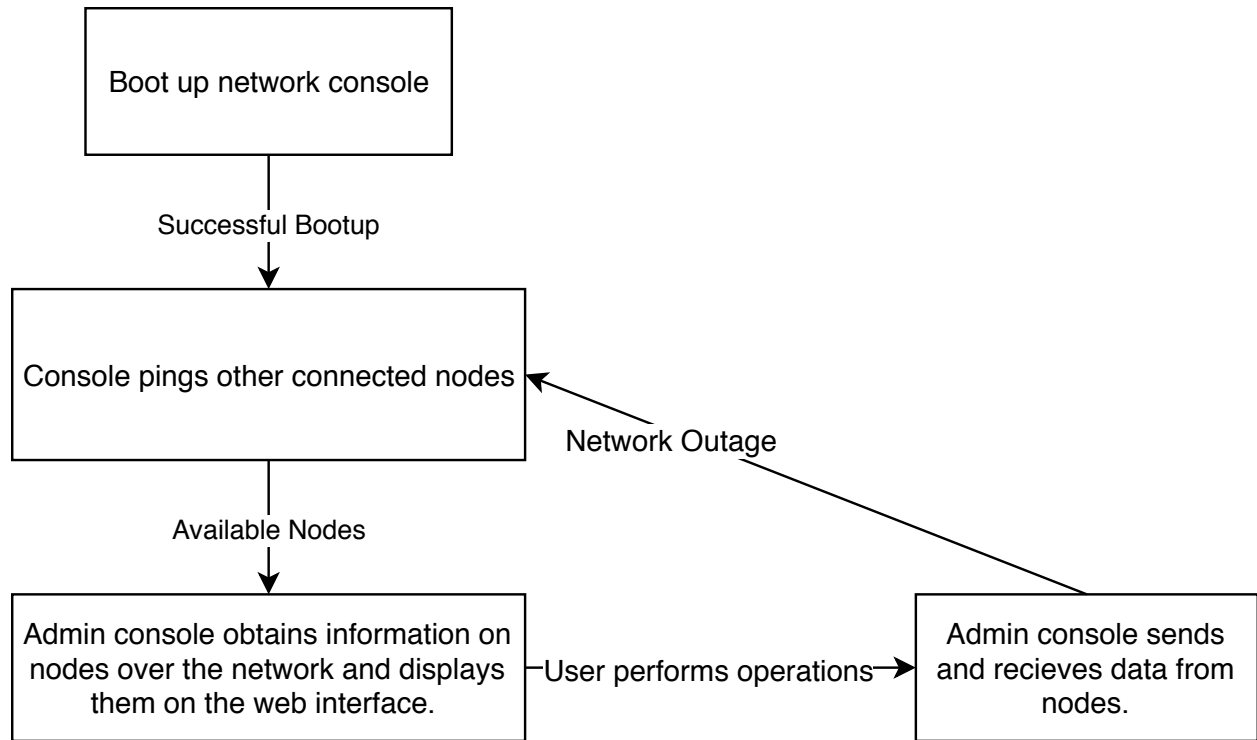


Figure 2.10: State diagram of admin console

Chapter 3

Implementation

3.1 Architectural Diagram

At a high level, our system is about collecting and managing data. We need to manage users' access to the Internet. We also need to manage the data that individual nodes collect and aggregate it. To that end, our architecture is somewhat data-centric. Below is a diagram showing our solution's architecture. As stated earlier, we have three kinds of nodes: gateway nodes, regular nodes and access points. Nodes are able to send data between themselves via our Node.js. Nodes store their own data locally on files that can be accessed and transferred later through APIs. Users on their devices are able to send and receive data on their network when connected to an access point node. Additionally, there is the admin controller (the laptop in this diagram), which provides access to the web interface. The admin controller receives data about other nodes and also has the ability to control and configure these nodes remotely. It is worth noting that, while the core nodes require mesh capabilities, clients connected to the system and the admin controller do not.

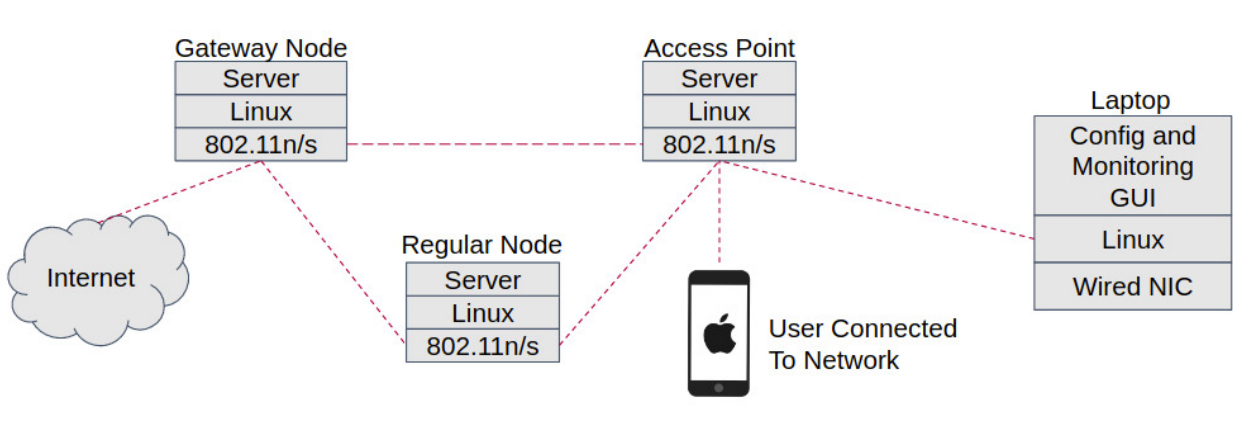


Figure 3.1: Architecture of Platform

3.2 Design Rationale

3.2.1 Technologies

We used 802.11s because it is a reliable protocol for getting things up and running. Studies show that 802.11s is more reliable than other protocols such as batman-adv. Thus its usage makes sense for this project. Additionally, it takes care of many of the simple routing operations and allows us to work at higher layers of the network stack.

The Raspberry Pi is a versatile piece of hardware. Our group also made use of OpenWrt systems for some tasks, but our main backbone is constructed with Raspberry Pis. This is due to Raspberry Pis having a wide array of software that they can handle almost any task. With a simple "apt-get" command, they can quickly and reliably install needed software. This allows us to readily configure all layers of the network stack, as well as make use of tools like traffic control.

Alfa wireless cards are our main method of mesh access (though this is subject to change). Alfa wireless cards are reliable and can handle a wide variety of protocols.

Node.js allows JavaScript to be run on the backend and works well with Angular on the frontend. The diverse library of Node.js modules makes setting up our web server more convenient. The flexibility of Node.js also complements the flexibility of our hardware and network architecture. By using Node.js, our system is future-proofed, and it is more adaptable to change based on a client's needs.

Python has so many different libraries - everything from data analysis to traffic control - that we use it for several tasks in our project. Python is used to parse network data and to quickly set up machine learning models for the system. Scikit-learn is a more lightweight tool that can set up simple machine learning models. It allows us to effectively parse traffic and generate meaningful data without worrying about whether we implemented the algorithms correctly. This allows us to gather insights about network traffic.

Another advantage of Python is the ease with which it can interact with a variety of data formats and structures. JSON is a data format that is natively built into Python, so we did not need additional packages for it. Since JSON is ubiquitous to both Python and Node.js, it made sense to choose it.

3.2.2 Layout

We have a number of pages - including a network management page, a current state of network page and various data entry pages. The design is simple and lightweight, and it provides an intuitive graphical layout of the network's status. We chose to emphasize simplicity over complexity in order to ensure a network admin can see valuable information without wasting time.

3.3 Description of System Implemented

3.3.1 Nodes

Nodes have a large number of features and are the backbone of our network. Every node has a small Node.js server running on it that handles API requests, as well as managing the data transfer between the nodes, running Python scripts and reading/writing to files.

- Traffic

Nodes communicate traffic logs through API endpoints setup by Express.js, a package for Node.js. From there, they can parse received data, add to it with their own traffic that they have forwarded, or execute traffic shaping scripts.

- Shaping of Traffic

Nodes shape traffic through the Linux program "TC". This utility has a wide array of functions that allow for the shaping and management of traffic.

- Data Exchange

Nodes exchange data intermittently so that each node can have an up-to-date record of the traffic being sent through the network.

- Data

Nodes store data in files on the server. The data itself is in JSON format. We did not see a need for the size and overhead of a central database.

3.3.2 Frontend

Unlike the nodes, the frontend for our system can be retrieved on any device that is connected to the network. The frontend queries the gateway node for data which contains metrics on the network as a whole. It can also send traffic shaping commands in the form of IP addresses that should be throttled or prioritized. The frontend is built with Angular which is a JavaScript framework built on top of Node.js.

- Graphics

The frontend can take traffic data from the gateway and display it as a graphs that show network admins how their network is being used.

- Node Graph

The frontend shows how nodes are connected in a graph representation. This allows for admins to view their networks in a very intuitive way.

- Traffic shaping

The frontend can shape traffic and control what sort of packets are prioritized or slowed.

3.4 Node Connectivity

Nodes communicate via the HTTP protocol. They exchange data such as the traffic that they have forwarded. Each node has a server implementing an API that other nodes can query that allows traffic data to be exchanged.

Chron tasks are set up to run various Python scripts. These scripts do a number of things such as:

- Getting traffic from other nodes
- Querying the ARP tables of other nodes
- Generating a local graph of traffic on their system that can be queried in the event that there is no frontend that can be accessed
- Generating a graph of their ARP tables
- Querying the traffic control directions of other nodes

3.5 Networking Protocols

- Hostapd: Hostapd is the software that we use to turn the Raspberry Pis into WiFi access points.
- DNSMASQ: DNSMASQ is software that we use to run a DHCP server. Typically we have one DHCP server running on a gateway.
- Bridge-Utilities: We use bridge utilities to bridge the WiFi network into the mesh network. This allows for every node to be on one IP address range.

3.6 Traffic data

Traffic data is stored in a JSON object and we chose this approach to save storage space. By parsing the data and storing the parsed result - as opposed to storing the entire capture in a PCAP file - we save on storage by only storing the relevant pieces. The keys of this dictionary are IP addresses of mesh nodes. The keys then map to a nested dictionary that contains more information such as:

- Destination

The dictionary stores the destinations that a node sent traffic to, as well as the number of packets that it sent to that destination. This is another useful statistic to tell network administrators how their bandwidth is being used.

- Time

The time in which a particular packet was sent is also stored in the destination field of the dictionary. This is primarily to give users an idea of when traffic was sent, but in our current implementation it is not heavily used.

- Route

This is a field in the data that essentially shows which node routed a certain node's traffic from source to destination. This helps network operators differentiate between core and edge nodes. This statistic is also used in the route count statistic.

- Protocol Dictionary

The protocol dictionary is a dictionary of the protocols that a particular node sent and how many times it was used. This information gives insight into who is behind the node. This can also indicate whether encrypted protocols are being used. SSL and HTTPS indicate that a user is safely encrypting traffic.

- Length

This is the average length of packets and is useful for determining what sort of website a user is visiting. A larger average length implies large data transfers.

- Count

This is the count of packets that a particular node has sent.

- Class

This is the type of node that the traffic recording script determines it is looking at.

3.7 ARP Data

ARP data is stored in a key-value dictionary format where the keys are the nodes and the value list is the IP addresses that the nodes can see. This is a much simpler structure than the traffic data but that is because less information is relevant. All that is needed for the graph is the interconnection of nodes so that is what is stored. This data can be parsed into a directed graph and displayed via the API or the frontend application.

3.8 Traffic Control Result

This data is a similar format to the ARP graphs with a key-value format. However, in this case, the values are the result of the forwarding operation. The main purpose of this data is to allow network operators to verify that nodes are making the correct assumptions about traffic shaping.

3.9 Algorithms and Techniques

3.9.1 Decision Trees

We use decision trees with the Python utility scikit-learn to do machine learning on packet data. The decision trees look at various factors of the packets such as what nodes they are connected to and how much traffic they have sent. From there, they can make decisions to determine how to throttle or prioritize nodes. The system can also take input from the admin, via the frontend application, to throttle and control traffic.

3.9.2 Statistics and Routing

One of the most important statistics that we use for shaping traffic is the route count. The route count is the number of nodes that a particular node routes traffic for. This statistic, more than any of the others, allows us to determine which nodes are necessary to the network and then throttle traffic accordingly. This is relevant for decisions such as making sure central nodes are able to send traffic and communicate.

Another statistic that we track is the average length of packets. This allows us to see the kind of traffic that a node is sending. This, along with packet count, helps us identify the biggest users of bandwidth. For example, this statistic helps us identify IPs that may be streaming videos or downloading large files so that we can them to ensure important traffic can get through the network.

3.9.3 Classification

Our system can classify nodes based on their behavior and what sort of traffic they send. The main classifications are as follows.

- Node
A basic node that is forwarding traffic.
- Bridge
An access point node that bridges regular nodes and user devices into the mesh.
- Gateway
A node that connects to the larger internet.

What is tricky about node classification is that our system only classifies nodes by the traffic that it sees and its ARP table. This is because we wanted to make a system that does networking from as high of a level as possible.

To that end, we can use the route count statistic stated in the previous section to classify nodes. The route count allows the system to determine which nodes are forwarding traffic. As a general rule, nodes that are forwarding traffic through our mesh are important to the network and should not be throttled extensively. Additionally, nodes that are

forwarding traffic are likely either gateways or bridges. The main focus of throttling is to ensure we only throttle exterior nodes that are using large amounts of data and not throttling our interior nodes.

Another statistic that is looked at for classification is a node's ARP table. Nodes that can "see" more and interact with more nodes are central to the network. From this data, we can determine whether a node is a gateway, bridge or a user device. Thus, the number of ARP nodes is another statistic for node classification.

3.9.4 Heuristics

In addition to the machine learning, we use some heuristics to shape traffic. The simplest of these heuristics is checking whether a node's packet count is past a certain threshold, but there are other important statistics. Among these is the protocol count of a node - or the number of different protocols that it has. This shows the diversity of traffic that a node is sending.

3.10 Frontend Information

3.10.1 Sent Traffic

The frontend displays collected data to users. When selecting a node from our home page, seen in previous sections, the user can see how much traffic is sent by a node, where the traffic is going and which protocols are used. We put this data easily accessible because this is the most important and useful data to an admin. The network traffic is displayed in a bar chart seen in figure 15.1. For the sake of an example, if the network admin saw the large disparity of data being sent by the IP ending in .82, it would make sense to throttle that IP via our traffic control menu. The network protocols used are seen in a pi chart. The pi chart is interactive and by clicking a label in the legend, you can hide that protocol. This way, one heavily used protocol does not hide the breakdown of less used protocols. These graphs were created with the Chart.js library.

The breakdown of the protocols can be useful in a number of ways. For instance, a node that is predominately sending HTTP is likely making a large number of web requests. If a user is using SSH, they are likely configuring one of the nodes. It is information such as this that gives insight into network activity in a very unique way.

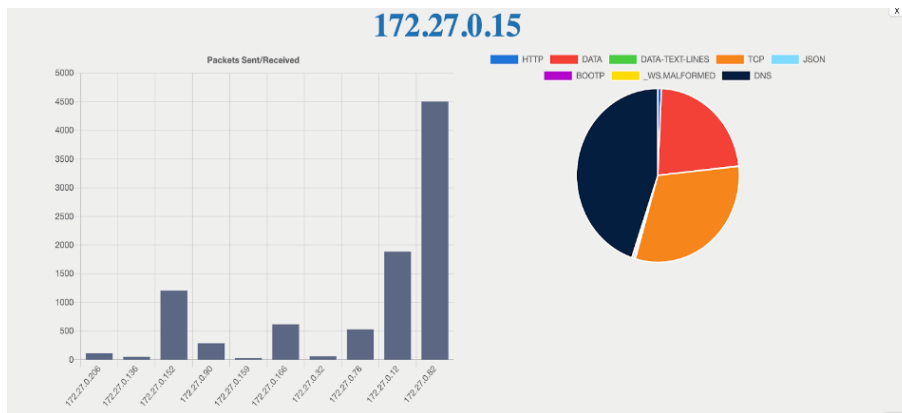


Figure 3.2: Network Status Mockup Diagram

3.10.2 Frontend Networking

As previously stated, the frontend works as a stand alone system and does not need mesh capabilities. In order to have the frontend available, all we have to do is create a build of the Angular project and then serve the project over an HTTP request to the node that is hosting it. The frontend is then able to gather the necessary data via HTTP requests to our API endpoints exposed on the gateway node. It takes the data files received from the API and parses them into data structures such as Arrays and Hashmaps. These data structures are the building blocks of our graphs, such as the one in Figure 15.1.

3.11 Node Functionality

While there is a main frontend that users are encouraged to utilize, another feature of our system is the ability to contact nodes directly. While this feature is used primarily for nodes to communicate between themselves, there are other use cases and actions that users can take with the nodes.

3.11.1 Graphical Images

The nodes themselves can display graphical images of data to users. This behavior is done by Python scripts on the backend, and it only displays images to keep the system lightweight. While not as powerful as the frontend, the images can give users valuable insights into their network. The types of images are listed below.

- ARP table graphs

ARP table graphs show the connectivity of nodes. They are rendered as a simple PNG image to keep the system light weight. As previously stated, these graphs show users a node's view of the network. From there, the user can more quickly diagnose errors.

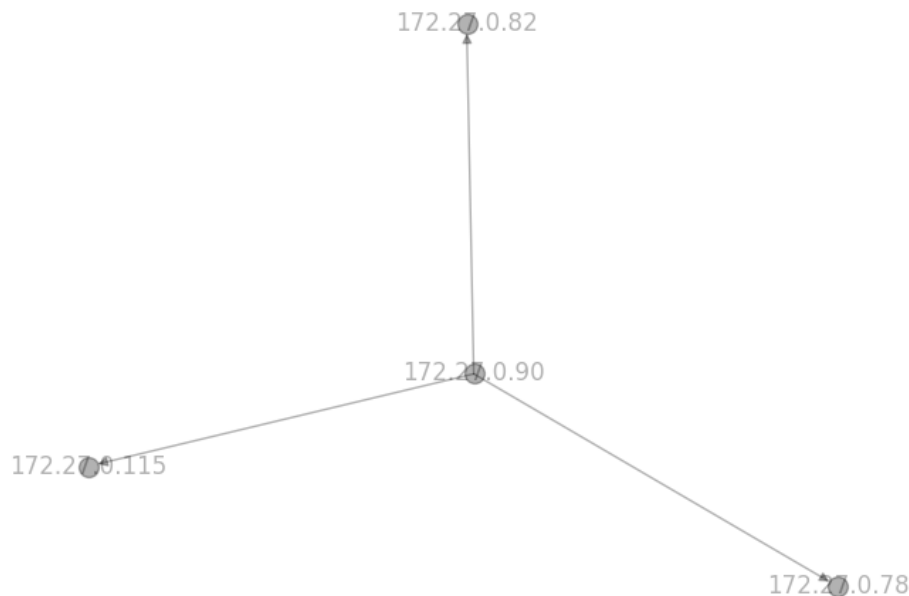


Figure 3.3: Network Status Mockup Diagram

- Traffic Data

Nodes can generate and display a graph of how much traffic they are sending - similar to what is viewed on the frontend in a picture form. The graph is rendered as an image to save space. These graphs can quickly give a

user insight into how their network is behaving.

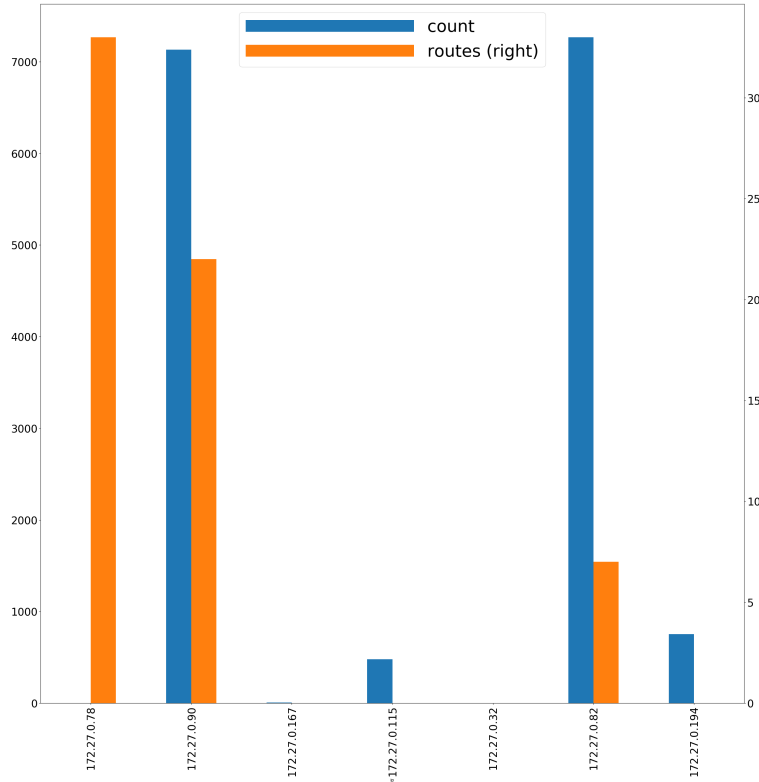


Figure 3.4: Network Status Mockup Diagram

3.11.2 Data

- Tc Info

The frontend can display the results of a traffic shaping operation and show users how it classified nodes. This helps users know if the network has made any assumptions about shaping that they should attempt to rectify.

- Inputting Traffic Control Info

There is an endpoint on the nodes for receiving traffic control information from the admin so that it can apply the changes.

- Listing Other Routes

The node can list the available routes that it has. This helps give users a clear idea of a node's capabilities.

- Getting Traffic Data

Network users can retrieve the raw traffic data in the JSON data format.

- Getting the ARP Data

Network users can get a node's ARP data (that is used to make the ARP graph) in a raw JSON format.

3.12 Example Work Flow of System

The following is a time-line for the system running and a description of its processes.

1. The server on all of the nodes is started.

The Node.js instance that runs all of the nodes is started. It is the Node.js backend that manages the cron tasks of the server.

2. The nodes receive traffic and parse it.

A cron task is run that collects traffic. This task is simply a Python script that collects traffic and parses it into a JSON format that we determined to be effective. This also helps to remove extraneous information from the traffic and save space.

3. The nodes check their ARP table and convert it into the JSON format.

This is done through Python scripts. A particular node is seen as a node on a graph and its ARP table is its connections.

4. The nodes shape traffic.

The script to shape traffic is run and nodes prioritize or throttle traffic according to a variety of heuristics, as previously discussed.

5. The nodes exchange traffic with each-other and generate graphics

These graphics in particular are primarily images.

6. The nodes receive traffic and generate graphics again, but this time with the information from other nodes.

The information from other nodes allows nodes to make more informed decisions about their network.

3.13 Test Plan

3.13.1 Verification

Verification of the product is tested by checking that data is transferred correctly. Tests have been done on files and transferred data to ensure that they have remained intact. Data is sent through the network to test that connectivity is working. Most of this testing essentially involved verifying that our low level protocols integrated well with our high level protocols and that our networking protocol worked correctly.

- Traffic sent by client devices will be sent correctly.
- Files will be transferred without error.
- Nodes will be configured properly and will alert the network if they are not.

3.13.2 Validation

The design has been tested against all of the requirements and constraints. Unit tests from the requirements have been performed to ensure the correct functionality. The product has been compared to the activity diagrams to make sure that it behaved as expected.

- The nodes on the network are mesh nodes with the ability to forward traffic themselves.
- The system has the ability to notify users if there is an error.
- Work flow is consistent with activity diagrams.

Chapter 4

Concluding Remarks

4.1 Challenges Encountered

Throughout this project, we faced many challenges integrating various features into our system.

4.1.1 Technology

- **Routing:**

One of the biggest challenges (that ultimately proved insurmountable) was to get dynamic routing (and by extension - load balancing) across all of the nodes. Part of the reason for this problem was that the protocol worked at a very low level so controlling it was difficult.

- **Maintaining a Cohesive Record of Traffic:**

Because traffic could be sent through multiple nodes, maintaining a consistent record of where all of the traffic was going proved to be a challenge.

4.2 Technology

- **Dealing with external workload increase:**

There were many times where the workload from other classes increased to the point where we were not able to work on the project for periods of time. This forced us to compromise on many features.

4.3 Suggested Changes

Overall, our mesh network was built to be flexible. This had a number of advantages - mainly an easier end user setup. However, this approach also had problems.

4.3.1 Node Functionality

- Individual Node Frontend:

In our current setup, nodes have an API endpoint that can be accessed in the event that the frontend is not available to get data. A potentially effective change would be to have more frontend functionality for each node in order to make the system depend on a single component. This could consist of dynamic graphs like what was included in the main frontend.

- Recognizing the Frontend:

4.3.2 Frontend Functionality

- Shaping traffic:

The frontend should possess expanded ability to shape traffic in diverse ways. Currently, it primarily works with IP addresses, but there are many changes that we could make to further extend the functionality of the frontend.

4.3.3 Core System Changes

- BATMAN-ADV:

While 802.11s served our needs well, batman-adv could potentially serve our needs better. This is because it has built-in API functionality and is put together in a way that makes the system easier to modify.

- Lower Level Traffic Control:

While the TC utility is both flexible, easy to install and effective, it does not look at protocols at a low enough level. For this reason, other methods of shaping traffic could potentially be more useful. A utility that could interact more closely with the protocol itself could make our system much more extendable.

4.4 Lessons Learned

4.4.1 Implementation

- The finer details of the implementation of this project should have been determined in advance.
- Features like traffic control, while functional, were implemented in a way that could have interacted better with the rest of the technology.

At the start, we looked at certain aspects of this project at a very high level. Because of this, we attempted to accomplish them in the simplest way possible. While this worked, it later constrained us in other ways. For instance, because of the way we implemented features like shaping, other features like load balancing became difficult to implement without extensive editing of other components. Were we to do this again, we would handle things differently. We would try to bridge low level components directly to high level components.

4.4.2 Team Coordination

- A more tight meeting schedule would have likely increased our productivity.

Initially, we planned our project anticipating an increased workload from schoolwork, but we could not have anticipated the amount of issues that would derail us. It would have helped to try and make more specific test cases during fall quarter. This would have moved the project along and we would have had a clearer path forward. Earlier on, we should have focused on aspects like data storage at a more specific level and built test cases around that.

4.5 Ethics and Societal Impact

4.5.1 The Politics of Ownership

Currently, wireless networks are controlled by large corporations that have the power to determine what content can and cannot be accessed. In contrast, mesh networks hold the promise of wireless networks that are owned and operated by individuals.

The fundamental issue with ownership of communication infrastructure today is that it is controlled. Control has been proven to be an issue when determining what traffic should be sent through a network. This debate has come up in numerous occasions such as the recent debate over net neutrality. The issue of ownership and control of networks is multifaceted and has a large number of stakeholders.

Mesh technologies enable an alternative community-based ownership model, which in turn enables a different control model - one that is driven by the needs of the community, not investors. Most of the mesh technologies discussed provide different ways to manage resources and decide what traffic should be prioritized. However, the most common criticism of group-based decision making is that good intentions may still lead to bad outcomes. For example, a user might prioritize their traffic and unintentionally impact another user. People will often prioritize according to their own self-interest, and this could lead to misuse of resources. Thus, a secondary goal of these technologies is to ensure the fair sharing a network resources. The main opportunity of mesh networking is to create a network that is communally owned - that is, owned by individuals and not large corporations.

4.5.2 Privacy

One concern that could be brought up about our system is that users lack privacy since members of the mesh network have visibility into everyone's traffic. There are many nodes - each of which could be operated by a different person. While privacy is a concern, the system does allow encryption. More importantly commercial ISPs look at traffic today and consumers have little or no insight to how their data is used. As the recent issues with Facebook have shown, sometimes a big company will sell targeted data without anyone's knowledge or permission.

4.5.3 Throttling and Fairness

Traffic management is an important component of commercial wireless networks. One criticism of commercial ISPs is that they can throttle any traffic they wish. This can be done either to prioritize services (like video streaming) that they typically profit from. That said, a key component of our mesh project is that it throttles traffic. However, the main difference in our use of traffic management vs commercial networks is that we are focused on the equitable distribution of network resources and not profit. Our traffic shaping features could be tuned to prevent network misuse and keep these networks from getting congested by greedy users.

Mesh networking is based on ad hoc connections and thus traffic management is critical to smooth operation. The big issue then becomes how to throttle traffic and manage networks in an equitable way. The solution that we found was to monitor the number of packets sent, but there are other solutions and other fields of packets that can be looked at. However, sometimes it is difficult to inspect packets because they might be encrypted. This brings up the issue of control. Do you trust how users are using the bandwidth of this network node? The answer to this question depends on a large number of factors that vary on a case-by-case basis. Thus, being aware of which users are generating the most amount of traffic is critical. In our project, we developed a real time monitoring application to allow us to see the results of our algorithms.

4.5.4 Environmental Sustainability

Due to the low power requirement to run Raspberry Pis, our system has the potential to make networks more sustainable. For example using solar panels to power the nodes would allow its deployment in areas that have little or no infrastructure such as remote sites or in emergency situations. Conceivably, our mesh network could help people in remote areas by powering nodes with solar panels and creating wireless relays to the public Internet.

4.5.5 Economic Sustainability

This project also has the ability to make wireless access to the Internet much more economically viable and thus sustainable in remote or low income areas. That is because the low cost of the Raspberry Pis could make infrastructure affordable in scenarios where it is currently not profitable. For example, rural America has suffered from poor cellular coverage as large corporations do not see enough profit to justify deployment. This happens because there is not enough density of users to justify a wireless network. In contrast, community-based mesh networks have no profit objectives.

4.5.6 Societal Impact

Making information more accessible to all citizens in a society empowers people to make better decisions. Mesh networks allow people in unconnected areas to communicate and organize themselves. By allowing users to host their own web servers on their mesh network, communities could create their own low cost wireless services without the need to work with a large corporation.

4.5.7 Usability

Ordinarily, traffic management is considered to be a complex concept. Thus, one of the important aspects of our project was to create a simple web interface that anyone can understand and configure. One of the most important insights we gained during this project was that knowledge is critical to effective control. Whether a network is controlled by a corporation or a community, if you don't know what's going on, then you don't know what to change. A second

insight was that all of the technologies must be integrated to enable large scale deployment. At least some insight into the issues that networks face is necessary in order to use them effectively and diagnose errors. Finding users who both need the technology and have the know-how to use it can be a challenge, but as more groups acquire technology skills, this should be doable.

Another consideration is that some societies prefer a simple life and do not want technologies such as this. There are people that do not want to connect to the Internet and don't feel like they need computers. In these settings, more education is needed in order to truly exploit the potential of mesh networking technology.

4.5.8 Lifelong Learning

Creating a traffic management solution for a mesh network required us to integrate a host of technologies. Previously, we studied these technologies individually, so getting them to work together was a big effort. But despite all of our work, there is still much more to do if our ambition is to support large scale deployment. To create something unique is hard work, and we learned the importance of continuous effort.

The technologies used in this project change constantly. Every year, new protocols are being developed or modified. Thus, one must be willing to keep up with the latest protocols and network techniques in order to be aware of what users want. Our software implements many techniques and uses many tools from a variety of sources. In this way, it has prepared us for the modern software universe in which a large number of technologies work together.

4.5.9 Compassion

One of the biggest issues in our society is the lack of communication. Time and time again, we see groups of people in our society that are isolated or marginalized. At its core, computer networking is about connecting groups and devices. Our technology has the ability to connect groups of people and improve their ability to coordinate. Since our technology uses a full protocol stack, it is compatible with most web software - meaning users can host almost anything. In this way, we have the ability to make the world much more compassionate by allowing groups to communicate and share their issues. This also extends to wireless sensors. When people have access to information, they become aware of problems in the world and can work to change them. Overall, our technology can make the world much more connected and compassionate, while preventing misuse of networks.

4.6 Conclusion

In this project we developed a web-based manager to monitor the mesh network and shape traffic within it. The backbone of this network was based on Raspberry Pi computers and Alfa WiFi cards. Each of the Raspberry Pis runs a Node.js mini server that schedules tasks such as collecting network data and shaping traffic.

The biggest lesson from this project was the importance of making complex technology easy to understand and control by users. Traffic management systems incorporate many layers of technology across the protocol stack. But unless the activity inside those layers is exposed in a way that is easy to understand, it is of no value. More importantly, users need a way to interact with the system. All of the technologies mentioned have their own logic models and capabilities. We chose to display the network performance via a web interface to simplify monitoring and management tasks.

The concept of layers is integral to mesh networks because their protocols operate on different layers of the TCP/IP stack. Software and operating systems implement tasks in distinct ways that interact with other components and their stack of layers. This, in turn, causes the entire system to change its behavior. In addition, mesh protocols also have their own distinctive techniques to interact with compute devices. Opting to look at things at a high level of the protocol stack has given us access to powerful utilities and technologies. However, it also has its problems - such as forcing us to interact with hardware level components that, at times, have difficult to use APIs. Ultimately, this project showed that mesh networking can be approached in a multitude of ways, and that each of these approaches has its own benefits and challenges.

This however relates to one of the major disadvantages of our approach: we lacked ability to view the bottom of the stack - the physical interfaces. Ideally we would have wanted to control the WiFi signals however the web-interface made that difficult. While there were utilities in Linux that supposedly had the ability to monitor antenna performance, they never did quite what we wanted. This was an issue that technologies like batman-adv could potentially have solved.

Looking to the future, utilizing machine learning to shape traffic would seem like an ideal approach to improve the performance of our system. Machine learning would make the system more flexible, while utilizing the same core components. This could be accomplished by implementing deep learning based systems or perhaps just coding more sophisticated algorithms directly. However, other modifications to the software are more challenging. One of these modifications is switching protocols from 802.11s to batman-adv. Most of the server should still work after this switch since both protocols have a full protocol stack, but there are other considerations. For one, batman-adv has many powerful APIs, but the server would need to be configured to run them. Moreover, features like traffic shaping would potentially need to be reconsidered.

As a final takeaway, mesh networking - more than any other type of networking - is use case dependent. Each

potential client will likely have needs that are specific to them. Thus, it is necessary to have a flexible system that is easily extendable. Our project has both of these features, and it served its use cases well. We did achieve most of the project objectives, such as the ability to monitor and manage traffic in real time, and we did this in way that can be easily extended. However, real time mesh traffic management would require more development before wide scale deployment is possible and I hope that this paper shed light on that.

Appendix A

User Manual

This is a guide for running the system.

A.1 Setting Up Nodes

Nodes need to be configured to start up their interface and connect to a single ESSID upon boot up. They also need to be configured to obtain an IP address through DHCP and forward traffic.

1. For each node, make sure that it has an 802.11s capable device.
2. Run these commands to start 802.11s
3. Run: `sudo iw dev wlan1 interface add mesh0 type mp mesh_id MYMESHID`
4. Run: `sudo iw dev mesh0 set channel 4`
5. Run: `sudo ifconfig wlan1 down`
6. Run: `sudo ifconfig mesh0 up`
7. Run: `sudo ip addr add 10.1.100.10/24 dev mesh0`
8. Install bridge utilities
9. Run: `sudo brctl addbr br0`
10. Run: `sudo brctl addif br0 mesh0`
11. If the node is an gateway run `sudo brctl addif br0 wlan0`
12. If the node is a gateway check that it can forward traffic to the internet through network address translation and that it is running a dhcp server. This will automatically point traffic toward it.
13. Clone the code from github
14. Install the requisite python utilities and node.js utilitites
15. Standard configurations of hostapd and dns masq can be used with the system. The ideal setup is to have DNSMASQ running on the gateway server and to have hostapd clients bridged in through bridge utilities. This setup allows for all nodes to be on one ip address range.
16. To start a node run `sudo npm run start`

The main items that are important for the network are that access point nodes are running Hostapd and that the bridge utilities are running on those computers to bridge them into the nodes connected by the mesh. The nodes also have to be configured so that they can connect to the public Internet via one of the nodes on the mesh. If these considerations are addressed, the server should work.

The process for installing Python dependencies will also vary greatly. The best way to get all of the dependencies is to repeatedly try and run all of the Python scripts and install the utilities that it says are missing. Search Google for instructions on how to install these utilities, as the process for installing them varies greatly.

A.2 API

The API is configured to run on port 3030 of the nodes. By going to port 3030, a list of valid APIs will be provided. From there, to get the desired information all that is required is to go to the URL in a browser. Example: `curl 172.27.0.15:3030/getFile` could be used to get a nodes traffic data.

A.3 Frontend

The computer running the frontend does not need to be 802.11s capable. It does, however, need to run Node.js with the requisite dependencies and be able to contact one of the mesh nodes (preferably a node running a DHCP server).

Set the IP address for the frontend in all of the configuration files that specify it. From there, the frontend will know what node to query for its information.

The frontend can be easily navigated from the bar at the top. To install the frontend, clone the GitHub repository and run `sudo npm install`. From there, the steps to install the dependencies will vary greatly.

Appendix B

code

B.1 routes.js

```
1  const express = require('express');
2  const path = require('path');
3  const fileController = require('../Controllers/fileController.js');
4
5  const fs = require('fs');
6  var os = require('os');
7  var ifaces = os.networkInterfaces();
8
9  var ip = '172.27.0.82';
10 var chosenAddress='172.27.0.82';
11
12 module.exports = function(app, express) {
13   console.log("running router script");
14   let router = express.Router();
15   // server ready to accept connections here
16   console.log("runing network interfaces list for ROUTES");
17   //const pythonProcess = spawn('python', ['protocolplot.py']);
18   var networkInterfaces = os.networkInterfaces( );
19
20
21   console.log( networkInterfaces );
22   //for(let ni of networkInterfaces){
23   for (var key in networkInterfaces) {
24     //if (networkInterfaces.hasOwnProperty(key)) { // this will check if key is owned by data
25     //  object and not by any of it's ancestors
26     console.log(key+' : '+networkInterfaces[key]); // this will show each key with it's value
27     for(k in networkInterfaces[key]){
28       console.log(k+' : '+networkInterfaces[key][k]);
29       /*for(l in networkInterfaces[key][k]){
30         console.log(l+": "+networkInterfaces[key][k][l])
31       }*/
32       if("address" in networkInterfaces[key][k] && k==0){
33
34         console.log("address is "+networkInterfaces[key][k]["address"])
35         var newaddr= networkInterfaces[key][k]["address"]
36         if (newaddr.includes("172.27.0")){
37           chosenAddress=newaddr;
38           ip=chosenAddress
39         }
40       }
41     }
42   }
43 }
```

```

40     /*for(let el of networkInterfaces[key][k]){
41         console.log(el)
42     }*/
43
44     }
45     //}
46 }
47 app.on('listening', function () {
48     // server ready to accept connections here
49     console.log("runing network interfaces list for ROUTES");
50     //const pythonProcess = spawn('python', ['protocolplot.py']);
51     var networkInterfaces = os.networkInterfaces( );
52
53
54     console.log( networkInterfaces );
55     //for(let ni of networkInterfaces){
56     for (var key in networkInterfaces) {
57         //if (networkInterfaces.hasOwnProperty(key)) { // this will check if key is owned by data
58             object and not by any of it's ancestors
59             console.log(key+' : '+networkInterfaces[key]); // this will show each key with it's value
60             for(k in networkInterfaces[key]){
61                 console.log(k+' : '+networkInterfaces[key][k]);
62                 /*for(l in networkInterfaces[key][k]){
63                     console.log(l+" : "+networkInterfaces[key][k][l])
64                 }*/
65                 if("address" in networkInterfaces[key][k] && k!=0){
66
67                     console.log("address is "+networkInterfaces[key][k]["address"])
68                     var newaddr= networkInterfaces[key][k]["address"]
69                     if (newaddr.includes("172.27.0")){
70                         chosenAddress=newaddr;
71                     }
72                 }
73                 /*for(let el of networkInterfaces[key][k]){
74                     console.log(el)
75                 }*/
76
77             }
78         //}
79     }
80
81 });
82
83 //setup a function that is listening for request and send back the files that are requested
84 //https://stackoverflow.com/questions/25463423/res-sendfile-absolute-path
85 app.get("/getFile", (req, res) => {
86     var networkInterfaces = os.networkInterfaces( );
87
88
89     console.log( networkInterfaces );
90     //for(let ni of networkInterfaces){
91     for (var key in networkInterfaces) {
92         //if (networkInterfaces.hasOwnProperty(key)) { // this will check if key is owned by data
93             object and not by any of it's ancestors
94             console.log(key+' : '+networkInterfaces[key]); // this will show each key with it's value
95             for(k in networkInterfaces[key]){
96                 console.log(k+' : '+networkInterfaces[key][k]);
97                 /*for(l in networkInterfaces[key][k]){

```

```

97         console.log(l+": "+networkInterfaces[key][k][l])
98     }*/
99     if("address" in networkInterfaces[key][k]){
100
101         console.log("address is "+networkInterfaces[key][k]["address"])
102         var newaddr= networkInterfaces[key][k]["address"]
103         if (newaddr.includes("172.27.0") && k==0){
104             ip=newaddr;
105         }
106     }
107     /*for(let el of networkInterfaces[key][k]){
108         console.log(el)
109     }*/
110
111     }
112     //}
113     }
114     console.log("chosen address is "+chosenAddress)
115     console.log("recieved request for files "+ip);
116     res.sendFile(path.join(__dirname, '../..', 'iplog' + ip + '.json'));
117 })
118 app.get("/editTrafficControl", (req, res) => {
119     console.log(req.body);
120     var stream = fs.createWriteStream("my_file.txt");
121     stream.once('open', function(fd) {
122         stream.write(req.body);
123         stream.end();
124     });
125     res.status(200).send({
126         data: "done"
127     })
128 });
129 app.get("/", (req, res) => {
130     console.log("running router script");
131
132     var response="";
133     app._router.stack.forEach(function(r){
134         if (r.route && r.route.path){
135             console.log(r.route.path)
136             //res.send(r.route.path)
137             response+=r.route.path+"\r\n"
138         }
139     });
140     res.status(200).send(response);
141
142
143     /*res.status(200).send({
144         data: "hello world"
145     });*/
146 })
147 app.get("/getTcDirections", (req, res) => {
148     res.sendFile(path.join(__dirname, '../..', 'tcdirs'+'.json'));
149 })
150 app.get("/getIplist", (req, res) => {
151     res.sendFile(path.join(__dirname, '../..', 'iplist'+'.json'));
152 })
153 app.get("/getImage1", (req, res) => {
154     res.sendFile(path.join(__dirname, '../..', 'output.png'));
155 })

```

```

156 app.get("/getGraph", (req, res) => {
157   res.sendFile(path.join(__dirname, '../..', 'griplog'+ip+'.png'));
158 })
159 app.get("/getProtocolGraph", (req, res) => {
160   res.sendFile(path.join(__dirname, '../..', 'protgriplog'+ip+'.png'));
161 })
162 app.get("/getArp", (req, res) => {
163   res.sendFile(path.join(__dirname, '../..', 'arpfile'+ip+'.json'));
164 })
165 app.get("/getArpGraph", (req, res) => {
166   console.log('graph'+ip+'.png')
167   res.sendFile(path.join(__dirname, '../..', 'graph'+ip+'.png'));
168 })
169 app.get("/getTcInfo", (req, res) => {
170   console.log('graph'+ip+'.png')
171   res.sendFile(path.join(__dirname, '../..', 'tcdirs.json'));
172 })
173 app.get("/getShapeFile", (req, res) => {
174   console.log('graph'+ip+'.png')
175   res.sendFile(path.join(__dirname, '../..', 'shapefile'+ip+'.json'));
176 })
177 var bodyParser = require('body-parser')
178 //app.use(express.bodyParser());
179 app.use(bodyParser.json());
180 app.post('/recieveTcInfo',function(req,res){
181   //var user_name=req.body.user;
182   //var password=req.body.password;
183   //console.log("User name = "+user_name+", password is "+password);
184   //res.end("yes");
185   console.log(req.body.toString())
186   console.log(JSON.stringify(req.body))
187   console.log(res.body)
188   fs.writeFile('./tcdirs.json',JSON.stringify(req.body), function(err) {
189     if (err) {
190       console.log("E!");
191     }
192     res.end("failure");
193   } else {
194     //fs.writeFile('./iplog172.27.0.90.json', body, function(err) {
195     //console.log("file saved "+ipname.toString());
196     //console.log(ipname)
197     res.end("success");
198   }
199 })
200 });
201 }

```

B.2 filecontroller.js

```

1 const cron = require('node-cron');
2 const path = require('path');
3 const sh = require('shelljs');
4 const {spawn} = require('child_process');
5 const fs = require('fs');
6 const request = require('request');
7 var gotResponse=1;
8 const mainServerList = ['172.27.0.90'];
9 var ipList = ['172.27.0.78','172.27.0.90','172.27.0.82','172.27.0.12'];

```

```

10 var os = require('os');
11 var ifaces = os.networkInterfaces();
12 var chosenAddress='172.27.0.82';
13
14
15 //each server on the nodes will intermittently send http requests out to every other node and ask
    for them to send their files back
16
17 //setup a function that is listening for request and send back the files that are requested
18 //https://stackoverflow.com/questions/25463423/res-sendfile-absolute-path
19
20 module.exports.runScripts = function() {
21   cron.schedule('* * * * *', () => {
22     //call a function to run python scripts in a local directory every 3 minutes;
23     console.log("runing traffic manage");
24     const pythonProcess = spawn('python3.5', ['nodecap9.py']);
25     console.log("spawned process");
26     pythonProcess.stdout.on('data', (data) => {
27       // Do something with the data returned from python script
28       console.log("printing data");
29       console.log(data.toString());
30       //let sentPackets = fs.readFileSync('./newlog.json', 'utf8');
31       //console.log(sentPackets);
32     });
33
34     pythonProcess.stderr.on('data', (data) => {
35       console.log(data.toString());
36     });
37   });
38 }
39 module.exports.runScriptsTraffic = function() {
40   cron.schedule('* * * * *', () => {
41     //call a function to run python scripts in a local directory every 3 minutes;
42     console.log("runing traffic control");
43     const pythonProcess = spawn('python3.5', ['pytc5.py']);
44     pythonProcess.stdout.on('data', (data) => {
45       // Do something with the data returned from python script
46       console.log(data.toString());
47       //let sentPackets = fs.readFileSync('./internthing1.json', 'utf8');
48       //console.log(sentPackets);
49     });
50
51     pythonProcess.stderr.on('data', (data) => {
52       console.log(data.toString());
53     });
54   });
55 }
56 module.exports.runScriptsArp = function() {
57   cron.schedule('* * * * *', () => {
58     //call a function to run python scripts in a local directory every 3 minutes;
59     console.log("runing arp script control");
60     const pythonProcess = spawn('python3.5', ['getarp5.py']);
61     pythonProcess.stdout.on('data', (data) => {
62       // Do something with the data returned from python script
63       console.log(data.toString());
64       //let sentPackets = fs.readFileSync('./internthing1.json', 'utf8');
65       //console.log(sentPackets);
66     });
67

```

```

68     pythonProcess.stderr.on('data', (data) => {
69         console.log(data.toString());
70     });
71 });
72 }
73 module.exports.createGraph = function() {
74     cron.schedule('* * * * *', () => {
75         //call a function to run python scripts in a local directory every 3 minutes;
76         console.log("runing create graph");
77         const pythonProcess = spawn('python3.5', ['trafficplot7.py']);
78         pythonProcess.stdout.on('data', (data) => {
79             // Do something with the data returned from python script
80             console.log(data.toString());
81             //let sentPackets = fs.readFileSync('./internthing1.json', 'utf8');
82             //console.log(sentPackets);
83         });
84
85         pythonProcess.stderr.on('data', (data) => {
86             console.log(data.toString());
87         });
88     });
89 }
90 module.exports.createProtocolGraph = function() {
91     cron.schedule('* * * * *', () => {
92         //call a function to run python scripts in a local directory every 3 minutes;
93         console.log("runing create graph");
94         const pythonProcess = spawn('python3.5', ['protocolplot.py']);
95         pythonProcess.stdout.on('data', (data) => {
96             // Do something with the data returned from python script
97             console.log(data.toString());
98             //let sentPackets = fs.readFileSync('./internthing1.json', 'utf8');
99             //console.log(sentPackets);
100         });
101
102         pythonProcess.stderr.on('data', (data) => {
103             console.log(data.toString());
104         });
105     });
106 }
107 module.exports.createArpGraph = function() {
108
109     cron.schedule('* * * * *', () => {
110         //call a function to run python scripts in a local directory every 3 minutes;
111         console.log("runing create arp graph");
112         const pythonProcess = spawn('python3.5', ['generatefullarp.py']);
113         pythonProcess.stdout.on('data', (data) => {
114             // Do something with the data returned from python script
115             console.log(data.toString());
116             //let sentPackets = fs.readFileSync('./internthing1.json', 'utf8');
117             //console.log(sentPackets);
118         });
119
120         pythonProcess.stderr.on('data', (data) => {
121             console.log(data.toString());
122         });
123     });
124     cron.schedule('* * * * *', () => {
125         //call a function to run python scripts in a local directory every 3 minutes;
126         console.log("runing create arp graph");

```

```

127     const pythonProcess = spawn('python3.5', ['comparearpgraphs3.py']);
128     pythonProcess.stdout.on('data', (data) => {
129         // Do something with the data returned from python script
130         console.log(data.toString());
131         //let sentPackets = fs.readFileSync('./internthing1.json', 'utf8');
132         //console.log(sentPackets);
133     });
134
135     pythonProcess.stderr.on('data', (data) => {
136         console.log(data.toString());
137     });
138 });
139 }
140 module.exports.autoShape = function() {
141     cron.schedule('* * * * *', () => {
142         //call a function to run python scripts in a local directory every 3 minutes;
143         console.log("runing auto shape");
144         const pythonProcess = spawn('python3.5', ['autoshape9.py']);
145         pythonProcess.stdout.on('data', (data) => {
146             // Do something with the data returned from python script
147             console.log(data.toString());
148             //let sentPackets = fs.readFileSync('./internthing1.json', 'utf8');
149             //console.log(sentPackets);
150         });
151
152         pythonProcess.stderr.on('data', (data) => {
153             console.log(data.toString());
154         });
155     });
156 }
157 module.exports.genArp = function() {
158     cron.schedule('* * * * *', () => {
159         //call a function to run python scripts in a local directory every 3 minutes;
160         console.log("runing auto shape");
161         const pythonProcess = spawn('python3.5', ['generatefullarp.py']);
162         pythonProcess.stdout.on('data', (data) => {
163             // Do something with the data returned from python script
164             console.log(data.toString());
165             //let sentPackets = fs.readFileSync('./internthing1.json', 'utf8');
166             //console.log(sentPackets);
167         });
168
169         pythonProcess.stderr.on('data', (data) => {
170             console.log(data.toString());
171         });
172     });
173 }
174
175 module.exports.getFiles = function() {
176     cron.schedule('* * * * *', () => {
177         console.log("running get file");
178         //call functision to make http requests to the other nodes in the network
179         //for each ip in ips, make a request
180         var filePath=path.join(__dirname,"../../iplist.txt")
181
182         ipList=fs.readFileSync(filePath)
183         console.log(ipList.toString().split(','))
184         ipList=ipList.toString().split(',')
185         console.log(ipList)

```

```

186   for(var ipl in ipList){
187       console.log(chosenAddress)
188       if(ipList[ipl] == chosenAddress || ipList[ipl].length<=0)continue;
189       console.log('requesting ip '+ipList[ipl]);
190       (function(ipname){
191           console.log('requesting ip name'+ipname);
192           request('http://'+ipList[ipl]+':3030/getFile', function (error, response, body) {
193               console.log('attempting to write to file'+ipname);
194               console.log('error is '+error);
195               console.log('response is '+response);
196               console.log('body is from '+ipname.toString()+ ' '+body);
197               if(error == null && response != null){
198                   fs.writeFile('./iplog'+ipname+'.json', body, function(err) {
199                       if (err) {
200                           console.log("E!");
201                       } else {
202                           //fs.writeFile('./iplog172.27.0.90.json', body, function(err) {
203                               console.log("file saved "+ipname.toString());
204                               console.log(ipname)
205                           }
206                       })
207                   }else{
208                       console.log("nothing recieved");
209                   }
210               });
211           })(ipList[ipl]);
212       }
213   })
214
215 }
216 module.exports.getFilesArp = function() {
217     cron.schedule('* * * * *', () => {
218         console.log("running get file arp ");
219         var filePath=path.join(__dirname,"../..//iplist.txt")
220
221         ipList=fs.readFileSync(filePath)
222         console.log(ipList.toString().split(', '))
223         ipList=ipList.toString().split(', ')
224         console.log(ipList)
225         for(var ipl in ipList){
226             console.log(chosenAddress)
227             if(ipList[ipl] == chosenAddress || ipList[ipl].length<=0)continue;
228             console.log('requesting ip '+ipList[ipl]);
229             (function(ipname){
230                 console.log('requesting ip name'+ipname);
231                 request('http://'+ipList[ipl]+':3030/getArp', function (error, response, body) {
232                     console.log('attempting to write to file'+ipname);
233                     console.log('error is '+error);
234                     console.log('response is '+response);
235                     console.log('body is from '+ipname.toString()+ ' '+body);
236                     if(error == null && response != null){
237                         fs.writeFile('./arpfile'+ipname+'.json', body, function(err) {
238                             if (err) {
239                                 console.log("E!");
240                             } else {
241                                 //fs.writeFile('./iplog172.27.0.90.json', body, function(err) {
242                                     console.log("file saved "+ipname.toString());
243                                     console.log(ipname)
244                                 }
245                             })
246                         }
247                     }else{
248                         console.log("nothing recieved");
249                     }
250                 });
251             })(ipList[ipl]);
252         }
253     });
254 }

```



```

245     })
246     }else{
247         console.log("nothing recieved");
248     }
249     });
250 })(ipList[ip1]);
251 }
252
253 /*
254 //call functision to make http requests to the other nodes in the network
255 //for each ip in ips, make a request
256 var filePath=path.join(__dirname,".././iplist.txt")
257
258 ipList=fs.readFileSync(filePath)
259 console.log(ipList.toString().split(','))
260 ipList=ipList.toString().split(',')
261 console.log(ipList)
262 for(var ip1 in ipList){
263     console.log(chosenAddress)
264     if(ipList[ip1] == chosenAddress || ipList[ip1].length<=0)continue;
265     console.log('requesting ip '+ipList[ip1]);
266     request('http://'+ipList[ip1]+':3030/getArp', function (error, response, body) {
267         console.log('attempting to write to file');
268         console.log('error is '+error);
269         console.log('arp response is '+response);
270         console.log('body is from '+ipList[ip1].toString()+ ' '+body);
271         if(error == null && response != null){
272             fs.writeFile('./arpfile'+ipList[ip1]+' .json', body, function(err) {
273                 if (err) {
274                     console.log("E!");
275                 } else {
276                     //fs.writeFile('./iplog172.27.0.90.json', body, function(err) {
277                         console.log("file saved "+ipList[ip1].toString());
278                         console.log(ipList[ip1])
279                     }
280                 })
281             }else{
282                 console.log("nothing recieved");
283             }
284         });
285     }*/
286 }
287
288 }
289 module.exports.getTcDirections = function() {
290     cron.schedule('* * * * *', () => {
291
292         console.log("running get tc directions");
293         //call function to make http requests to the other nodes in the network
294         //for each ip in ips, make a request
295
296         for(var mIp in mainServerList){
297             console.log('http://'+mainServerList[mIp]+' :3030/getTcDirections');
298             request('http://'+mainServerList[mIp]+' :3030/getTcDirections'/*).on('response'*/, function (
299                 error, response, body) {
300                 console.log('attempting to write to file');
301                 console.log('attempting to write to file');
302                 console.log('error is '+error);
303                 console.log('response is '+response);

```

```

303     console.log('body is '+body);
304     if(error==null){
305         fs.writeFile('./tcdirs.json', body, function(err) {
306             if (err) {
307                 gotResponse=0
308                 console.log("E!");
309             } else {
310                 //fs.writeFile('./test1.json', body, function(err) {
311
312                     console.log("file saved in orig list");
313                 }
314             })
315         }else{
316             gotResponse=0
317             console.log("error with request")
318         }
319     });
320 }
321 if(gotResponse==0){
322     console.log("defaulting to old list")
323     for(var mIp in ipList){
324         console.log('http://'+ipList[mIp]+':3030/getTcDirections');
325         request('http://'+ipList[mIp]+':3030/getTcDirections', function (error, response, body) {
326             console.log('attempting to write to file');
327             console.log('error is '+error);
328             console.log('response is '+response);
329             console.log('body is '+body);
330             if(error==null){
331                 fs.writeFile('./test1.json', body, function(err) {
332                     if (err) {
333                         console.log("E!");
334                     } else {
335                         //fs.writeFile('./test1.json', body, function(err) {
336                             gotResponse=1
337                             console.log("file saved");
338                         }
339                     })
340                 }else{
341                     console.log("error with request")
342                 }
343             });
344         }
345
346
347         gotResponse=1;
348     }
349 })
350 }
351 module.exports.getNetworkInterfaces = function() {
352     console.log("runing network interfaces list");
353     //const pythonProcess = spawn('python', ['protocolplot.py']);
354     var networkInterfaces = os.networkInterfaces( );
355
356
357     console.log( networkInterfaces );
358     //for(let ni of networkInterfaces){
359         for (var key in networkInterfaces) {
360             //if (networkInterfaces.hasOwnProperty(key)) { // this will check if key is owned by data
361                 object and not by any of it's ancestors

```

```

361 console.log(key+' : '+networkInterfaces[key]); // this will show each key with it's value
362 for(k in networkInterfaces[key]){
363     console.log(k+' : '+networkInterfaces[key][k]);
364     /*for(l in networkInterfaces[key][k]){
365         console.log(l+" : "+networkInterfaces[key][k][l])
366     }*/
367     if("address" in networkInterfaces[key][k]){
368
369         console.log("address is "+networkInterfaces[key][k]["address"])
370         var newaddr= networkInterfaces[key][k]["address"]
371         if (newaddr.includes("172.27.0") && k==0){
372             chosenAddress=newaddr;
373             console.log("is the mac "+networkInterfaces[key][k]["mac"])
374             console.log("is the internal "+networkInterfaces[key][k]["internal"])
375             console.log("is the cidr "+networkInterfaces[key][k]["cidr"])
376             console.log("is the family "+networkInterfaces[key][k]["family"])
377             console.log("on address "+key+" is k"+k);
378         }
379     }
380     /*for(let el of networkInterfaces[key][k]){
381         console.log(el)
382     }*/
383
384 }
385 //}
386 }
387 console.log("chosen address is "+chosenAddress)
388 cron.schedule('* * * * *', () => {
389     //call a function to run python scripts in a local directory every 3 minutes;
390     console.log("runing network interfaces list");
391     //const pythonProcess = spawn('python', ['protocolplot.py']);
392     var networkInterfaces = os.networkInterfaces( );
393
394
395     console.log( networkInterfaces );
396     //for(let ni of networkInterfaces){
397         for (var key in networkInterfaces) {
398             //if (networkInterfaces.hasOwnProperty(key)) { // this will check if key is owned by data
399                 //object and not by any of it's ancestors
400                 console.log(key+' : '+networkInterfaces[key]); // this will show each key with it's value
401                 for(k in networkInterfaces[key]){
402                     console.log(k+' : '+networkInterfaces[key][k]);
403                     /*for(l in networkInterfaces[key][k]){
404                         console.log(l+" : "+networkInterfaces[key][k][l])
405                     }*/
406                     if("address" in networkInterfaces[key][k]){
407
408                         console.log("address is "+networkInterfaces[key][k]["address"])
409                         var newaddr= networkInterfaces[key][k]["address"]
410                         if (newaddr.includes("172.27.0")){
411                             chosenAddress=newaddr;
412                         }
413                     }
414                     /*for(let el of networkInterfaces[key][k]){
415                         console.log(el)
416                     }*/
417                 }
418             //}

```

```

419     }
420     console.log("chosen address is "+chosenAddress)
421     //}
422
423 });
424 }

```

B.3 nodecap9.py

```

1 import traceback
2 import json
3 import pyshark
4 from collections import Counter
5 import os
6 import datetime
7 #cap = pyshark.LiveCapture(interface='wlan1')
8 #cap.sniff(packet_count=50)
9 from sklearn import tree
10 import netifaces as ni
11 from joblib import dump, load
12 import math
13 bridgeflag=0
14
15 def chooseInterface():
16     import netifaces as ni
17     chosenInterface='wlp3s0'
18     for interface in ni.interfaces():
19         print( str(interface))
20         try:
21             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
22             if '172.27.0.' in ip:
23                 chosenInterface=interface
24             if 'br' in interface: bridgeflag=1
25         except:
26             print( "error reading interface")
27         print( "chosen interface is "+str(chosenInterface))
28     return chosenInterface
29 interface=chooseInterface()
30 ni.ifaddresses(interface)
31 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
32 clf = tree.DecisionTreeClassifier()
33 try:
34     clf = load('tree'+ip+'.joblib')
35 except:
36     #continue
37     print("error reading data")
38
39 print(ip) # should print "192.168.100.37"
40 packetlog=dict();
41 userlog=dict();
42 storedlog=dict();
43 templog=dict();
44
45
46 import glob
47 for filename in glob.glob('iplog*'):
48     print(filename)
49     #if ip in filename: continue;

```

```

50     #if ip in filename: continue;
51
52
53     with open(filename,'r')as f:
54         try:
55             templog= json.load(f)
56         except:
57             continue
58     #print((str(templog)))
59     #storedlog={x: storedlog.get(x,0)+templog.get(x,0) for x in set(storedlog).union(templog)}
60     #if templog[]
61     for key,val in list(templog.items()):
62
63
64         if key in storedlog:
65             print("key is in stored log "+str(key))
66             if '172.27.0' not in key: continue
67             if not isinstance(storedlog[key],dict): storedlog[key]=dict()
68             if 'remoteadded' not in storedlog[key]: stored[key]['remoteadded']=list()
69             if isinstance(storedlog[key],dict) and 'remoteadded' in storedlog[key]:
70                 if isinstance(storedlog[key]['remoteadded'],list):
71                     #if filename[(filename.find('g')+1):(filename.find('.j'))] not in
72                         storedlog[key]['remoteadded']: storedlog[key]['remoteadded'].append
73                             (filename[(filename.find('g')+1):(filename.find('.j'))])
74
75                     if 'originip' in templog and templog['originip'] not in storedlog[key
76                         ]['remoteadded']: storedlog[key]['remoteadded'].append(templog['
77                             originip'])
78
79
80
81
82             if 'remoteadded' in val:
83                 for ij in val['remoteadded']:
84                     if ij not in storedlog[key]['remoteadded']:storedlog[key]['remoteadded
85                         '].append(ij)
86
87
88
89
90             if 'route' not in storedlog[key]:
91                 storedlog[key]['route']=[]
92             if 'route' in storedlog[key] and 'route' in val:
93
94                 #storedlog[key]['route']=val['route']+storedlog[key]['route']
95                 for il in val['route']:
96                     if il not in storedlog[key]['route']: storedlog[key]['route'].append(il
97                         )
98
99                 #if filename[(filename.find('g')+1):(filename.find('.j'))] not in storedlog
100                     [key]['route']: storedlog[key]['route'].append(filename[(filename.find
101                         ('g')+1):(filename.find('.j'))]) #remove later
102             if 'route' in storedlog[key] and 'route' not in val:
103                 print("adding file name")
104
105                 #if filename[(filename.find('g')+1):(filename.find('.j'))] not in storedlog
106                     [key]['route']: storedlog[key]['route'].append(filename[(filename.find
107                         ('g')+1):(filename.find('.j'))])
108
109
110             #print("debug route "+str(storedlog[key]['route']))
111             #storedlog[key]['remoteadded']=filename[5:17]
112             #try:
113             #print "doing addition "+str(storedlog[key])+ " and "+str(val)

```

```

99         ...
100         storedlog[key]['count']= storedlog[key]['count']+val['count']
101         for key1,val1 in list(val.items()):
102             if not isinstance(val1,dict) or 'count' not in val1:
103                 storedlog[key][key1]=val1
104                 continue
105                 print("key 1 is "+str(key1)+" val 1 is"+str(val1))
106                 if key1=='count': continue
107                 if key1 in storedlog[key] and 'count' in storedlog[key][key1]:
108                     storedlog[key][key1]=val1
109                     if storedlog[key][key1]['count']!=val1['count']:
110                         storedlog[key][key1]['count']= storedlog[key][key1]['count']+val1['
111                             count']
112                         else:
113                             storedlog[key][key1]['count']= storedlog[key][key1]['count']
114                 elif key1 in storedlog[key]:
115                     storedlog[key][key1]=val1
116         ...
117
118
119         #print "result is "+str(storedlog[key])
120     #except:
121         #print "error doing addition"
122         #exit()
123
124
125     else:
126         print("adding new key to stored log")
127         #val['remoteadded']=filename[5:17]
128         if isinstance(val,dict) and 'remoteadded' in val:
129             if isinstance(val['remoteadded'],list):
130                 val['remoteadded'].append(templog['originip'])
131             else:
132
133                 val['remoteadded']=[templog['originip']]
134         elif isinstance(val,dict):
135             if 'originip' in templog: val['remoteadded']=[templog['originip']]
136             else:val['remoteadded']=[]# fix later
137
138         print("adding value "+str(val))
139         storedlog[key]=val
140         #os.system('mv '+filename+' '+oldlog'+filename[4:])
141
142         print("printing new log")
143         print((json.dumps(storedlog)))
144
145     userlog=dict(storedlog)
146     #if ip in storedlog: userlog=storedlog[ip]
147     storedlogbackup=dict(storedlog)
148     #with open("datathing1.json",'r')as f:
149     # userlog= json.load(f)
150
151     externpacketlog=dict()
152     #with open("externthing1.json",'r')as f:
153     # externpacketlog= json.load(f)
154     #print json.dumps(userlog)
155     internpacketlog=dict()
156     #with open("internthing1.json",'r')as f:

```

```

157 # internpacketlog= json.load(f)
158 #userlog=dict();
159 print((json.dumps(storedlog)));
160 #print json.dumps(userlog);
161 #print json.dumps(internpacketlog)
162 #print json.dumps(externpacketlog)
163 print("sniffing packets")
164 errorcount =0;
165 cap = pyshark.LiveCapture(interface=interface)
166 print("sniffed packets")
167 cap.sniff(packet_count=500)
168 for packet in cap:
169     ipstr='ip'
170     if ipstr not in packet: ipstr='ipv6'
171     #print packet
172     #print packet.highest_layer
173     try:
174         if ('tcp' in packet or 'udp' in packet) and ('172.27.0' in str(packet[ipstr].src)):
175             print("key is "+str(packet[ipstr].src))
176             #print(packet)
177             #print packet.highest_layer
178             #print packet.tcp.dstport
179             print (packet[ipstr].src)
180
181     try:
182         if packet[ipstr].src in userlog:
183             print("adding to dict again ")
184             #packetlog[packet.tcp.dstport] = 1+packetlog[packet.tcp.dstport];
185             print(("added to user log "+str(packet[ipstr].src)+" "+str(userlog[packet[
186                 ipstr].src])))
187             userlog[packet[ipstr].src]['count'] = userlog[packet[ipstr].src]['count']+1;
188             #userlog[packet[ipstr].src]['length'] = int(packet.length)
189             userlog[packet[ipstr].src]['totalsize'] = int(packet.length)+int(userlog[
190                 packet[ipstr].src]['totalsize'])
191             userlog[packet[ipstr].src]['length'] = int(userlog[packet[ipstr].src]['
192                 totalsize']/userlog[packet[ipstr].src]['count'])
193             #userlog[packet[ipstr].src]['time'].append(str((packet.sniff_timestamp)))
194             if 'route' not in userlog[packet[ipstr].src] and packet[ipstr].src != ip and
195                 packet[ipstr].dst != ip:
196                 print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
197                     ipstr].dst))
198                 print("adding route")
199                 userlog[packet[ipstr].src]['route']=[ip]
200             elif packet[ipstr].src != ip and packet[ipstr].dst != ip and ip not in userlog
201                 [packet[ipstr].src]['route']:
202                 print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
203                     ipstr].dst))
204                 print("adding route")
205                 userlog[packet[ipstr].src]['route'].append(ip)
206             if packet.highest_layer not in userlog[packet[ipstr].src]['protocol']:
207                 print("adding new protocol")
208                 userlog[packet[ipstr].src]['protocol'][packet.highest_layer]=1
209             else:
210                 userlog[packet[ipstr].src]['protocol'][packet.highest_layer]=userlog[packet
211                     [ipstr].src]['protocol'][packet.highest_layer]+1
212
213         if packet[ipstr].dst in userlog[packet[ipstr].src]:
214             #userlog[packet[ipstr].src]+1;

```

```

208         print(("incrementing user log before "+str(userlog[packet[ipstr].src][
                packet[ipstr].dst])))
209         userlog[packet[ipstr].src][packet[ipstr].dst]['count']= userlog[packet[
                ipstr].src][packet[ipstr].dst]['count']+1;
210         print(("incrementing user log after "+str(userlog[packet[ipstr].src][
                packet[ipstr].dst])))
211         if 'time' in userlog[packet[ipstr].src][packet[ipstr].dst]: userlog[
                packet[ipstr].src][packet[ipstr].dst]['time'].append(str(packet.
                sniff_timestamp))
212         else: userlog[packet[ipstr].src][packet[ipstr].dst]['time']= [str(
                packet.sniff_timestamp)];
213     else:
214         userlog[packet[ipstr].src][packet[ipstr].dst]= dict();
215         userlog[packet[ipstr].src][packet[ipstr].dst]['count']= 1;
216         userlog[packet[ipstr].src][packet[ipstr].dst]['time']= [str(packet.sniff_
                timestamp)];
217
218     #if packet[ipstr].src== "172.27.0.90":
219     #if packet[ipstr].src == "172.27.0.90" or packet[ipstr].src== "172.27.0.155"
        or packet[ipstr].src == "172.27.0.15" or packet[ipstr].src
        == "172.27.0.227":
220         #internpacketlog[packet[ipstr].src] = internpacketlog[packet[ipstr].src]+1;
221     #else:
222         #externpacketlog[packet[ipstr].dst] = externpacketlog[packet[ipstr].dst]+1;
223
224     else:
225         #print "adding to dict"
226         #packetlog[packet.tcp.dstport]=1;
227
228         #userlog[packet[ipstr].src]=1;
229         userlog[packet[ipstr].src]=dict()
230         userlog[packet[ipstr].src]['count']=1
231         userlog[packet[ipstr].src]['totalsize']=packet.length
232         userlog[packet[ipstr].src]['protocol']=dict()
233         userlog[packet[ipstr].src]['protocol'][packet.highest_layer]=1
234         userlog[packet[ipstr].src][packet[ipstr].dst]= dict();
235         userlog[packet[ipstr].src][packet[ipstr].dst]['count']= 1;
236         if 'route' not in userlog[packet[ipstr].src] and packet[ipstr].src != ip and
                packet[ipstr].dst != ip:
237             print("adding route")
238             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
                ipstr].dst))
239             userlog[packet[ipstr].src]['route]=[ip]
240         elif packet[ipstr].src != ip and packet[ipstr].dst != ip and ip not in userlog
                [packet[ipstr].src]['route']:
241             print("adding route")
242             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
                ipstr].dst))
243             userlog[packet[ipstr].src]['route'].append(ip)
244
245         '''
246         elif packet[ipstr].src != ip:
247             print("appendig route")
248             userlog[packet[ipstr].src]['route'].append(ip)
249         '''
250     #else:
251         #userlog[packet[ipstr].src]['route'].append(ip)
252
253

```



```

254         userlog[packet[ipstr].src]['port']=dict()
255         #userlog[packet[ipstr].src]['time']=[str(packet.sniff_timestamp)]
256         if 'dstport' in packet:
257             userlog[packet[ipstr].src]['port'][packet[ipstr].dstport]=1
258
259
260     except:
261         errorcount=errorcount+1;
262         print("error with fields")
263         print((traceback.format_exc()))
264         #print packet
265 if ('tcp' in packet or 'udp' in packet) and ('172.27.0' in str(packet[ipstr].dst)):
266     print("key is "+str(packet[ipstr].dst))
267     #print(packet)
268     #print packet.highest_layer
269     #print packet.tcp.dstport
270     print (packet[ipstr].dst)
271
272     try:
273         if packet[ipstr].dst in userlog:
274             print("adding to dict again ")
275             #packetlog[packet.tcp.dstport] = 1+packetlog[packet.tcp.dstport];
276             print(("added to user log "+str(packet[ipstr].dst)+" "+str(userlog[packet[
277                 ipstr].dst])))
278             userlog[packet[ipstr].dst]['count'] = userlog[packet[ipstr].dst]['count']+1;
279             #userlog[packet[ipstr].dst]['length'] = int(packet.length)
280             userlog[packet[ipstr].dst]['totalsize'] = int(packet.length)+int(userlog[
281                 packet[ipstr].dst]['totalsize'])
282             userlog[packet[ipstr].dst]['length'] = int(userlog[packet[ipstr].dst]['
283                 totalsize']/userlog[packet[ipstr].dst]['count'])
284             if 'route' not in userlog[packet[ipstr].dst] and packet[ipstr].dst != ip and
285                 packet[ipstr].src != ip:
286                 print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
287                     ipstr].dst))
288                 userlog[packet[ipstr].dst]['route']=[ip]
289             elif 'route' in userlog[packet[ipstr].dst] and packet[ipstr].dst != ip and ip
290                 not in userlog[packet[ipstr].dst]['route'] and packet[ipstr].src != ip:
291                 print("appending route ")
292                 print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
293                     ipstr].dst))
294                 userlog[packet[ipstr].dst]['route'].append(ip)
295             if packet.highest_layer not in userlog[packet[ipstr].dst]['protocol']:
296                 print("adding new protocol")
297                 userlog[packet[ipstr].dst]['protocol'][packet.highest_layer]=1
298             else:
299                 userlog[packet[ipstr].dst]['protocol'][packet.highest_layer]=userlog[packet
300                     [ipstr].dst]['protocol'][packet.highest_layer]+1
301
302         if packet[ipstr].src in userlog[packet[ipstr].dst]:
303             #userlog[packet[ipstr].dst]+1;
304             print(("incrementing user log before "+str(userlog[packet[ipstr].dst][
305                 packet[ipstr].src])))
306             userlog[packet[ipstr].dst][packet[ipstr].src]['count']= userlog[packet[
307                 ipstr].dst][packet[ipstr].src]['count']+1;
308             print(("incrementing user log after "+str(userlog[packet[ipstr].dst][
309                 packet[ipstr].src])))
310             if 'time' in userlog[packet[ipstr].dst][packet[ipstr].src]['time']:
311                 userlog[packet[ipstr].dst][packet[ipstr].src]['time'].append(str(

```

```

        packet.sniff_timestamp));
301     else: userlog[packet[ipstr].dst][packet[ipstr].src]['time']= [str(
        packet.sniff_timestamp)];
302     else:
303         userlog[packet[ipstr].dst][packet[ipstr].src]= dict();
304         userlog[packet[ipstr].dst][packet[ipstr].src]['count']= 1;
305         userlog[packet[ipstr].dst][packet[ipstr].src]['time']= [str(packet.sniff_
            timestamp)];
306
307     #if packet[ipstr].src== "172.27.0.90":
308     #if packet[ipstr].src == "172.27.0.90" or packet[ipstr].src== "172.27.0.155"
        or packet[ipstr].src == "172.27.0.15" or packet[ipstr].src
        == "172.27.0.227":
309         #internpacketlog[packet[ipstr].src] = internpacketlog[packet[ipstr].src]+1;
310     #else:
311         #externpacketlog[packet[ipstr].dst] = externpacketlog[packet[ipstr].dst]+1;
312
313     else:
314         #print "adding to dict"
315         #packetlog[packet.tcp.dstport]=1;
316
317         #userlog[packet[ipstr].src]=1;
318         userlog[packet[ipstr].dst]=dict()
319         userlog[packet[ipstr].dst]['count']=1
320         userlog[packet[ipstr].dst]['totalsize']=packet.length
321         userlog[packet[ipstr].dst]['protocol']=dict()
322         userlog[packet[ipstr].dst]['protocol'][packet.highest_layer]=1
323         userlog[packet[ipstr].dst][packet[ipstr].src]= dict();
324         userlog[packet[ipstr].dst][packet[ipstr].src]['count']= 1;
325         userlog[packet[ipstr].src]['port']=dict()
326         if 'route' not in userlog[packet[ipstr].dst] and packet[ipstr].dst != ip and
            packet[ipstr].src != ip:
327             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
                ipstr].dst))
328             userlog[packet[ipstr].dst]['route']=[ip]
329         elif packet[ipstr].dst != ip and packet[ipstr].src != ip and ip not in userlog
            [packet[ipstr].dst]['route']:
330             print("appending route " +str(ip)+" ")
331             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
                ipstr].dst))
332             userlog[packet[ipstr].dst]['route'].append(ip)
333
334         #userlog[packet[ipstr].src]['time']=[str(packet.sniff_timestamp)]
335         if 'srcport' in packet:
336             userlog[packet[ipstr].src]['port'][packet[ipstr].srcport]=1
337
338
339     except:
340         errorcount=errorcount+1;
341         print("error with fields")
342         print((traceback.format_exc()))
343         #print packet
344
345     except:
346         print("error ")
347         print((traceback.format_exc()))
348         #print packet
349
350 #print json.dumps(packetlog)

```

```

351 #import netifaces as ni
352 #ni.ifaddresses('wlan1')
353 #ip = ni.ifaddresses('wlan1')[ni.AF_INET][0]['addr']
354 #print ip # should print "192.168.100.37
355 for key,val in list(userlog.items()):
356     if not isinstance(val,dict)or 'route' not in val or not isinstance(val['route'],list): continue
357     for el in val['route']:
358         print("checking route "+el)
359         routecount=0
360         for key1,val1 in userlog.items():
361             print(val1)
362             if isinstance(val1,dict) and 'route' in val1 and isinstance(val1['route'],list) and el in
                 val1['route'] and key1 != el:
363                 if len(val1['protocol'])>2 or val1['count']>20:
364                     routecount=routecount+1+len(val1['protocol'])
365             if el not in userlog:
366                 userlog[el]=dict()
367             print("adding route count to "+str(el)+" which is "+str(routecount) )
368             if 'routecount' not in userlog[el] or userlog[el]['routecount']<routecount: userlog[el]['
                 routecount']=routecount
369
370 arplog=dict()
371 try:
372     with open("fullarp"+ip+".json",'r')as f:
373         arplog= json.load(f)
374 except:
375     print("error reading file")
376 classifierinit=0
377 try:
378     clf = load('tree'+ip+'.joblib')
379     classifierinit=1
380 except:
381     print("error with classifier")
382 for key,val in list(userlog.items()):
383     if isinstance(val,dict) and 'route' in val:
384         for el in list(val['route']):
385             print("checking "+str(el))
386             treepred='node'
387             if el in userlog:
388                 if (el in userlog and 'class' in userlog[el] and userlog[el]['class'] != 'node') or
                    classifierinit ==0 : continue
389             try:
390                 '''
391                 pdata=[]
392                 val=userlog[el]
393                 if 'count' in val and 'length' in val: pdata.append(val['count']*int(math.log(val
                    ['length'])))
394                 else: pdata.append(0)
395                 if 'count' in val: pdata.append(val['count'])
396                 else: pdata.append(0)
397                 if 'length' in val: pdata.append(val['length'])
398                 else: pdata.append(0)
399                 #pdata.append(0)
400                 dests=0
401                 #if 'class' in val: pdata['class']=int(val['class'])
402                 #else: pdata.append(int('node'))
403                 if 'routecount' in val: pdata.append(val['routecount'])
404                 else: pdata.append(0)
405                 for key1,val1 in val.items():

```

```

406         print(key1)
407         if '172.27.0' in key1:
408             dests=dests+1
409     pdata.append(dests)
410     ',,'
411     pdata=[]
412     if 'length' in val:
413         pdata.append((val['count']*int(math.log(val['length']))))
414         pdata.append(val['count'])
415         pdata.append(val['length'])
416         #pdata.append(0)
417         dests=0
418         #if 'class' in val: pdata['class']=int(val['class'])
419         #else: pdata.append(int('node'))
420         if 'routecount' in val: pdata.append(val['routecount'])
421         else: pdata.append(0)
422         for key1,val1 in val.items():
423             #print(key1)
424             if '172.27.0' in key1:
425                 dests=dests+1
426         pdata.append(dests)
427         try:
428             arpdict=dict()
429             with open("arpfile"+ip+".json") as f:
430                 arpdict=json.load(f)
431             if key in arpdict:
432                 pdata.append(len(arpdict[key]))
433             else:
434                 pdata.append(0)
435         except:
436             pdata.append(0)
437
438     treepred=clf.predict([pdata])[0]
439     print("tree pred is"+treepred+" for "+str(el))
440     userlog[el]['class']=treepred
441     #break
442     continue
443 except:
444     if 'routecount' in userlog[el] and userlog[el]['routecount']>6: userlog[el]['class']
445         '= 'gateway'
446     else: userlog[el]['class']='node'
447
448 if el not in userlog and '172.27.0.' in el:
449     userlog[el]=dict()
450     userlog[el]['class']='bridge'
451 elif '172.27.0.' in el:
452     if 'routecount' in userlog[el] and userlog[el]['routecount']>6: userlog[el]['class']='
453         gateway'
454     else: userlog[el]['class']='node'
455
456 for key,val in userlog.items():
457     if isinstance(val,dict) and ('class' not in val or val['class']=='node'):
458         if 'routecount' in userlog[key] and userlog[key]['routecount']>=3: userlog[key]['
459             class']='gateway'
460         else: userlog[key]['class']='node'
461     for key1,val1 in val.items():
462         if '172.27.0' in key1 and key1 in userlog:
463             if 'routecount' in userlog[key1] and userlog[key1]['routecount']>=3: userlog[

```

```

462         key1['class']='gateway'
463     else: userlog[key1]['class']='node'
464     elif '172.27.0' in key1 and key1 not in userlog:userlog[key1]['class']='node'
465
466 savedLog= dict();
467 #storedlog=userlog;
468
469 print((json.dumps(storedlogbackup)))
470 print((json.dumps(storedlog)))
471 #print json.dumps(internpacketlog)
472 #print json.dumps(externpacketlog)
473 import random
474 dictid=random.randint(1,21)*5
475 userlog['id']=dictid
476 userlog['originip']=ip
477 userlog['bridge']=bridgeflag
478
479 packetfile='iplog'+ip+'.json';
480 with open(packetfile,'w') as outfile:
481     json.dump(userlog,outfile)
482 with open('datathing1.json','w') as outfile:
483     json.dump(userlog,outfile)
484
485
486 import os
487 print("new val is")
488 os.system('cat '+packetfile)
489
490 print("\n\n\nold val is "+str(storedlog)))
491 exit()
492 #os.system('cat internthing1.json')
493 #os.system('cat externthing1.json')

```

B.4 autoshape9.py

```

1 import pandas as pd
2 import traceback
3 import json
4 import math
5 import os
6
7 from sklearn import tree
8 from joblib import dump, load
9 import socket
10 def valid_ip(address):
11     try:
12         socket.inet_aton(address)
13         return True
14     except:
15         return False
16 ip='172.27.0.82'
17 def chooseInterface():
18     import netifaces as ni
19     chosenInterface='wlp3s0'
20     for interface in ni.interfaces():
21         print(( str(interface)))
22         try:

```

```

23             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
24             if '172.27.0.' in ip:
25                 chosenInterface=interface
26         except:
27             print( "error reading interface")
28             print(( "chosen interface is "+str(chosenInterface)))
29         return chosenInterface
30 import netifaces as ni
31 interface=chooseInterface()
32 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
33 data=dict()
34 with open('iplog'+ip+'.json','r')as f:
35     data= json.load(f)
36 prevlog=dict()
37 try:
38     with open("shapefile"+ip+".json",'r')as f:
39         prevlog= json.load(f)
40 except:
41     print("error reading json dictionary")
42 datalist=[]
43 classlist=[]
44 resultlist=[]
45 sublist=[]
46 manualforward=dict()
47 with open('tcdirs'+'.json','r')as f:
48     manualforward= json.load(f)
49 priolist=manualforward['prioritize']
50 slowlist=manualforward['slow']
51 for key,val in data.items():
52     if not isinstance(val,dict) or 'count' not in val: continue
53
54     pdata=dict()
55     print(str(key)+" "+str(val['count']))
56     pdata['ip']=key
57     if 'length' in val:
58         pdata['countlen']=val['count']*int(math.log(val['length']))
59         pdata['count']=val['count']
60         pdata['length']=val['length']
61         pdata['dests']=0
62         if 'class' in val: pdata['class']=val['class']
63         else: pdata['class']='node'
64         if 'routecount' in val: pdata['routecount']=val['routecount']
65         else: pdata['routecount']=0
66         for key1,val1 in val.items():
67             if not valid_ip(key1): continue
68             subdests=dict()
69             subdests['ip']=key1
70             subdests['count']=val1['count']
71             sublist.append(subdests)
72
73             print(key1)
74             if '172.27.0.' in key1:
75                 pdata['dests']=pdata['dests']+1
76
77     else:
78         pdata['count']=val['count']
79         pdata['length']=0
80         pdata['class']=val['class']
81         if 'routecount' in val: pdata['routecount']=val['routecount']

```

```

82     else: pdata['routecount']=0
83     try:
84         arpdict=dict()
85         with open("arpfile"+ip+".json") as f:
86             arpdict=json.load(f)
87             if key in arpdict:
88                 pdata['arp']=len(arpdict[key])
89             else:
90                 pdata['arp']=0
91     except:
92         #pdata.append(0)
93         pdata['arp']=0
94     #datalist.append(pdata)
95
96
97     if not isinstance(val,dict) or 'count' not in val or 'length' not in val: continue
98     classdata=[]
99     print(str(key)+" "+str(val['count']))
100    #pdata['ip']=key
101    classdata.append(val['count']*int(math.log(val['length'])))
102    classdata.append(val['count'])
103    classdata.append(val['length'])
104    #classdata.append(0)
105    dests=0
106    #if 'class' in val: classdata['class']=int(val['class'])
107    #else: classdata.append(int('node'))
108    if 'routecount' in val: classdata.append(val['routecount'])
109    else: classdata.append(0)
110    for key1,val1 in val.items():
111        print(key1)
112        if '172.27.0' in key1:
113            dests=dests+1
114    classdata.append(dests)
115    #print(clf.predict([classdata]))
116    #if key=='172.27.0.82': resultlist.append('node')
117    #elif key=='172.27.0.15': resultlist.append('gate')
118    #elif key=='172.27.0.90': resultlist.append('gate')
119    #elif key=='172.27.0.78': resultlist.append('bridge')
120
121    #elif key=='172.27.0.12': resultlist.append('bridge')
122    #else: resultlist.append('node')
123    try:
124        arpdict=dict()
125        with open("arpfile"+ip+".json") as f:
126            arpdict=json.load(f)
127            if key in arpdict:
128                classdata.append(len(arpdict[key]))
129            else:
130                classdata.append(0)
131    except:
132        classdata.append(0)
133
134
135        #datalist.append(classdata)
136    pdata['tree']=classdata
137
138    datalist.append(pdata)
139    #classlist.append(classdata)
140    if key in priolist:

```

```

141     classlist.append(classdata)
142     resultlist.append('prioritize')
143 elif key in slowlist:
144
145     classlist.append(classdata)
146     resultlist.append('slow')
147 else:
148     #print("doing nothing")
149     classlist.append(classdata)
150     resultlist.append('middle')
151     #if pdata['count']>500:
152     # resultlist.append('slow')
153     #elif pdata['count']<100:
154     # resultlist.append('prioritize')
155
156     #else: resultlist.append('middle')
157 print(str(datalist))
158 print(str(sublist))
159 manualforward=dict()
160 with open('tcdirs'+'.json','r')as f:
161     manualforward= json.load(f)
162 shapelist=dict()
163 print(str(data))
164 os.system("sudo tc qdisc del dev "+interface+" root")
165 os.system("tc qdisc add dev "+interface+" root handle 1: htb default 30")
166 os.system("tc class add dev "+interface+" parent 1: classid 1:1 htb rate 6mbit burst 15k")
167 os.system("tc class add dev "+interface+" parent 1:1 classid 1:10 htb rate 5mbit ceil 10mbit burst
168     15k")
169 os.system("tc class add dev "+interface+" parent 1:1 classid 1:20 htb rate 4mbit ceil 6mbit burst
170     15k")
171 os.system("tc class add dev "+interface+" parent 1:1 classid 1:30 htb rate 40kbit ceil 40kbit burst
172     10k")
173 for el in manualforward['prioritize']:
174     shapelist[el]=dict()
175     shapelist[el]['comm']=[]
176     faststring1="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst "+el
177     +" flowid 1:10"
178     faststring2="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src "+el
179     +" flowid 1:10"
180     print(faststring1)
181     os.system(faststring1)
182     print(faststring2)
183     os.system(faststring2)
184     shapelist[el]['class']='prio'
185     shapelist[el]['comm'].append(faststring1)
186     shapelist[el]['comm'].append(faststring2)
187
188 for el in manualforward['slow']:
189     shapelist[el]=dict()
190     shapelist[el]['comm']=[]
191     slowstring1="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst "+el
192     +" flowid 1:30"
193     slowstring2="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src "+el
194     +" flowid 1:30"
195     print(slowstring1)
196     os.system(slowstring1)
197     print(slowstring2)
198     os.system(slowstring2)
199     shapelist[el]['class']='slow'

```



```

193     shapelist[el]['comm'].append(slowstring1)
194     shapelist[el]['comm'].append(slowstring2)
195     clf = tree.DecisionTreeClassifier()
196     print("data list is "+str(datalist))
197     print("result list is "+str(resultlist))
198     clf = clf.fit(classlist, resultlist)
199     dump(clf, 'shapetree'+ip+'.joblib')
200     clf = load('shapetree'+ip+'.joblib')
201     print(str(shapelist))
202     for el in datalist:
203         print(el)
204         if el['ip'] in shapelist: continue
205         try:
206             print(str(el['tree']))
207             print("prediction for "+str(el['ip'])+" "+str(clf.predict([el['tree']])[0]))
208             prediction=clf.predict([el['tree']])[0]
209             if prediction == 'prioritize':
210                 shapelist[el['ip']]=dict()
211                 shapelist[el['ip']]['comm']=[]
212                 faststring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst
                "+el['ip']+" flowid 1:10"
213                 print(faststring)
214                 shapelist[el['ip']]['class']='prio'
215                 shapelist[el['ip']]['comm'].append(faststring)
216
217                 os.system(faststring)
218                 faststring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src
                "+el['ip']+" flowid 1:10"
219                 print(faststring)
220                 os.system(faststring)
221                 shapelist[el['ip']]['comm'].append(faststring)
222             elif prediction=='slow':
223                 shapelist[el['ip']]=dict()
224                 shapelist[el['ip']]['comm']=[]
225                 shapelist[el['ip']]['class']='slow'
226                 slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst
                "+el['ip']+" flowid 1:30"
227                 print(slowstring)
228                 os.system(slowstring)
229                 shapelist[el['ip']]['comm'].append(slowstring)
230                 slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src
                "+el['ip']+" flowid 1:30"
231                 print(slowstring)
232                 os.system(slowstring)
233                 shapelist[el['ip']]['comm'].append(slowstring)
234             else:
235                 shapelist[el['ip']]=dict()
236                 shapelist[el['ip']]['comm']=[]
237                 '''
238                 shapelist[el['ip']]=dict()
239                 shapelist[el['ip']]['comm']=[]
240                 midstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst
                "+el['ip']+" flowid 1:20"
241                 print(midstring)
242                 os.system(midstring)
243                 shapelist[el['ip']].append(midstring)
244                 midstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src
                "+el['ip']+" flowid 1:20"
245                 print(midstring)

```

```

246     os.system(midstring)
247     shapelist[el['ip']].append(midstring)
248     '''
249
250     if (el['count']> el['length'] or el['dests']>4 or el['count']> 300) and ('class' not in
        el or el['class'] == 'node') and (el['ip'] not in prevlog or el['ip'] in prevlog and
        prevlog[el['ip']]['class'] != 'slow'):
251
252         slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip
            dst "+el['ip']+" flowid 1:30"
253         print(slowstring)
254         os.system(slowstring)
255         shapelist[el['ip']]['class']='slow'
256         shapelist[el['ip']]['comm'].append(slowstring)
257         slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip
            src "+el['ip']+" flowid 1:30"
258         print(slowstring)
259         os.system(slowstring)
260         shapelist[el['ip']]['comm'].append(slowstring)
261     elif el['class']=='gateway' or el['class']=='bridge':
262         shapelist[el['ip']]['class']='prio'
263
264         faststring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip
            dst "+el['ip']+" flowid 1:10"
265         print(faststring)
266         os.system(faststring)
267         shapelist[el['ip']]['comm'].append(faststring)
268         faststring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip
            src "+el['ip']+" flowid 1:10"
269         print(faststring)
270         os.system(faststring)
271     else:
272
273         midstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip
            dst "+el['ip']+" flowid 1:20"
274         print(midstring)
275         os.system(midstring)
276         midstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip
            src "+el['ip']+" flowid 1:20"
277         print(midstring)
278         os.system(midstring)
279         shapelist[el['ip']]['class']='med'
280
281
282
283     continue
284 except:
285     print((traceback.format_exc()))
286     print("faulire with tree")
287     exit()
288
289 if (el['count']> el['length'] or el['dests']>4 or el['count']> 300) and ('class' not in el or
    el['class'] == 'node'):
290
291     slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst "+
        el['ip']+" flowid 1:30"
292     print(slowstring)
293     os.system(slowstring)
294     slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src "+

```

```

        el['ip']+" flowid 1:30"
295     print(slowstring)
296     os.system(slowstring)
297 elif el['class']=='gateway' or el['class']=='bridge':
298
299     faststring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst "+
        el['ip']+" flowid 1:10"
300     print(faststring)
301     os.system(faststring)
302     faststring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src "+
        el['ip']+" flowid 1:10"
303     print(faststring)
304     os.system(faststring)
305     os.system(faststring)
306 else:
307
308     midstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst "+
        el['ip']+" flowid 1:20"
309     print(midstring)
310     os.system(midstring)
311     os.system(midstring)
312     midstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src "+
        el['ip']+" flowid 1:20"
313     print(midstring)
314     os.system(midstring)
315     os.system(midstring)
316
317
318 for el in sublist:
319     if el['count']>300 and el['ip'] not in shapelist:
320         shapelist[el['ip']]=dict()
321         shapelist[el['ip']]['comm']=[]
322         print("leaf prdction")
323         slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst "+
            el['ip']+" flowid 1:30"
324         print(slowstring)
325         os.system(slowstring)
326         shapelist[el['ip']]['comm'].append(slowstring)
327
328         slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip src "+
            el['ip']+" flowid 1:30"
329         print(slowstring)
330         os.system(slowstring)
331         shapelist[el['ip']]['comm'].append(slowstring)
332 with open("shapefile"+ip+".json",'w') as outfile:
333     print(json.dump(shapelist,outfile))

autosshape9.py
1 import traceback
2 import json
3 import pyshark
4 from collections import Counter
5 import os
6 import datetime
7 #cap = pyshark.LiveCapture(interface='wlan1')
8 #cap.sniff(packet_count=50)
9 from sklearn import tree
10 import netifaces as ni
11 from joblib import dump, load

```

```

12 import math
13 bridgeflag=0
14
15 def chooseInterface():
16     import netifaces as ni
17     chosenInterface='wlp3s0'
18     for interface in ni.interfaces():
19         print( str(interface))
20         try:
21             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
22             if '172.27.0.' in ip:
23                 chosenInterface=interface
24             if 'br' in interface: bridgeflag=1
25         except:
26             print( "eror reading interface")
27             print( "chosen interface is "+str(chosenInterface))
28     return chosenInterface
29 interface=chooseInterface()
30 ni.ifaddresses(interface)
31 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
32 clf = tree.DecisionTreeClassifier()
33 try:
34     clf = load('tree'+ip+'.joblib')
35 except:
36     #continue
37     print("error reading data")
38
39 print(ip) # should print "192.168.100.37"
40 packetlog=dict();
41 userlog=dict();
42 storedlog=dict();
43 templog=dict();
44
45
46 import glob
47 for filename in glob.glob('iplog*'):
48     print(filename)
49     #if ip in filename: continue;
50     #if ip in filename: continue;
51
52
53     with open(filename,'r')as f:
54         try:
55             templog= json.load(f)
56         except:
57             continue
58         #print((str(templog)))
59         #storedlog={x: storedlog.get(x,0)+templog.get(x,0) for x in set(storedlog).union(templog)}
60         #if templog[]
61         for key,val in list(templog.items()):
62
63
64             if key in storedlog:
65                 print("key is in stored log "+str(key))
66                 if '172.27.0' not in key: continue
67                 if not isinstance(storedlog[key],dict): storedlog[key]=dict()
68                 if 'remoteadded' not in storedlog[key]: stored[key]['remoteadded']=list()
69                 if isinstance(storedlog[key],dict) and 'remoteadded' in storedlog[key]:
70                     if isinstance(storedlog[key]['remoteadded'],list):

```

```

71         #if filename[(filename.find('g')+1):(filename.find('.j'))] not in
           storedlog[key]['remotedded']: storedlog[key]['remotedded'].append
           (filename[(filename.find('g')+1):(filename.find('.j'))])
72         if 'originip' in templog and templog['originip'] not in storedlog[key
           ]['remotedded']: storedlog[key]['remotedded'].append(templog['
           originip'])
73
74     if 'remotedded' in val:
75         for ij in val['remotedded']:
76             if ij not in storedlog[key]['remotedded']:storedlog[key]['remotedded
           '].append(ij)
77
78
79
80
81
82     if 'route' not in storedlog[key]:
83         storedlog[key]['route']=[]
84     if 'route' in storedlog[key] and 'route' in val:
85
86         #storedlog[key]['route']=val['route']+storedlog[key]['route']
87         for il in val['route']:
88             if il not in storedlog[key]['route']: storedlog[key]['route'].append(il
           )
89         #if filename[(filename.find('g')+1):(filename.find('.j'))] not in storedlog
           [key]['route']: storedlog[key]['route'].append(filename[(filename.find
           ('g')+1):(filename.find('.j'))]) #remove later
90     if 'route' in storedlog[key] and 'route' not in val:
91         print("adding file name")
92
93         #if filename[(filename.find('g')+1):(filename.find('.j'))] not in storedlog
           [key]['route']: storedlog[key]['route'].append(filename[(filename.find
           ('g')+1):(filename.find('.j'))])
94
95     #print("debug route "+str(storedlog[key]['route']))
96     #storedlog[key]['remotedded']=filename[5:17]
97     #try:
98     #print "doing addition "+str(storedlog[key])+ " and "+str(val)
99     ','
100    storedlog[key]['count']= storedlog[key]['count']+val['count']
101    for key1,val1 in list(val.items()):
102        if not isinstance(val1,dict) or 'count' not in val1:
103            storedlog[key][key1]=val1
104            continue
105        print("key 1 is "+str(key1)+" val 1 is"+str(val1))
106        if key1=='count': continue
107        if key1 in storedlog[key] and 'count' in storedlog[key][key1]:
108            storedlog[key][key1]=val1
109            if storedlog[key][key1]['count']!=val1['count']:
110                storedlog[key][key1]['count']= storedlog[key][key1]['count']+val1['
           count']
111        else:
112            storedlog[key][key1]['count']= storedlog[key][key1]['count']
113        elif key1 in storedlog[key]:
114            storedlog[key][key1]=val1
115    ','
116
117
118

```

```

119             #print "result is "+str(storedlog[key])
120         #except:
121             #print "error doing addition"
122             #exit()
123
124
125     else:
126         print("adding new key to stored log")
127         #val['remoteadded']=filename[5:17]
128         if isinstance(val,dict) and 'remoteadded' in val:
129             if isinstance(val['remoteadded'],list):
130                 val['remoteadded'].append(templog['originip'])
131             else:
132
133                 val['remoteadded']=[templog['originip']]
134         elif isinstance(val,dict):
135             if 'originip' in templog: val['remoteadded']=[templog['originip']]
136             else:val['remoteadded']=[]# fix later
137
138         print("adding value "+str(val))
139         storedlog[key]=val
140     #os.system('mv '+filename+' '+oldlog+filename[4:])
141
142     print("printing new log")
143     print((json.dumps(storedlog)))
144
145     userlog=dict(storedlog)
146     #if ip in storedlog: userlog=storedlog[ip]
147     storedlogbackup=dict(storedlog)
148     #with open("datathing1.json",'r')as f:
149     # userlog= json.load(f)
150
151     externpacketlog=dict()
152     #with open("externthing1.json",'r')as f:
153     # externpacketlog= json.load(f)
154     #print json.dumps(userlog)
155     internpacketlog=dict()
156     #with open("internthing1.json",'r')as f:
157     # internpacketlog= json.load(f)
158     #userlog=dict();
159     print((json.dumps(storedlog)));
160     #print json.dumps(userlog);
161     #print json.dumps(internpacketlog)
162     #print json.dumps(externpacketlog)
163     print("sniffing packets")
164     errorcount =0;
165     cap = pyshark.LiveCapture(interface=interface)
166     print("sniffed packets")
167     cap.sniff(packet_count=500)
168     for packet in cap:
169         ipstr='ip'
170         if ipstr not in packet: ipstr='ipv6'
171         #print packet
172         #print packet.highest_layer
173         try:
174             if ('tcp' in packet or 'udp' in packet) and ('172.27.0' in str(packet[ipstr].src)):
175                 print("key is "+str(packet[ipstr].src))
176                 #print(packet)
177                 #print packet.highest_layer

```

```

178 #print packet.tcp.dstport
179 print (packet[ipstr].src)
180
181 try:
182     if packet[ipstr].src in userlog:
183         print("adding to dict again ")
184         #packetlog[packet.tcp.dstport] = 1+packetlog[packet.tcp.dstport];
185         print(("added to user log "+str(packet[ipstr].src)+" "+str(userlog[packet[
186             ipstr].src])))
187         userlog[packet[ipstr].src]['count'] = userlog[packet[ipstr].src]['count']+1;
188         #userlog[packet[ipstr].src]['length'] = int(packet.length)
189         userlog[packet[ipstr].src]['totalsize'] = int(packet.length)+int(userlog[
190             packet[ipstr].src]['totalsize'])
191         userlog[packet[ipstr].src]['length'] = int(userlog[packet[ipstr].src]['
192             totalsize']/userlog[packet[ipstr].src]['count'])
193         #userlog[packet[ipstr].src]['time'].append(str((packet.sniff_timestamp)))
194         if 'route' not in userlog[packet[ipstr].src] and packet[ipstr].src != ip and
195             packet[ipstr].dst != ip:
196             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
197                 ipstr].dst))
198             print("adding route")
199             userlog[packet[ipstr].src]['route']=[ip]
200         elif packet[ipstr].src != ip and packet[ipstr].dst != ip and ip not in userlog
201             [packet[ipstr].src]['route']:
202             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
203                 ipstr].dst))
204             print("adding route")
205             userlog[packet[ipstr].src]['route'].append(ip)
206         if packet.highest_layer not in userlog[packet[ipstr].src]['protocol']:
207             print("adding new protocol")
208             userlog[packet[ipstr].src]['protocol'][packet.highest_layer]=1
209         else:
210             userlog[packet[ipstr].src]['protocol'][packet.highest_layer]=userlog[packet
211                 [ipstr].src]['protocol'][packet.highest_layer]+1
212
213         if packet[ipstr].dst in userlog[packet[ipstr].src]:
214             #userlog[packet[ipstr].src]+1;
215             print(("incrementing user log before "+str(userlog[packet[ipstr].src][
216                 packet[ipstr].dst])))
217             userlog[packet[ipstr].src][packet[ipstr].dst]['count']= userlog[packet[
218                 ipstr].src][packet[ipstr].dst]['count']+1;
219             print(("incrementing user log after "+str(userlog[packet[ipstr].src][
220                 packet[ipstr].dst])))
221             if 'time' in userlog[packet[ipstr].src][packet[ipstr].dst]: userlog[
222                 packet[ipstr].src][packet[ipstr].dst]['time'].append(str(packet.
223                     sniff_timestamp))
224             else: userlog[packet[ipstr].src][packet[ipstr].dst]['time']= [str(
225                 packet.sniff_timestamp)];
226         else:
227             userlog[packet[ipstr].src][packet[ipstr].dst]= dict();
228             userlog[packet[ipstr].src][packet[ipstr].dst]['count']= 1;
229             userlog[packet[ipstr].src][packet[ipstr].dst]['time']= [str(packet.sniff_
230                 timestamp)];
231
232         #if packet[ipstr].src== "172.27.0.90":
233         #if packet[ipstr].src == "172.27.0.90" or packet[ipstr].src== "172.27.0.155"
234             or packet[ipstr].src == "172.27.0.15" or packet[ipstr].src
235             == "172.27.0.227":

```

```

220         #internpacketlog[packet[ipstr].src] = internpacketlog[packet[ipstr].src]+1;
221     #else:
222         #externpacketlog[packet[ipstr].dst] = externpacketlog[packet[ipstr].dst]+1;
223
224     else:
225         #print "adding to dict"
226         #packetlog[packet.tcp.dstport]=1;
227
228         #userlog[packet[ipstr].src]=1;
229         userlog[packet[ipstr].src]=dict()
230         userlog[packet[ipstr].src]['count']=1
231         userlog[packet[ipstr].src]['totalsize']=packet.length
232         userlog[packet[ipstr].src]['protocol']=dict()
233         userlog[packet[ipstr].src]['protocol'][packet.highest_layer]=1
234         userlog[packet[ipstr].src][packet[ipstr].dst]= dict();
235         userlog[packet[ipstr].src][packet[ipstr].dst]['count']= 1;
236         if 'route' not in userlog[packet[ipstr].src] and packet[ipstr].src != ip and
            packet[ipstr].dst != ip:
237             print("adding route")
238             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
                ipstr].dst))
239             userlog[packet[ipstr].src]['route']=[ip]
240         elif packet[ipstr].src != ip and packet[ipstr].dst != ip and ip not in userlog
            [packet[ipstr].src]['route']:
241             print("adding route")
242             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
                ipstr].dst))
243             userlog[packet[ipstr].src]['route'].append(ip)
244
245         ',,'
246         elif packet[ipstr].src != ip:
247             print("appending route")
248             userlog[packet[ipstr].src]['route'].append(ip)
249         ',,'
250         #else:
251             #userlog[packet[ipstr].src]['route'].append(ip)
252
253
254         userlog[packet[ipstr].src]['port']=dict()
255         #userlog[packet[ipstr].src]['time']=[str(packet.sniff_timestamp)]
256         if 'dstport' in packet:
257             userlog[packet[ipstr].src]['port'][packet[ipstr].dstport]=1
258
259
260     except:
261         errorcount=errorcount+1;
262         print("error with fields")
263         print((traceback.format_exc()))
264         #print packet
265     if ('tcp' in packet or 'udp' in packet) and ('172.27.0' in str(packet[ipstr].dst)):
266         print("key is "+str(packet[ipstr].dst))
267         #print(packet)
268         #print packet.highest_layer
269         #print packet.tcp.dstport
270         print (packet[ipstr].dst)
271
272     try:
273         if packet[ipstr].dst in userlog:
274             print("adding to dict again ")

```



```

275     #packetlog[packet.tcp.dstport] = 1+packetlog[packet.tcp.dstport];
276     print(("added to user log "+str(packet[ipstr].dst)+" "+str(userlog[packet[
    ipstr].dst])))
277     userlog[packet[ipstr].dst]['count'] = userlog[packet[ipstr].dst]['count']+1;
278     #userlog[packet[ipstr].dst]['length'] = int(packet.length)
279     userlog[packet[ipstr].dst]['totalsize'] = int(packet.length)+int(userlog[
    packet[ipstr].dst]['totalsize'])
280     userlog[packet[ipstr].dst]['length'] = int(userlog[packet[ipstr].dst]['
    totalsize']/userlog[packet[ipstr].dst]['count'])
281     if 'route' not in userlog[packet[ipstr].dst] and packet[ipstr].dst != ip and
    packet[ipstr].src != ip:
282         print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
    ipstr].dst))
283         userlog[packet[ipstr].dst]['route']=[ip]
284     elif 'route' in userlog[packet[ipstr].dst] and packet[ipstr].dst != ip and ip
    not in userlog[packet[ipstr].dst]['route'] and packet[ipstr].src != ip:
285         print("appending route ")
286         print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
    ipstr].dst))
287         userlog[packet[ipstr].dst]['route'].append(ip)
288     if packet.highest_layer not in userlog[packet[ipstr].dst]['protocol']:
289         print("adding new protocol")
290         userlog[packet[ipstr].dst]['protocol'][packet.highest_layer]=1
291     else:
292         userlog[packet[ipstr].dst]['protocol'][packet.highest_layer]=userlog[packet
    [ipstr].dst]['protocol'][packet.highest_layer]+1
293
294
295     if packet[ipstr].src in userlog[packet[ipstr].dst]:
296         #userlog[packet[ipstr].dst]+1;
297         print(("incrementing user log before "+str(userlog[packet[ipstr].dst][
    packet[ipstr].src])))
298         userlog[packet[ipstr].dst][packet[ipstr].src]['count']= userlog[packet[
    ipstr].dst][packet[ipstr].src]['count']+1;
299         print(("incrementing user log after "+str(userlog[packet[ipstr].dst][
    packet[ipstr].src])))
300         if 'time' in userlog[packet[ipstr].dst][packet[ipstr].src]['time']:
    userlog[packet[ipstr].dst][packet[ipstr].src]['time'].append(str(
    packet.sniff_timestamp));
301         else: userlog[packet[ipstr].dst][packet[ipstr].src]['time']= [str(
    packet.sniff_timestamp)];
302     else:
303         userlog[packet[ipstr].dst][packet[ipstr].src]= dict();
304         userlog[packet[ipstr].dst][packet[ipstr].src]['count']= 1;
305         userlog[packet[ipstr].dst][packet[ipstr].src]['time']= [str(packet.sniff_
    timestamp)];
306
307     #if packet[ipstr].src== "172.27.0.90":
308     #if packet[ipstr].src == "172.27.0.90" or packet[ipstr].src== "172.27.0.155"
    or packet[ipstr].src == "172.27.0.15" or packet[ipstr].src
    == "172.27.0.227":
309         #internpacketlog[packet[ipstr].src] = internpacketlog[packet[ipstr].src]+1;
310     #else:
311         #externpacketlog[packet[ipstr].dst] = externpacketlog[packet[ipstr].dst]+1;
312
313     else:
314         #print "adding to dict"
315         #packetlog[packet.tcp.dstport]=1;
316

```

```

317         #userlog[packet[ipstr].src]=1;
318         userlog[packet[ipstr].dst]=dict()
319         userlog[packet[ipstr].dst]['count']=1
320         userlog[packet[ipstr].dst]['totalsize']=packet.length
321         userlog[packet[ipstr].dst]['protocol']=dict()
322         userlog[packet[ipstr].dst]['protocol'][packet.highest_layer]=1
323         userlog[packet[ipstr].dst][packet[ipstr].src]= dict();
324         userlog[packet[ipstr].dst][packet[ipstr].src]['count']= 1;
325         userlog[packet[ipstr].src]['port']=dict()
326         if 'route' not in userlog[packet[ipstr].dst] and packet[ipstr].dst != ip and
            packet[ipstr].src != ip:
327             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
                ipstr].dst))
328             userlog[packet[ipstr].dst]['route']=[ip]
329         elif packet[ipstr].dst != ip and packet[ipstr].src != ip and ip not in userlog
            [packet[ipstr].dst]['route']:
330             print("appending route " +str(ip)+" ")
331             print("adding route "+str(ip)+" "+str(packet[ipstr].src)+" "+str( packet[
                ipstr].dst))
332             userlog[packet[ipstr].dst]['route'].append(ip)
333
334         #userlog[packet[ipstr].src]['time']=[str(packet.sniff_timestamp)]
335         if 'srcport' in packet:
336             userlog[packet[ipstr].src]['port'][packet[ipstr].srcport]=1
337
338
339     except:
340         errorcount=errorcount+1;
341         print("error with fields")
342         print((traceback.format_exc()))
343         #print packet
344
345 except:
346     print("error ")
347     print((traceback.format_exc()))
348     #print packet
349
350 #print json.dumps(packetlog)
351 #import netifaces as ni
352 #ni.ifaddresses('wlan1')
353 #ip = ni.ifaddresses('wlan1')[ni.AF_INET][0]['addr']
354 #print ip # should print "192.168.100.37
355 for key,val in list(userlog.items()):
356     if not isinstance(val,dict)or 'route' not in val or not isinstance(val['route'],list): continue
357     for el in val['route']:
358         print("checking route "+el)
359         routecount=0
360         for key1,val1 in userlog.items():
361             print(val1)
362             if isinstance(val1,dict) and 'route' in val1 and isinstance(val1['route'],list) and el in
                val1['route'] and key1 != el:
363                 if len(val1['protocol'])>2 or val1['count']>20:
364                     routecount=routecount+1+len(val1['protocol'])
365         if el not in userlog:
366             userlog[el]=dict()
367         print("adding route count to "+str(el)+" which is "+str(routecount) )
368         if 'routecount' not in userlog[el] or userlog[el]['routecount']<routecount: userlog[el]['
            routecount']=routecount
369

```

```

370 arplog=dict()
371 try:
372     with open("fullarp"+ip+".json",'r')as f:
373         arplog= json.load(f)
374 except:
375     print("error reading file")
376 classifierinit=0
377 try:
378     clf = load('tree'+ip+'.joblib')
379     classifierinit=1
380 except:
381     print("error with classifier")
382 for key,val in list(userlog.items()):
383     if isinstance(val,dict) and 'route' in val:
384         for el in list(val['route']):
385             print("checking "+str(el))
386             treepred='node'
387             if el in userlog:
388                 if (el in userlog and 'class' in userlog[el] and userlog[el]['class'] != 'node') or
389                     classifierinit ==0 : continue
390                 try:
391                     pdata=[]
392                     val=userlog[el]
393                     if 'count' in val and 'length' in val: pdata.append(val['count']*int(math.log(val
394                         ['length'])))
395                     else: pdata.append(0)
396                     if 'count' in val: pdata.append(val['count'])
397                     else: pdata.append(0)
398                     if 'length' in val: pdata.append(val['length'])
399                     else: pdata.append(0)
400                     #pdata.append(0)
401                     dests=0
402                     #if 'class' in val: pdata['class']=int(val['class'])
403                     #else: pdata.append(int('node'))
404                     if 'routecount' in val: pdata.append(val['routecount'])
405                     else: pdata.append(0)
406                     for key1,val1 in val.items():
407                         print(key1)
408                         if '172.27.0' in key1:
409                             dests=dests+1
410                     pdata.append(dests)
411                 '''
412                 pdata=[]
413                 if 'length' in val:
414                     pdata.append((val['count']*int(math.log(val['length']))))
415                     pdata.append(val['count'])
416                     pdata.append(val['length'])
417                     #pdata.append(0)
418                     dests=0
419                     #if 'class' in val: pdata['class']=int(val['class'])
420                     #else: pdata.append(int('node'))
421                     if 'routecount' in val: pdata.append(val['routecount'])
422                     else: pdata.append(0)
423                     for key1,val1 in val.items():
424                         #print(key1)
425                         if '172.27.0' in key1:
426                             dests=dests+1
427                     pdata.append(dests)

```

```

427         try:
428             arpdict=dict()
429             with open("arpfile"+ip+".json") as f:
430                 arpdict=json.load(f)
431             if key in arpdict:
432                 pdata.append(len(arpdict[key]))
433             else:
434                 pdata.append(0)
435         except:
436             pdata.append(0)
437
438         treepred=clf.predict([pdata])[0]
439         print("tree pred is"+treepred+" for "+str(el))
440         userlog[el]['class']=treepred
441         #break
442         continue
443     except:
444         if 'routecount' in userlog[el] and userlog[el]['routecount']>6: userlog[el]['class
445             ']='gateway'
446         else: userlog[el]['class']='node'
447
448     if el not in userlog and '172.27.0.' in el:
449         userlog[el]=dict()
450         userlog[el]['class']='bridge'
451     elif '172.27.0.' in el:
452         if 'routecount' in userlog[el] and userlog[el]['routecount']>6: userlog[el]['class']='
453             gateway'
454         else: userlog[el]['class']='node'
455
456     for key,val in userlog.items():
457         if isinstance(val,dict) and( 'class' not in val or val['class']=='node'):
458             if 'routecount' in userlog[key] and userlog[key]['routecount']>=3: userlog[key]['
459                 class']='gateway'
460             else: userlog[key]['class']='node'
461             for key1,val1 in val.items():
462                 if '172.27.0.' in key1 and key1 in userlog:
463                     if 'routecount' in userlog[key1] and userlog[key1]['routecount']>=3: userlog[
464                         key1]['class']='gateway'
465                     else: userlog[key1]['class']='node'
466                 elif '172.27.0.' in key1 and key1 not in userlog:userlog[key1]['class']='node'
467
468     savedLog= dict();
469     #storedlog=userlog;
470
471     print((json.dumps(storedlogbackup)))
472     print((json.dumps(storedlog)))
473     #print json.dumps(internpacketlog)
474     #print json.dumps(externpacketlog)
475     import random
476     dictid=random.randint(1,21)*5
477     userlog['id']=dictid
478     userlog['originip']=ip
479     userlog['bridge']=bridgeflag
480
481     packetfile='iplog'+ip+'.json';
482     with open(packetfile,'w') as outfile:
483         json.dump(userlog,outfile)

```

```

482 with open('datathing1.json','w') as outfile:
483     json.dump(userlog,outfile)
484
485
486 import os
487 print("new val is")
488 os.system('cat '+packetfile)
489
490 print("\n\n\nold val is "+str(storedlog))
491 exit()
492 #os.system('cat internthing1.json')
493 #os.system('cat externthing1.json')

```

B.5 genarp5.py

```

1
2 import python_arptable
3 from python_arptable import ARPTABLE
4 import netifaces as ni
5 import json
6 import os
7 import socket
8
9 def valid_ip(address):
10     try:
11         socket.inet_aton(address)
12         return True
13     except:
14         return False
15 def chooseInterface():
16     import netifaces as ni
17     chosenInterface='wlp3s0'
18     for interface in ni.interfaces():
19         print(( str(interface)))
20         try:
21             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
22             if '172.27.0.' in ip:
23                 chosenInterface=interface
24         except:
25             print( "error reading interface")
26             print(( "chosen interface is "+str(chosenInterface)))
27     return chosenInterface
28 interface=chooseInterface()
29 gateway=''
30 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
31 for key,val in (ni.gateways()).items():
32     if isinstance(val,list):
33         for el in val:
34             if interface in el:
35                 print("gateway is "+el[0])
36                 gateway=el[0]
37     else:
38         if interface in val:
39             print("gateway is"+val[0])
40             gateway=val[0]
41 print("gateway is "+gateway)
42 if gateway == '':
43     templog=dict()

```

```

44     with open('iplog'+ip+'.json','r') as infile:
45         templog=json.load(infile)
46     print(templog)
47     if ip in templog:
48         if 'class' in templog[ip]:
49             if templog[ip]['class']=='gateway' or templog[ip]['class']=='gate':
50                 gateway=ip
51     if gateway == '':
52         gateway=ip
53     print(ARPTABLE)
54     arpdict=ARPTABLE
55     print(arpdict)
56     iplist=[gateway]
57
58     print(iplist)
59     ipdict=dict()
60     ipdict[ip]=iplist
61     print(ipdict)
62
63     print(ARPTABLE)
64     arpdict=ARPTABLE
65     print(arpdict)
66     iplist=[gateway]
67
68     #iplist.append(ip)
69     for di in arpdict:
70         print(di)
71         if '172.27.0' in di['IP address'] and di['IP address'] not in iplist and di['HW address'] !=
            '00:00:00:00:00:00' and di['Device']==interface:
72             response = os.system("ping -c 1 " + di['IP address'])
73             print(response)
74             if response ==0:
75                 print("hostup "+str(e1))
76                 iplist.append(di['IP address'])
77             else: print("error")
78
79     ipdict[ip]=iplist
80     print(iplist)
81     import glob
82     for filename in glob.glob('arpfile*'):
83         print("filename")
84
85
86
87
88
89
90     with open(filename,'r')as f:
91         try:
92             templog= json.load(f)
93             print((str(templog)))
94             for key,val in list(templog.items()):
95                 if valid_ip(key): ipdict[key]=val
96
97                 for i in val:
98                     response = os.system("ping -c 1 " +i)
99                     print(response)
100                    if response ==0:
101                        if i not in iplist: iplist.append(i)

```

```

102
103
104             print("hostup "+i)
105
106         except:
107             continue
108
109     with open("arpfile"+ip+".json",'w') as outfile:
110         print(json.dump(ipdict,outfile))
111     #file=open("iplist.txt","w")
112     #for ip in iplist:
113     # file.write(ip+",")
114     dictlist=dict()
115     dictlist['ips']=iplist
116     print(dictlist)
117     with open("iplist.json",'w') as outfile:
118         print(json.dump(dictlist,outfile))
119
120     file=open("iplist.txt","w")
121     for ip in iplist:
122         file.write(ip+",")

```

B.6 comparearpgraphs3.py

```

1 import traceback
2 import json
3 import pyshark
4 from collections import Counter
5 import os
6 import matplotlib
7 matplotlib.use('Agg')
8 import pandas as pd
9 import numpy as np
10 import networkx as nx
11 import matplotlib.pyplot as plt
12 import json
13 import scipy
14 #cap = pyshark.LiveCapture(interface='wlan1')
15 #cap.sniff(packet_count=50)
16 def valid_ip(address):
17     try:
18         socket.inet_aton(address)
19         return True
20     except:
21         return False
22 import netifaces as ni
23 def chooseInterface():
24     import netifaces as ni
25     chosenInterface='wlp3s0'
26     for interface in ni.interfaces():
27         print( str(interface))
28         try:
29             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
30             if '172.27.0.' in ip:
31                 chosenInterface=interface
32     except:
33         print( "error reading interface")
34     print( "chosen interface is "+str(chosenInterface))

```

```

35         return chosenInterface
36 interface=chooseInterface()
37 ni.ifaddresses(interface)
38 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
39 print(ni.gateways())
40 gateway=''
41 for key,val in (ni.gateways()).items():
42     if isinstance(val,list):
43         for el in val:
44             if interface in el:
45                 print("gateway is "+el[0])
46                 gateway=el[0]
47     else:
48         if interface in val:
49             print("gateway is"+val[0])
50             gateway=val[0]
51 print("gateway is "+gateway)
52 #exit()
53 print(ip) # should print "192.168.100.37"
54 packetlog=dict();
55 userlog=dict();
56 storedlog=dict();
57 templog=dict();
58 fromlist=[]
59 tolist=[]
60
61
62
63 try:
64     with open("arpfile"+ip+".json",'r')as f:
65         templog= json.load(f)
66         print((str(templog)))
67         for key,val in list(templog.items()):
68             print("is key "+str(key))
69             if valid_ip(key) or '172.27.0' in key:
70                 storedlog[key]=val
71                 for i in val:
72                     if valid_ip(key) or '172.27.0' in key:
73                         fromlist.append(key)
74                         tolist.append(i)
75 except:
76     exit()
77     #continue
78 '''
79 with open("fullarp"+ip+".json",'w') as outfile:
80     print(json.dump(storedlog,outfile))
81 '''
82 print(fromlist)
83 print(tolist)
84 df = pd.DataFrame({'from':fromlist, 'to':tolist})
85 df
86
87 # Build your graph. Note that we use the DiGraph function to create the graph!
88 G=nx.from_pandas_edgelist(df, 'from', 'to', create_using=nx.DiGraph() )
89
90 # Make the graph
91 pos = nx.spring_layout(G,scale=100,k=1,iterations=20)
92 #layout = nx.kamada_kawai_layout(G, dist=20)
93 '''

```



```

94 df = pd.DataFrame(index=G.nodes(), columns=G.nodes())
95 for row, data in nx.shortest_path_length(G):
96     for col, dist in data.items():
97         df.loc[row,col] = dist*2
98 df = df.fillna(df.max().max())
99 layout = nx.kamada_kawai_layout(G, dist=df.to_dict())
100 '''
101 nx.draw(G, layout=pos,with_labels=True,node_size=100, alpha=0.3, arrows=True, edge_color='black',
        node_color='black')
102 plt.axis("off")
103 #nx.draw(G, with_labels=True, node_size=1500, alpha=0.3, arrows=True)
104 #pos = nx.spring_layout(G,k=0.6,scale=4)
105 #plt.show()
106 #plt.draw()
107 plt.savefig("graph"+ip+".png",dpi=100)
108 #plt.show()
109 #plt.show()
110 plt.close()

```

B.7 pytc5.py

```

1 def chooseInterface():
2     import netifaces as ni
3     chosenInterface='mesh0'
4     for interface in ni.interfaces():
5         print(( str(interface)))
6         try:
7             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
8             if '172.27.0.' in ip:
9                 chosenInterface=interface
10        except:
11            print( "error reading interface")
12            print(( "chosen interface is "+str(chosenInterface)))
13        return chosenInterface
14 interface=chooseInterface()
15 print("starting script")
16 import json
17 prioritizedict=dict()
18 with open('tcdirs.json','r')as f:
19     prioritizedict= json.load(f)
20 print(json.dumps(prioritizedict))
21
22 prioritizelist=prioritizedict["prioritize"]
23 slowlist=prioritizedict["slow"]
24 print(prioritizelist)
25
26 for i in prioritizelist:
27     print(i)
28
29 import os
30 import sys
31
32 os.system("sudo tc qdisc del dev "+interface+" root")
33 os.system("tc qdisc add dev "+interface+" root handle 1: htb default 30")
34 os.system("tc class add dev "+interface+" parent 1: classid 1:1 htb rate 6mbit burst 15k")
35 os.system("tc class add dev "+interface+" parent 1:1 classid 1:10 htb rate 5mbit burst 15k")
36 os.system("tc class add dev "+interface+" parent 1:1 classid 1:20 htb rate 3mbit ceil 6mbit burst
        15k")

```

```

37 os.system("tc class add dev "+interface+" parent 1:1 classid 1:30 htb rate 10kbit ceil 10kbit burst
    10k")
38 os.system("tc qdisc add dev "+interface+" parent 1:10 handle 10: sfq perturb 10")
39 os.system("tc qdisc add dev "+interface+" parent 1:20 handle 20: sfq perturb 10")
40 os.system("tc qdisc add dev "+interface+" parent 1:30 handle 30: sfq perturb 10")
41 os.system("tc filter add dev "+interface+" protocol ip parent 1: prio 1 u32 match ip dport 22 0
    xffff flowid 1:10")
42 os.system("tc filter add dev "+interface+" parent 1: protocol ip prio 1 u32 match ip dst 10.0.0.97
    match ip dport 1234 0xffff flowid 1:30")
43
44 for i in prioritizelist:
45     slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst "+i+"
        flowid 1:10"
46     os.system(slowstring)
47     print(slowstring)
48
49 for i in slowlist:
50     slowstring="tc filter add dev "+interface+" parent 1: protocol ip prio 2 u32 match ip dst "+i+"
        flowid 1:30"
51     os.system(slowstring)
52     print(slowstring)

```

B.8 trafficplot7.py

```

1 #import matplotlib.pyplot as plt
2 import matplotlib
3 matplotlib.use('Agg')
4 import matplotlib.pyplot as plt
5 import numpy as np
6 import pandas as pd
7 from io import StringIO
8 import json
9 import math
10 import netifaces as ni
11 import socket
12
13 def valid_ip(address):
14     try:
15         socket.inet_aton(address)
16         return True
17     except:
18         return False
19 def chooseInterface():
20     import netifaces as ni
21     chosenInterface='wlp3s0'
22     for interface in ni.interfaces():
23         print(( str(interface)))
24         try:
25             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
26             if '172.27.0.' in ip:
27                 chosenInterface=interface
28         except:
29             print( "error reading interface")
30             print(( "chosen interface is "+str(chosenInterface)))
31     return chosenInterface
32
33 interface=chooseInterface()
34 ni.ifaddresses(interface)

```

```

35 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
36 data=dict()
37 with open('iplog'+ip+'.json','r')as f:
38     data= json.load(f)
39     print(str(data))
40     datalist=[]
41     for key,val in data.items():
42         try:
43             if not valid_ip(key): continue
44
45
46             pdata=dict()
47             #print(str(key)+" "+str(val['count']))
48             pdata['ip']=key
49             #if 'length' in val:
50             if 'length'in val and 'count' in val: pdata['countlen']=val['count']*val['length']
51             else: pdata['countlen']=0
52             if 'count' in val:pdata['count']=val['count']
53             else: pdata['count']=0
54             if 'length'in val :pdata['length']=val['length']
55             else: pdata['length']=0
56             if 'route' in val: pdata['routes']=val['route']
57             else: pdata['routes']=0
58             datalist.append(pdata)
59         #pdata['totalsize']=int((val['totalsize']))
60         #else:
61             #pdata['count']=val['count']
62     except:
63         #print(str(key)+" "+str(val['count']))
64         continue;
65     #print("appending to list "+str(pdata))
66     #datalist.append(pdata)
67 '''
68 for ind,el in enumerate(datalist):
69     print(str(el))
70     routecount=0
71     locip=el['ip']
72     for key,val in data.items():
73         print("is val "+str(val))
74         if not isinstance(val,dict): continue
75         if 'route' in val and locip in val['route']: routecount=routecount+1
76     el['routes']=routecount
77 '''
78 print(str(datalist))
79 df = pd.DataFrame(datalist)
80 #ax=df.plot.add_subplot()
81 '''
82 ax1 = df1.plot()
83 ax2 = ax1.twinx()
84 ax2.spines['right'].set_position(('axes', 1.0))
85 df2.plot(ax=ax2)
86 ax3 = ax1.twinx()
87 ax3.spines['right'].set_position(('axes', 1.1))
88 df3.plot(ax=ax3)
89 #plt.show()
90 '''
91 '''
92 import pylab as pl
93 fig = pl.figure()

```

```

94 ax1 = pl.subplot(111,ylabel='count')
95 #ax2 = gcf().add_axes(ax1.get_position(), sharex=ax1, frameon=False, ylabel='axes2')
96 ax2 =ax1.twinx()
97 ax2.set_ylabel('length')
98 ax1.bar(df.index,df.count, width =0.4, color = 'g', align = 'center')
99 ax2.bar(df.index,df.length, width = 0.4, color='r', align = 'edge')
100 ax1.legend(['count'], loc = 'upper left')
101 ax2.legend(['length'], loc = 'upper right')
102 fig.show()
103 '''
104 #fig = plt.figure() # Create matplotlib figure
105
106 #ax = fig.add_subplot(111) # Create matplotlib axes
107 #ax2 = ax.twinx() # Create another axes that shares the same x-axis as ax.
108
109 #width = 0.4
110
111 #df['count'].plot(kind='bar', color='red',x='ip', ax=ax, width=width, position=1)
112 #df['length'].plot(kind='bar', color='blue',x='ip', ax=ax2, width=width, position=0)
113
114 #ax.set_ylabel('count')
115 #ax2.set_ylabel('length')
116 plot1=df.plot(kind='bar',x='ip', y=['count', 'routes'],figsize=(30,30),fontsize=25,secondary_y='
    routes')
117 #df.plot(kind='bar',secondary_y=True,x='ip',y='length')
118 fig=plot1.get_figure()
119 fig.savefig("griplog"+ip+".png")
120
121 #plt.show()

```

B.9 generatefullarp.py

```

1 import traceback
2 import json
3 import pyshark
4 from collections import Counter
5 import os
6 import python_arptable
7 from python_arptable import ARPTABLE
8
9 #cap = pyshark.LiveCapture(interface='wlan1')
10 #cap.sniff(packet_count=50)
11 import netifaces as ni
12 def chooseInterface():
13     import netifaces as ni
14     chosenInterface='wlp3s0'
15     for interface in ni.interfaces():
16         print( str(interface))
17         try:
18             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
19             if '172.27.0.' in ip:
20                 chosenInterface=interface
21         except:
22             print( "eror reading interface")
23             print( "chosen interface is "+str(chosenInterface))
24     return chosenInterface
25 interface=chooseInterface()
26 ni.ifaddresses(interface)

```

```

27 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
28 print(ni.gateways())
29 gateway=''
30 for key,val in (ni.gateways()).items():
31     if isinstance(val,list):
32         for el in val:
33             if interface in el:
34                 print("gateway is "+el[0])
35                 gateway=el[0]
36     else:
37         if interface in val:
38             print("gateway is"+val[0])
39             gateway=val[0]
40 print("gateway is "+gateway)
41 #exit()
42 print(ip) # should print "192.168.100.37"
43 packetlog=dict();
44 userlog=dict();
45 storedlog=dict();
46 templog=dict();
47 fromlist=[]
48 tolist=[]
49 iplist=[gateway]
50 import glob
51 for filename in glob.glob('arpfile*'):
52     print(filename)
53     #if ip in filename: continue;
54
55     try:
56         with open(filename,'r')as f:
57             templog= json.load(f)
58             print((str(templog)))
59             for key,val in list(templog.items()):
60                 print(str(key)+" "+str(val))
61                 if key != ip: storedlog[key]=val
62                 elif key==ip and filename[filename.find('e')+1:filename.find('.')]==ip:
63                     storedlog[key]=val
64
65             '''
66             for i in val:
67                 response = os.system("ping -c 1 " +i)
68                 print(response)
69                 if response ==0:
70                     print("hostup "+i)
71                     if i not in storedlog[ip]: storedlog[ip].append(i)
72                     if i not in iplist: iplist.append(i)
73                     #iplist.append(di['IP address'])
74                     fromlist.append(key)
75                     tolist.append(i)
76             '''
77         except:
78             print("error")
79             continue
80
81 with open("fullarp"+ip+".json",'w') as outfile:
82     print(json.dump(storedlog,outfile))
83 arplog=dict()
84 with open("fullarp"+ip+".json",'r')as f:
85     arplog= json.load(f)

```

```

86 print(arplog)
87 import os
88 '''
89 for key,val in list(arplog.items()):
90     response = os.system("ping -c 1 " + key)
91     print(response)
92     if response ==0: print("hostup")
93     else: print("error")
94     for el in list(val):
95         response = os.system("ping -c 1 " + el)
96         print(response)
97         if response ==0: print("hostup "+str(el))
98         else: print("error")
99 print(ARPTABLE)
100 arpdict=ARPTABLE
101 print(arpdict)
102 iplist=[gateway]
103 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
104 #iplist.append(ip)
105 for di in arpdict:
106     print(di)
107     if '172.27.0' in di['IP address'] and di['IP address'] not in iplist and di['HW address'] !=
        '00:00:00:00:00:00':
108         iplist.append(di['IP address'])
109 '''
110 print(iplist)

```

B.10 protocolplot.py

```

1 import pandas as pd
2 import json
3 import math
4 import json
5 import math
6 import netifaces as ni
7 import matplotlib
8 matplotlib.use('Agg')
9 def chooseInterface():
10     import netifaces as ni
11     chosenInterface='wlp3s0'
12     for interface in ni.interfaces():
13         print(( str(interface)))
14         try:
15             ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
16             if '172.27.0.' in ip:
17                 chosenInterface=interface
18         except:
19             print( "error reading interface")
20             print(("chosen interface is "+str(chosenInterface)))
21     return chosenInterface
22
23 interface=chooseInterface()
24 ni.ifaddresses(interface)
25 ip = ni.ifaddresses(interface)[ni.AF_INET][0]['addr']
26
27 data=dict()
28 with open('iplog'+ip+'.json','r')as f:
29     data= json.load(f)

```

```

30 print(str(data))
31 datalist=[]
32 for key,val in data.items():
33     try:
34         pdata=dict()
35         print(str(key)+" "+str(val['count']))
36         for key1,val1 in val['protocol'].items():
37             pdata['ipprot']=key+key1
38             pdata['count']=val1
39             datalist.append(pdata)
40     except:
41         continue
42
43
44 print(str(datalist))
45 df = pd.DataFrame(datalist)
46 print (df)
47 plot1=df.plot(kind='bar',x='ipprot', y='count',figsize=(100,20))
48 fig=plot1.get_figure()
49 fig.savefig("protgriplog"+ip+".png")
50 '''
51 print (df)
52 plot1=df.plot()
53 fig=plot1.get_figure()
54 fig.savefig("output.png")
55 '''

```

B.11 Frontend Angular Build

```

1 (window["webpackJsonp"] = window["webpackJsonp"] || []).push([[["main"],{
2
3  /**/ "./node_modules/moment/locale sync recursive ^\\.\\.\\.\\.\\.*$":
4  /*!*****!*\
5  !*** ./node_modules/moment/locale sync ^\\.\\.\\.\\.\\.*$ ***/
6  \*****/
7  /*! no static exports found */
8  /**/ (function(module, exports, __webpack_require__) {
9
10 var map = {
11   "./af": "./node_modules/moment/locale/af.js",
12   "./af.js": "./node_modules/moment/locale/af.js",
13   "./ar": "./node_modules/moment/locale/ar.js",
14   "./ar-dz": "./node_modules/moment/locale/ar-dz.js",
15   "./ar-dz.js": "./node_modules/moment/locale/ar-dz.js",
16   "./ar-kw": "./node_modules/moment/locale/ar-kw.js",
17   "./ar-kw.js": "./node_modules/moment/locale/ar-kw.js",
18   "./ar-ly": "./node_modules/moment/locale/ar-ly.js",
19   "./ar-ly.js": "./node_modules/moment/locale/ar-ly.js",
20   "./ar-ma": "./node_modules/moment/locale/ar-ma.js",
21   "./ar-ma.js": "./node_modules/moment/locale/ar-ma.js",
22   "./ar-sa": "./node_modules/moment/locale/ar-sa.js",
23   "./ar-sa.js": "./node_modules/moment/locale/ar-sa.js",
24   "./ar-tn": "./node_modules/moment/locale/ar-tn.js",
25   "./ar-tn.js": "./node_modules/moment/locale/ar-tn.js",
26   "./ar.js": "./node_modules/moment/locale/ar.js",
27   "./az": "./node_modules/moment/locale/az.js",
28   "./az.js": "./node_modules/moment/locale/az.js",
29   "./be": "./node_modules/moment/locale/be.js",

```

30 ".be.js": "./node_modules/moment/locale/be.js",
31 ".bg": "./node_modules/moment/locale/bg.js",
32 ".bg.js": "./node_modules/moment/locale/bg.js",
33 ".bm": "./node_modules/moment/locale/bm.js",
34 ".bm.js": "./node_modules/moment/locale/bm.js",
35 ".bn": "./node_modules/moment/locale/bn.js",
36 ".bn.js": "./node_modules/moment/locale/bn.js",
37 ".bo": "./node_modules/moment/locale/bo.js",
38 ".bo.js": "./node_modules/moment/locale/bo.js",
39 ".br": "./node_modules/moment/locale/br.js",
40 ".br.js": "./node_modules/moment/locale/br.js",
41 ".bs": "./node_modules/moment/locale/bs.js",
42 ".bs.js": "./node_modules/moment/locale/bs.js",
43 ".ca": "./node_modules/moment/locale/ca.js",
44 ".ca.js": "./node_modules/moment/locale/ca.js",
45 ".cs": "./node_modules/moment/locale/cs.js",
46 ".cs.js": "./node_modules/moment/locale/cs.js",
47 ".cv": "./node_modules/moment/locale/cv.js",
48 ".cv.js": "./node_modules/moment/locale/cv.js",
49 ".cy": "./node_modules/moment/locale/cy.js",
50 ".cy.js": "./node_modules/moment/locale/cy.js",
51 ".da": "./node_modules/moment/locale/da.js",
52 ".da.js": "./node_modules/moment/locale/da.js",
53 ".de": "./node_modules/moment/locale/de.js",
54 ".de-at": "./node_modules/moment/locale/de-at.js",
55 ".de-at.js": "./node_modules/moment/locale/de-at.js",
56 ".de-ch": "./node_modules/moment/locale/de-ch.js",
57 ".de-ch.js": "./node_modules/moment/locale/de-ch.js",
58 ".de.js": "./node_modules/moment/locale/de.js",
59 ".dv": "./node_modules/moment/locale/dv.js",
60 ".dv.js": "./node_modules/moment/locale/dv.js",
61 ".el": "./node_modules/moment/locale/el.js",
62 ".el.js": "./node_modules/moment/locale/el.js",
63 ".en-SG": "./node_modules/moment/locale/en-SG.js",
64 ".en-SG.js": "./node_modules/moment/locale/en-SG.js",
65 ".en-au": "./node_modules/moment/locale/en-au.js",
66 ".en-au.js": "./node_modules/moment/locale/en-au.js",
67 ".en-ca": "./node_modules/moment/locale/en-ca.js",
68 ".en-ca.js": "./node_modules/moment/locale/en-ca.js",
69 ".en-gb": "./node_modules/moment/locale/en-gb.js",
70 ".en-gb.js": "./node_modules/moment/locale/en-gb.js",
71 ".en-ie": "./node_modules/moment/locale/en-ie.js",
72 ".en-ie.js": "./node_modules/moment/locale/en-ie.js",
73 ".en-il": "./node_modules/moment/locale/en-il.js",
74 ".en-il.js": "./node_modules/moment/locale/en-il.js",
75 ".en-nz": "./node_modules/moment/locale/en-nz.js",
76 ".en-nz.js": "./node_modules/moment/locale/en-nz.js",
77 ".eo": "./node_modules/moment/locale/eo.js",
78 ".eo.js": "./node_modules/moment/locale/eo.js",
79 ".es": "./node_modules/moment/locale/es.js",
80 ".es-do": "./node_modules/moment/locale/es-do.js",
81 ".es-do.js": "./node_modules/moment/locale/es-do.js",
82 ".es-us": "./node_modules/moment/locale/es-us.js",
83 ".es-us.js": "./node_modules/moment/locale/es-us.js",
84 ".es.js": "./node_modules/moment/locale/es.js",
85 ".et": "./node_modules/moment/locale/et.js",
86 ".et.js": "./node_modules/moment/locale/et.js",
87 ".eu": "./node_modules/moment/locale/eu.js",
88 ".eu.js": "./node_modules/moment/locale/eu.js",


```

89  "./fa": "./node_modules/moment/locale/fa.js",
90  "./fa.js": "./node_modules/moment/locale/fa.js",
91  "./fi": "./node_modules/moment/locale/fi.js",
92  "./fi.js": "./node_modules/moment/locale/fi.js",
93  "./fo": "./node_modules/moment/locale/fo.js",
94  "./fo.js": "./node_modules/moment/locale/fo.js",
95  "./fr": "./node_modules/moment/locale/fr.js",
96  "./fr-ca": "./node_modules/moment/locale/fr-ca.js",
97  "./fr-ca.js": "./node_modules/moment/locale/fr-ca.js",
98  "./fr-ch": "./node_modules/moment/locale/fr-ch.js",
99  "./fr-ch.js": "./node_modules/moment/locale/fr-ch.js",
100  "./fr.js": "./node_modules/moment/locale/fr.js",
101  "./fy": "./node_modules/moment/locale/fy.js",
102  "./fy.js": "./node_modules/moment/locale/fy.js",
103  "./ga": "./node_modules/moment/locale/ga.js",
104  "./ga.js": "./node_modules/moment/locale/ga.js",
105  "./gd": "./node_modules/moment/locale/gd.js",
106  "./gd.js": "./node_modules/moment/locale/gd.js",
107  "./gl": "./node_modules/moment/locale/gl.js",
108  "./gl.js": "./node_modules/moment/locale/gl.js",
109  "./gom-latn": "./node_modules/moment/locale/gom-latn.js",
110  "./gom-latn.js": "./node_modules/moment/locale/gom-latn.js",
111  "./gu": "./node_modules/moment/locale/gu.js",
112  "./gu.js": "./node_modules/moment/locale/gu.js",
113  "./he": "./node_modules/moment/locale/he.js",
114  "./he.js": "./node_modules/moment/locale/he.js",
115  "./hi": "./node_modules/moment/locale/hi.js",
116  "./hi.js": "./node_modules/moment/locale/hi.js",
117  "./hr": "./node_modules/moment/locale/hr.js",
118  "./hr.js": "./node_modules/moment/locale/hr.js",
119  "./hu": "./node_modules/moment/locale/hu.js",
120  "./hu.js": "./node_modules/moment/locale/hu.js",
121  "./hy-am": "./node_modules/moment/locale/hy-am.js",
122  "./hy-am.js": "./node_modules/moment/locale/hy-am.js",
123  "./id": "./node_modules/moment/locale/id.js",
124  "./id.js": "./node_modules/moment/locale/id.js",
125  "./is": "./node_modules/moment/locale/is.js",
126  "./is.js": "./node_modules/moment/locale/is.js",
127  "./it": "./node_modules/moment/locale/it.js",
128  "./it-ch": "./node_modules/moment/locale/it-ch.js",
129  "./it-ch.js": "./node_modules/moment/locale/it-ch.js",
130  "./it.js": "./node_modules/moment/locale/it.js",
131  "./ja": "./node_modules/moment/locale/ja.js",
132  "./ja.js": "./node_modules/moment/locale/ja.js",
133  "./jv": "./node_modules/moment/locale/jv.js",
134  "./jv.js": "./node_modules/moment/locale/jv.js",
135  "./ka": "./node_modules/moment/locale/ka.js",
136  "./ka.js": "./node_modules/moment/locale/ka.js",
137  "./kk": "./node_modules/moment/locale/kk.js",
138  "./kk.js": "./node_modules/moment/locale/kk.js",
139  "./km": "./node_modules/moment/locale/km.js",
140  "./km.js": "./node_modules/moment/locale/km.js",
141  "./kn": "./node_modules/moment/locale/kn.js",
142  "./kn.js": "./node_modules/moment/locale/kn.js",
143  "./ko": "./node_modules/moment/locale/ko.js",
144  "./ko.js": "./node_modules/moment/locale/ko.js",
145  "./ku": "./node_modules/moment/locale/ku.js",
146  "./ku.js": "./node_modules/moment/locale/ku.js",
147  "./ky": "./node_modules/moment/locale/ky.js",

```

```

148  "./ky.js": "./node_modules/moment/locale/ky.js",
149  "./lb": "./node_modules/moment/locale/lb.js",
150  "./lb.js": "./node_modules/moment/locale/lb.js",
151  "./lo": "./node_modules/moment/locale/lo.js",
152  "./lo.js": "./node_modules/moment/locale/lo.js",
153  "./lt": "./node_modules/moment/locale/lt.js",
154  "./lt.js": "./node_modules/moment/locale/lt.js",
155  "./lv": "./node_modules/moment/locale/lv.js",
156  "./lv.js": "./node_modules/moment/locale/lv.js",
157  "./me": "./node_modules/moment/locale/me.js",
158  "./me.js": "./node_modules/moment/locale/me.js",
159  "./mi": "./node_modules/moment/locale/mi.js",
160  "./mi.js": "./node_modules/moment/locale/mi.js",
161  "./mk": "./node_modules/moment/locale/mk.js",
162  "./mk.js": "./node_modules/moment/locale/mk.js",
163  "./ml": "./node_modules/moment/locale/ml.js",
164  "./ml.js": "./node_modules/moment/locale/ml.js",
165  "./mn": "./node_modules/moment/locale/mn.js",
166  "./mn.js": "./node_modules/moment/locale/mn.js",
167  "./mr": "./node_modules/moment/locale/mr.js",
168  "./mr.js": "./node_modules/moment/locale/mr.js",
169  "./ms": "./node_modules/moment/locale/ms.js",
170  "./ms-my": "./node_modules/moment/locale/ms-my.js",
171  "./ms-my.js": "./node_modules/moment/locale/ms-my.js",
172  "./ms.js": "./node_modules/moment/locale/ms.js",
173  "./mt": "./node_modules/moment/locale/mt.js",
174  "./mt.js": "./node_modules/moment/locale/mt.js",
175  "./my": "./node_modules/moment/locale/my.js",
176  "./my.js": "./node_modules/moment/locale/my.js",
177  "./nb": "./node_modules/moment/locale/nb.js",
178  "./nb.js": "./node_modules/moment/locale/nb.js",
179  "./ne": "./node_modules/moment/locale/ne.js",
180  "./ne.js": "./node_modules/moment/locale/ne.js",
181  "./nl": "./node_modules/moment/locale/nl.js",
182  "./nl-be": "./node_modules/moment/locale/nl-be.js",
183  "./nl-be.js": "./node_modules/moment/locale/nl-be.js",
184  "./nl.js": "./node_modules/moment/locale/nl.js",
185  "./nn": "./node_modules/moment/locale/nn.js",
186  "./nn.js": "./node_modules/moment/locale/nn.js",
187  "./pa-in": "./node_modules/moment/locale/pa-in.js",
188  "./pa-in.js": "./node_modules/moment/locale/pa-in.js",
189  "./pl": "./node_modules/moment/locale/pl.js",
190  "./pl.js": "./node_modules/moment/locale/pl.js",
191  "./pt": "./node_modules/moment/locale/pt.js",
192  "./pt-br": "./node_modules/moment/locale/pt-br.js",
193  "./pt-br.js": "./node_modules/moment/locale/pt-br.js",
194  "./pt.js": "./node_modules/moment/locale/pt.js",
195  "./ro": "./node_modules/moment/locale/ro.js",
196  "./ro.js": "./node_modules/moment/locale/ro.js",
197  "./ru": "./node_modules/moment/locale/ru.js",
198  "./ru.js": "./node_modules/moment/locale/ru.js",
199  "./sd": "./node_modules/moment/locale/sd.js",
200  "./sd.js": "./node_modules/moment/locale/sd.js",
201  "./se": "./node_modules/moment/locale/se.js",
202  "./se.js": "./node_modules/moment/locale/se.js",
203  "./si": "./node_modules/moment/locale/si.js",
204  "./si.js": "./node_modules/moment/locale/si.js",
205  "./sk": "./node_modules/moment/locale/sk.js",
206  "./sk.js": "./node_modules/moment/locale/sk.js",

```

```

207  "./sl": "./node_modules/moment/locale/sl.js",
208  "./sl.js": "./node_modules/moment/locale/sl.js",
209  "./sq": "./node_modules/moment/locale/sq.js",
210  "./sq.js": "./node_modules/moment/locale/sq.js",
211  "./sr": "./node_modules/moment/locale/sr.js",
212  "./sr-cyrl": "./node_modules/moment/locale/sr-cyrl.js",
213  "./sr-cyrl.js": "./node_modules/moment/locale/sr-cyrl.js",
214  "./sr.js": "./node_modules/moment/locale/sr.js",
215  "./ss": "./node_modules/moment/locale/ss.js",
216  "./ss.js": "./node_modules/moment/locale/ss.js",
217  "./sv": "./node_modules/moment/locale/sv.js",
218  "./sv.js": "./node_modules/moment/locale/sv.js",
219  "./sw": "./node_modules/moment/locale/sw.js",
220  "./sw.js": "./node_modules/moment/locale/sw.js",
221  "./ta": "./node_modules/moment/locale/ta.js",
222  "./ta.js": "./node_modules/moment/locale/ta.js",
223  "./te": "./node_modules/moment/locale/te.js",
224  "./te.js": "./node_modules/moment/locale/te.js",
225  "./tet": "./node_modules/moment/locale/tet.js",
226  "./tet.js": "./node_modules/moment/locale/tet.js",
227  "./tg": "./node_modules/moment/locale/tg.js",
228  "./tg.js": "./node_modules/moment/locale/tg.js",
229  "./th": "./node_modules/moment/locale/th.js",
230  "./th.js": "./node_modules/moment/locale/th.js",
231  "./tl-ph": "./node_modules/moment/locale/tl-ph.js",
232  "./tl-ph.js": "./node_modules/moment/locale/tl-ph.js",
233  "./tlh": "./node_modules/moment/locale/tlh.js",
234  "./tlh.js": "./node_modules/moment/locale/tlh.js",
235  "./tr": "./node_modules/moment/locale/tr.js",
236  "./tr.js": "./node_modules/moment/locale/tr.js",
237  "./tzm": "./node_modules/moment/locale/tzm.js",
238  "./tzm.js": "./node_modules/moment/locale/tzm.js",
239  "./tzm-latn": "./node_modules/moment/locale/tzm-latn.js",
240  "./tzm-latn.js": "./node_modules/moment/locale/tzm-latn.js",
241  "./tzm.js": "./node_modules/moment/locale/tzm.js",
242  "./ug-cn": "./node_modules/moment/locale/ug-cn.js",
243  "./ug-cn.js": "./node_modules/moment/locale/ug-cn.js",
244  "./uk": "./node_modules/moment/locale/uk.js",
245  "./uk.js": "./node_modules/moment/locale/uk.js",
246  "./ur": "./node_modules/moment/locale/ur.js",
247  "./ur.js": "./node_modules/moment/locale/ur.js",
248  "./uz": "./node_modules/moment/locale/uz.js",
249  "./uz-latn": "./node_modules/moment/locale/uz-latn.js",
250  "./uz-latn.js": "./node_modules/moment/locale/uz-latn.js",
251  "./uz.js": "./node_modules/moment/locale/uz.js",
252  "./vi": "./node_modules/moment/locale/vi.js",
253  "./vi.js": "./node_modules/moment/locale/vi.js",
254  "./x-pseudo": "./node_modules/moment/locale/x-pseudo.js",
255  "./x-pseudo.js": "./node_modules/moment/locale/x-pseudo.js",
256  "./yo": "./node_modules/moment/locale/yo.js",
257  "./yo.js": "./node_modules/moment/locale/yo.js",
258  "./zh-cn": "./node_modules/moment/locale/zh-cn.js",
259  "./zh-cn.js": "./node_modules/moment/locale/zh-cn.js",
260  "./zh-hk": "./node_modules/moment/locale/zh-hk.js",
261  "./zh-hk.js": "./node_modules/moment/locale/zh-hk.js",
262  "./zh-tw": "./node_modules/moment/locale/zh-tw.js",
263  "./zh-tw.js": "./node_modules/moment/locale/zh-tw.js",
264  };
265

```

```

266
267
268 function webpackContext(req) {
269     var id = webpackContextResolve(req);
270     return __webpack_require__(id);
271 }
272 function webpackContextResolve(req) {
273     var id = map[req];
274     if(!(id + 1)) { // check for number or string
275         var e = new Error("Cannot find module '" + req + "'");
276         e.code = 'MODULE_NOT_FOUND';
277         throw e;
278     }
279     return id;
280 }
281 webpackContext.keys = function webpackContextKeys() {
282     return Object.keys(map);
283 };
284 webpackContext.resolve = webpackContextResolve;
285 module.exports = webpackContext;
286 webpackContext.id = "./node_modules/moment/locale sync recursive ^\\.\\.\\./.*$";
287
288 /**/ },
289
290 /**/ "./src/$_lazy_route_resource lazy recursive":
291 /*!*****!\
292     !*** ./src/$_lazy_route_resource lazy namespace object ***!
293     \*****/
294 /*! no static exports found */
295 /**/ (function(module, exports) {
296
297     function webpackEmptyAsyncContext(req) {
298         // Here Promise.resolve().then() is used instead of new Promise() to prevent
299         // uncaught exception popping up in devtools
300         return Promise.resolve().then(function() {
301             var e = new Error("Cannot find module '" + req + "'");
302             e.code = 'MODULE_NOT_FOUND';
303             throw e;
304         });
305     }
306     webpackEmptyAsyncContext.keys = function() { return []; };
307     webpackEmptyAsyncContext.resolve = webpackEmptyAsyncContext;
308     module.exports = webpackEmptyAsyncContext;
309     webpackEmptyAsyncContext.id = "./src/$_lazy_route_resource lazy recursive";
310
311     /**/ },
312
313     /**/ "./src/app/app-routing.module.ts":
314     /*!*****!\
315         !*** ./src/app/app-routing.module.ts ***!
316         \*****/
317     /*! exports provided: AppRoutingModuleModule */
318     /**/ (function(module, __webpack_exports__, __webpack_require__) {
319
320         "use strict";
321         __webpack_require__.r(__webpack_exports__);
322         /* harmony export (binding) */ __webpack_require__.d(__webpack_exports__, "AppRoutingModule",
            function() { return AppRoutingModuleModule; });
323         /* harmony import */ var _angular_core__WEBPACK_IMPORTED_MODULE_0__ = __webpack_require__(/*!

```

```

    @angular/core */ "./node_modules/@angular/core/fesm5/core.js");
324 /* harmony import */ var _angular_router__WEBPACK_IMPORTED_MODULE_1__ = __webpack_require__(/*!
    @angular/router */ "./node_modules/@angular/router/fesm5/router.js");
325 /* harmony import */ var _home_home_component__WEBPACK_IMPORTED_MODULE_2__ = __webpack_require
    __(/*! ./home/home.component */ "./src/app/home/home.component.ts");
326 var __decorate = (undefined && undefined.__decorate) || function (decorators, target, key, desc) {
327     var c = arguments.length, r = c < 3 ? target : desc === null ? desc = Object.
        getOwnPropertyDescriptor(target, key) : desc, d;
328     if (typeof Reflect === "object" && typeof Reflect.decorate === "function") r = Reflect.decorate
        (decorators, target, key, desc);
329     else for (var i = decorators.length - 1; i >= 0; i--) if (d = decorators[i]) r = (c < 3 ? d(r)
        : c > 3 ? d(target, key, r) : d(target, key)) || r;
330     return c > 3 && r && Object.defineProperty(target, key, r), r;
331 };
332
333
334
335 var routes = [
336     { path: '', component: _home_home_component__WEBPACK_IMPORTED_MODULE_2__["HomeComponent"] },
337 ];
338 var AppRoutingModuleModule = /** @class */ (function () {
339     function AppRoutingModuleModule() {
340     }
341     AppRoutingModuleModule = __decorate([
342         Object(_angular_core__WEBPACK_IMPORTED_MODULE_0__["NgModule"])({
343             imports: [
344                 _angular_router__WEBPACK_IMPORTED_MODULE_1__["RouterModule"].forRoot(routes)
345             ],
346             exports: [
347                 _angular_router__WEBPACK_IMPORTED_MODULE_1__["RouterModule"]
348             ]
349         })
350     ], AppRoutingModuleModule);
351     return AppRoutingModuleModule;
352 }());
353
354
355
356 /****/ }),
357
358 /****/ "./src/app/app.component.css":
359 /*!*****!*\
360     !*** ./src/app/app.component.css ***!
361     \******/
362 /*! no static exports found */
363 /****/ (function(module, exports) {
364
365     module.exports = ""
366
367 /****/ }),
368
369 /****/ "./src/app/app.component.html":
370 /*!*****!*\
371     !*** ./src/app/app.component.html ***!
372     \******/
373 /*! no static exports found */
374 /****/ (function(module, exports) {
375
376     module.exports = "<router-outlet></router-outlet>\n\n"

```

```

377
378 /****/ }),
379
380 /****/ "./src/app/app.component.ts":
381 /*!*****!*\
382 !*** ./src/app/app.component.ts ***!
383 \*****/
384 /*! exports provided: AppComponent */
385 /****/ (function(module, __webpack_exports__, __webpack_require__) {
386
387 "use strict";
388 __webpack_require__.r(__webpack_exports__);
389 /* harmony export (binding) */ __webpack_require__.d(__webpack_exports__, "AppComponent", function
    () { return AppComponent; });
390 /* harmony import */ var _angular_core__WEBPACK_IMPORTED_MODULE_0__ = __webpack_require__(/*!
    @angular/core */ "./node_modules/@angular/core/fesm5/core.js");
391 var __decorate = (undefined && undefined.__decorate) || function (decorators, target, key, desc) {
392     var c = arguments.length, r = c < 3 ? target : desc === null ? desc = Object.
        getOwnPropertyDescriptor(target, key) : desc, d;
393     if (typeof Reflect === "object" && typeof Reflect.decorate === "function") r = Reflect.decorate
        (decorators, target, key, desc);
394     else for (var i = decorators.length - 1; i >= 0; i--) if (d = decorators[i]) r = (c < 3 ? d(r)
        : c > 3 ? d(target, key, r) : d(target, key)) || r;
395     return c > 3 && r && Object.defineProperty(target, key, r), r;
396 };
397
398 var AppComponent = /** @class */ (function () {
399     function AppComponent() {
400         this.title = 'meshFrontend';
401     }
402     AppComponent = __decorate([
403         Object(_angular_core__WEBPACK_IMPORTED_MODULE_0__["Component"])({
404             selector: 'app-root',
405             template: __webpack_require__(/*! ./app.component.html */ "./src/app/app.component.html")
406             ,
407             styles: [__webpack_require__(/*! ./app.component.css */ "./src/app/app.component.css")]
408         }
409         ), AppComponent);
410     return AppComponent;
411 }());
412
413 /****/ }),
414
415 /****/ "./src/app/app.module.ts":
416 /*!*****!*\
417 !*** ./src/app/app.module.ts ***!
418 \*****/
419 /*! exports provided: AppModule */
420 /****/ (function(module, __webpack_exports__, __webpack_require__) {
421
422 "use strict";
423 __webpack_require__.r(__webpack_exports__);
424 /* harmony export (binding) */ __webpack_require__.d(__webpack_exports__, "AppModule", function() {
    return AppModule; });
425 /* harmony import */ var _angular_platform_browser__WEBPACK_IMPORTED_MODULE_0__ = __webpack_require
    (/*! @angular/platform-browser */ "./node_modules/@angular/platform-browser/fesm5/platform-
    browser.js");

```

```

427 /* harmony import */ var _angular_core__WEBPACK_IMPORTED_MODULE_1__ = __webpack_require__(/*!
    @angular/core */ "./node_modules/@angular/core/fesm5/core.js");
428 /* harmony import */ var _angular_forms__WEBPACK_IMPORTED_MODULE_2__ = __webpack_require__(/*!
    @angular/forms */ "./node_modules/@angular/forms/fesm5/forms.js");
429 /* harmony import */ var _app_component__WEBPACK_IMPORTED_MODULE_3__ = __webpack_require__(/*! ./
    app.component */ "./src/app/app.component.ts");
430 /* harmony import */ var _home_home_component__WEBPACK_IMPORTED_MODULE_4__ = __webpack_require
    __(/*! ./home/home.component */ "./src/app/home/home.component.ts");
431 /* harmony import */ var _app_routing_module__WEBPACK_IMPORTED_MODULE_5__ = __webpack_require__(/*!
    ./app-routing.module */ "./src/app/app-routing.module.ts");
432 /* harmony import */ var _angular_common_http__WEBPACK_IMPORTED_MODULE_6__ = __webpack_require
    __(/*! @angular/common/http */ "./node_modules/@angular/common/fesm5/http.js");
433 /* harmony import */ var _swimlane_ngx_graph__WEBPACK_IMPORTED_MODULE_7__ = __webpack_require__(/*!
    @swimlane/ngx-graph */ "./node_modules/@swimlane/ngx-graph/release/index.js");
434 /* harmony import */ var _swimlane_ngx_graph__WEBPACK_IMPORTED_MODULE_7___default = /*#__PURE__*/__
    webpack_require__._n(_swimlane_ngx_graph__WEBPACK_IMPORTED_MODULE_7__);
435 /* harmony import */ var _angular_platform_browser_animations__WEBPACK_IMPORTED_MODULE_8__ = __
    webpack_require__(/*! @angular/platform-browser/animations */ "./node_modules/@angular/platform
    -browser/fesm5/animations.js");
436 var __decorate = (undefined && undefined.__decorate) || function (decorators, target, key, desc) {
437   var c = arguments.length, r = c < 3 ? target : desc === null ? desc = Object.
    getOwnPropertyDescriptor(target, key) : desc, d;
438   if (typeof Reflect === "object" && typeof Reflect.decorate === "function") r = Reflect.decorate
    (decorators, target, key, desc);
439   else for (var i = decorators.length - 1; i >= 0; i--) if (d = decorators[i]) r = (c < 3 ? d(r)
    : c > 3 ? d(target, key, r) : d(target, key)) || r;
440   return c > 3 && r && Object.defineProperty(target, key, r), r;
441 };
442
443
444
445
446
447
448
449
450
451 var AppModule = /** @class */ (function () {
452   function AppModule() {
453   }
454   AppModule = __decorate([
455     Object(_angular_core__WEBPACK_IMPORTED_MODULE_1__["NgModule"])(
456       {
457         declarations: [
458           _app_component__WEBPACK_IMPORTED_MODULE_3__["AppComponent"],
459           _home_home_component__WEBPACK_IMPORTED_MODULE_4__["HomeComponent"]
460         ],
461         imports: [
462           _angular_platform_browser__WEBPACK_IMPORTED_MODULE_0__["BrowserModule"],
463           _app_routing_module__WEBPACK_IMPORTED_MODULE_5__["AppRoutingModule"],
464           _angular_common_http__WEBPACK_IMPORTED_MODULE_6__["HttpClientModule"],
465           _swimlane_ngx_graph__WEBPACK_IMPORTED_MODULE_7__["NgxGraphModule"],
466           _angular_platform_browser_animations__WEBPACK_IMPORTED_MODULE_8__["
467             BrowserModuleAnimationsModule"],
468           _angular_forms__WEBPACK_IMPORTED_MODULE_2__["FormsModule"]
469         ],
470         schemas: [_angular_core__WEBPACK_IMPORTED_MODULE_1__["CUSTOM_ELEMENTS_SCHEMA"]],
471         providers: [],
472         bootstrap: [_app_component__WEBPACK_IMPORTED_MODULE_3__["AppComponent"]]
473       }
474     )
475   ])(AppModule);

```

```

472     ], AppModule);
473     return AppModule;
474 }());
475
476
477
478 /**/ }},
479
480 /**/ "./src/app/home/home.component.css":
481 /*!*****!*\
482 !*** ./src/app/home/home.component.css ***!
483 \*****/
484 /*! no static exports found */
485 /**/ (function(module, exports) {
486
487 module.exports = ".graph-container {\n\theight: 700px; \n\twidth: 70%;\n\tmargin: 0px auto;\n}\n\n.
    node:hover {\n\tcursor:pointer;\n}\n\n.selectedTab {\n\tbackground-color: #A3ADC4;\n}\n\n.
    navButtons {\n\tmargin:0px auto;\n\ttext-align: center;\n\tmargin-top: 25px;\n\tmargin-bottom:
    25px;\n}\n\n.navButton {\n\tmargin-left: 5px;\n\tmargin-right: 5px;\n\tfont-size: 30px;\n\t
    tborder-radius: 5px;\n\tbox-shadow: 2px 2px 2px 2px #888888;\n}\n\n.navButton:hover {\n\tcursor
    : pointer;\n}\n\n.navButton:focus {\n\toutline: none;\n}\n\n#mynetwork {\n\tborder: 5px solid
    #05B2DC;\n\tborder-radius: 20px;\n\tbackground: #dddddd;\n\tdisplay: inline-block;\n\theight: 500px
    ;\n\twidth: 70%;\n\tbox-shadow: 5px 5px 5px 5px #888888;\n}\n\n.exit {\n\tfloat: right;\n}\n\n.
    network {\n\ttext-align: center;\n}\n\n.bottomBut {\n\ttext-align: center;\n\tdisplay: block;\n
    \tmargin:0px auto;\n\tfont-size: 24px;\n}\n\n.view-data-modal {\n\tposition: absolute;\n\ttop:
    50%;\n\tleft: 50%; \n\t-webkit-transform: translate(-50%, -50%); \n\ttransform: translate
    (-50%, -50%);\n\tbackground-color: #efefef;\n\ttext-align: center;\n\twidth: 90%;\n\theight:
    80%;\n\tborder: 5px solid #05B2DC;\n\tborder-radius: 5px;\n}\n\n.view-data-title {\n\tfont-size
    : 45px;\n\tcolor: #0270B1;\n\tmargin-top: 4px;\n\tmargin-bottom: 4px;\n}\n\n.dataTab {\n\t
    tbackground-color: #efefef;\n}\n\n.barChart {\n\theight: 80%;\n\twidth: 45%;\n\tdisplay: inline
    -block;\n}\n\n.piChart {\n\theight: 80%;\n\twidth: 45%;\n\tdisplay: inline-block;\n}\n\n.
    chartContainer {\n\twidth: 50%;\n\tdisplay: inline-block;\n\tvertical-align: top;\n}\n\n.
    tcChartContainer {\n\twidth: 50%;\n\tdisplay: inline-block;\n\tvertical-align: top;\n}\n\n.
    tcTab {\n\ttext-align: center;\n}\n\n.inputContainer {\n\tdisplay: block;\n\twidth: 100%;\n\t
    tmargin: 0px auto;\n\ttext-align: center;\n\tmargin-top: 20px;\n\tmargin-bottom: 20px;\n\tfont-
    size: 20px;\n}\n\n.lists {\n\twidth: 70%;\n\tmargin: 0px auto;\n}\n\n.ipList {\n\tdisplay:
    inline-block;\n\tvertical-align: top;\n\tmargin-left: 20px;\n\tmargin-right: 20px;\n\tborder: 4
    px solid #05B2DC;\n\twidth: 40%;\n\tbackground-color: #dddddd;\n\tborder-radius: 10px;\n}\n\n.
    listTitle {\n\tcolor: #5B6272;\n}"
488
489 /**/ }},
490
491 /**/ "./src/app/home/home.component.html":
492 /*!*****!*\
493 !*** ./src/app/home/home.component.html ***!
494 \*****/
495 /*! no static exports found */
496 /**/ (function(module, exports) {
497
498 module.exports = "<!-- <button (click)=\"test()\">Test</button> -->\n\n<div class=\"navButtons\">\n\n
    \t<button [ngClass]=\"{'selectedTab': currentTab == 'network'}\" class=\"navButton\" (click)=\"
    currentTab='network'; closeModal()\">Network</button>\n\n\t<button [ngClass]=\"{'selectedTab':
    currentTab == 'data'}\" class=\"navButton\" (click)=\"currentTab='data'\">Data</button>\n\n\t<
    button [ngClass]=\"{'selectedTab': currentTab == 'tc'}\" class=\"navButton\" (click)=\"
    currentTab='tc'\">Traffic Control</button>\n\n</div>\n\n<div [hidden]=\"network\"
    class=\"network\">\n\n\t<div #graphContainer id=\"mynetwork\" class=\"graph-container\"></div>\n
    </div>\n\n<div [hidden]=\"!viewingIp || currentTab != 'network'\" class=\"view-data-modal\"> \n
    \t<div class=\"exit\">\n\n\t\t<button (click)=\"closeModal()\">X</button>\n\n\t</div>\n\n\t<h2 class
    =\"view-data-title\">\n\n\t\t{viewingIp}\n\n\t</h2>\n\n\t<div class=\"barChart\">\n\n\t\t<canvas id
  
```



```

=>\trafficChart\>{{packetChart}}</canvas>\n\t</div>\n\t<div class=\"piChart\">\n\t\t<canvas id
=\"protChart\">{{protChart}}</canvas>\n\t</div>\n\t<button class=\"bottomBut\" (click)=\"
closeModal()\>Close</button>\n\t</div>\n\t<div [hidden]=\"currentTab != 'data'\" class=\"dataTab
\">\n\t<div class=\"chartContainer\">\n\t\t<canvas id=\"traffic\">{{testChart}}</canvas>\n\t\t<
canvas id=\"length\">{{lengthChart}}</canvas>\n\t</div>\n\t<div class=\"piChartContainer\">\n\t
\t<canvas id=\"protocols\">{{totalProtChart}}</canvas>\n\t\t<canvas id=\"externals\">{{
externalChart}}</canvas>\n\t</div>\n\t</div>\n\t<div class=\"tcTab\" [hidden]=\"currentTab != 'tc
'\">\n\t\t<div class=\"lists\">\n\t\t\t<div class=\"ipList\">\n\t\t\t\t<div class=\"inputContainer
\">\n\t\t\t\t\t<label>Throttle IP:</label>\n\t\t\t\t\t<input [(ngModel)]=\"throttleInput\" name=\"
ti\">\n\t\t\t\t\t<button (click)=\"throttleIP()\">Submit</button>\n\t\t\t\t\t</div>\n\t\t\t\t\t<h1 class
=\"listTitle\">Throttled IPs</h1>\n\t\t\t\t\t<p *ngFor=\"let ip of slowedIps\">{{ip}}</p>\n\t\t\t\t</
div>\n\t\t\t\t<div class=\"ipList\">\n\t\t\t\t\t<div class=\"inputContainer\">\n\t\t\t\t\t\t<label>
Prioritize IP:</label>\n\t\t\t\t\t\t<input [(ngModel)]=\"prioritizeInput\" name=\"pi\">\n\t\t\t\t\t\t<
button (click)=\"prioritizeIP()\">Submit</button>\n\t\t\t\t\t\t</div>\n\t\t\t\t\t\t<h1 class=\"listTitle\">
Prioritized IPs</h1>\n\t\t\t\t\t\t<p *ngFor=\"let ip of priorityIps\">{{ip}}</p>\n\t\t\t\t</div>\n\t</div>
>\n\t</div>

```

```

499
500 /**/ }},
501
502 /**/ "./src/app/home/home.component.ts":
503 /*!*****!*\
504 !*** ./src/app/home/home.component.ts ***!
505 \*****/
506 /*! exports provided: HomeComponent */
507 /**/ (function(module, __webpack_exports__, __webpack_require__) {
508
509 "use strict";
510 __webpack_require__.r(__webpack_exports__);
511 /* harmony export (binding) */ __webpack_require__.d(__webpack_exports__, "HomeComponent", function
() { return HomeComponent; });
512 /* harmony import */ var _angular_core__WEBPACK_IMPORTED_MODULE_0__ = __webpack_require__(/*!
@angular/core */ "./node_modules/@angular/core/fesm5/core.js");
513 /* harmony import */ var _mesh_service__WEBPACK_IMPORTED_MODULE_1__ = __webpack_require__(/*! ../
mesh.service */ "./src/app/mesh.service.ts");
514 /* harmony import */ var chart_js__WEBPACK_IMPORTED_MODULE_2__ = __webpack_require__(/*! chart.js
*/ "./node_modules/chart.js/dist/Chart.js");
515 /* harmony import */ var chart_js__WEBPACK_IMPORTED_MODULE_2___default = /*#__PURE__*/__webpack_
require__.n(chart_js__WEBPACK_IMPORTED_MODULE_2__);
516 /* harmony import */ var vis__WEBPACK_IMPORTED_MODULE_3__ = __webpack_require__(/*! vis */ "./node_
modules/vis/dist/vis.js");
517 /* harmony import */ var vis__WEBPACK_IMPORTED_MODULE_3___default = /*#__PURE__*/__webpack_require
__.n(vis__WEBPACK_IMPORTED_MODULE_3__);
518 var __decorate = (undefined && undefined.__decorate) || function (decorators, target, key, desc) {
519   var c = arguments.length, r = c < 3 ? target : desc === null ? desc = Object.
    getOwnPropertyDescriptor(target, key) : desc, d;
520   if (typeof Reflect === "object" && typeof Reflect.decorate === "function") r = Reflect.decorate
    (decorators, target, key, desc);
521   else for (var i = decorators.length - 1; i >= 0; i--) if (d = decorators[i]) r = (c < 3 ? d(r)
    : c > 3 ? d(target, key, r) : d(target, key)) || r;
522   return c > 3 && r && Object.defineProperty(target, key, r), r;
523 };
524 var __metadata = (undefined && undefined.__metadata) || function (k, v) {
525   if (typeof Reflect === "object" && typeof Reflect.metadata === "function") return Reflect.
    metadata(k, v);
526 };
527
528
529
530

```

```

531  /*
532     Pi IPs:
533     172.27.0.15 (gateway) x
534     172.27.0.32 (access point) x
535     172.27.0.12 (routing)
536     172.27.0.90 (access point)
537     172.27.0.78 (routing) x
538  */
539  var HomeComponent = /** @class */ (function () {
540      function HomeComponent(meshService) {
541          this.meshService = meshService;
542          this.trafficChart = [];
543          this.lengthChart = [];
544          this.totalProtChart = [];
545          this.externalChart = [];
546          this.ips = ["172.27.0.15", "172.27.0.152", "172.27.0.12", "172.27.0.90", "172.27.0.78"];
547          this.runningIps = [];
548          this.IPChart = [];
549          this.packetChart = [];
550          this.protChart = [];
551          //traffic control
552          this.slowedIps = ["172.27.0.68", "172.27.0.178"];
553          this.priorityIps = ["172.27.0.15"];
554          this.graphData = {};
555      }
556      HomeComponent.prototype.ngOnInit = function () {
557          //first, get IPs and Link Files
558          //Second get Packet Data
559          this.currentTab = "network";
560          // this.networkData = {"172.27.0.15": {"172.27.0.90": {"count": 287}, "totalsize": 64440, "
561          // count": 287, "protocol": {"TCP": 240, "DATA-TEXT-LINES": 2, "JSON": 19, "HTTP": 26}, "
562          // length": 224},
563          // "172.27.0.78": {"totalsize": 630244, "port": {}, "224.0.0.251": {"count": 389}, "count":
564          // 5095, "172.27.0.90": {"count": 4618}, "172.27.255.255": {"count": 3}, "35.222.85.5": {"
565          // count": 11}, "35.224.99.156": {"count": 74}, "protocol": {"TCP": 137, "NBNS": 3, "DATA-
566          // TEXT-LINES": 1, "HTTP": 24, "_WS.MALFORMED": 1, "JSON": 7, "DNS": 4922}, "length": 123},
567          // "172.27.0.90": {"totalsize": 764260, "port": {}, "172.27.0.82": {"count": 4618},
568          // "172.27.0.115": {"count": 10}, "count": 5313, "172.27.0.78": {"count": 287},
569          // "172.27.0.135": {"count": 62}, "172.27.0.32": {"count": 336}, "protocol": {"TCP": 540, "
570          // DATA-TEXT-LINES": 3, "HTTP": 59, "_WS.MALFORMED": 1, "BOOTP": 116, "JSON": 51, "DNS":
571          // 4543}, "length": 143},
572          // "172.27.0.32": {"totalsize": 86898, "port": {}, "count": 336, "172.27.0.90": {"count":
573          // 336}, "protocol": {"TCP": 232, "BOOTP": 54, "JSON": 25, "HTTP": 25}, "length": 258},
574          // "172.27.0.12": {"172.27.0.90": {"count": 62}, "totalsize": 21204, "count": 62, "protocol
575          // ": {"BOOTP": 62}, "length": 342},
576          // "172.27.0.115": {"173.194.12.91": {"count": 12}, "protocol": {"QUIC": 410, "TCP": 48, "
577          // SSL": 17, "HTTP": 16, "XML": 1, "DNS": 12}, "224.0.0.251": {"count": 2}, "172.217.6.78":
578          // {"count": 31}, "port": {}, "172.27.0.90": {"count": 10}, "239.255.255.250": {"count":
579          // 15}, "172.217.6.68": {"count": 19}, "length": 803, "172.217.0.42": {"count": 35},
580          // "74.125.170.123": {"count": 11}, "74.125.170.122": {"count": 298}, "count": 504,
581          // "52.49.211.202": {"count": 13}, "216.58.194.174": {"count": 31}, "216.58.195.67": {"
582          // count": 5}, "172.217.5.106": {"count": 15}, "totalsize": 405081, "216.58.195.68": {"
583          // count": 7}}};
584          this.networkLinks = { "172.27.0.15": ["172.27.0.90", "172.27.0.12", "172.27.0.152",
585          // "172.27.0.78"], "172.27.0.152": ["172.27.0.12"], "172.27.0.12": ["172.27.0.90"],
586          // "172.27.0.90": ["172.27.0.78"] };
587          this.getData();
588      };
589      HomeComponent.prototype.getTrafficControl = function () {

```

```

570 //General logic:
571 //Initial:
572 //Parse JSON Data and get array of slowed and prioritized IPs
573 //Add IP:
574 //Add new IP to current list, add to JSON Object, send to server to update
575 //Remove IP:
576 //Remove from current list, remove from JSON Object, send to server to update file
577 //Make a call to API endpoint to get a list of the currently slowed IPs
578 //Make a call to API endpoint to get a list of the currently prioritized IPs
579 };
580 HomeComponent.prototype.throttleIP = function () {
581 };
582 HomeComponent.prototype.prioritizeIP = function () {
583 };
584 HomeComponent.prototype.createNetGraph = function () {
585     var nodes = new vis__WEBPACK_IMPORTED_MODULE_3__["DataSet"]();
586     nodes.add({ id: "Internet", x: 0, y: -600, shape: 'image', image: '../assets/images/
        internet.png', size: 50 });
587     var x = 0;
588     var y = -500;
589     for (var i = 0; i < this.ips.length; i++) {
590         nodes.add({ id: this.ips[i], label: this.ips[i], x: x, y: y });
591         if (i == 0) {
592             x = 300;
593             y = -300;
594         }
595         else {
596             x -= 200;
597             y = -300;
598         }
599     }
600     // create an array with edges
601     var edges = new vis__WEBPACK_IMPORTED_MODULE_3__["DataSet"]();
602     edges.add({ from: "Internet", to: "172.27.0.15" });
603     for (var key in this.networkLinks) {
604         for (var i = 0; i < this.networkLinks[key].length; i++) {
605             edges.add({ from: key, to: this.networkLinks[key][i], color: 'red' });
606         }
607     }
608     // create a network
609     var container = this.graph.nativeElement;
610     // provide the data in the vis format
611     var data = {
612         nodes: nodes,
613         edges: edges
614     };
615     var width = 400;
616     var height = 400;
617     var options = {
618         edges: {
619             color: {
620                 color: 'white',
621                 highlight: 'blue'
622             },
623             smooth: true,
624             length: 300,
625             width: 8
626         },
627         interaction: {

```

```

628         dragNodes: false,
629         dragView: false,
630         hover: true,
631         zoomView: false
632     },
633     physics: {
634         stabilization: true,
635     },
636     nodes: {
637         fixed: {
638             x: true,
639             y: true
640         },
641         shape: 'box',
642         font: {
643             size: 28
644         },
645         color: {
646             background: '#A3ADC4',
647             border: '#253031'
648         },
649         borderWidth: 3
650     }
651 };
652 var network = new vis__WEBPACK_IMPORTED_MODULE_3__["Network"](container, data, options);
653 //click event handler
654 var self = this;
655 network.on("click", function (params) {
656     var trafficLabels = [];
657     var trafficData = [];
658     if (params.nodes[0] != "Internet") {
659         self.viewingIp = params.nodes[0];
660         var data = self.networkData[params.nodes[0]];
661         for (var key in data) {
662             if (data[key].count && trafficLabels.length < 21) {
663                 trafficLabels.push(key);
664                 trafficData.push(data[key].count);
665             }
666             else if (trafficLabels.length > 20) {
667                 break;
668             }
669         }
670         self.packetChart = new chart_js__WEBPACK_IMPORTED_MODULE_2__["Chart"]('trafficChart',
671             {
672                 type: 'bar',
673                 data: {
674                     datasets: [{
675                         data: trafficData,
676                         backgroundColor: "#5C6784"
677                     }],
678                     labels: trafficLabels
679                 },
680                 options: {
681                     legend: {
682                         display: false
683                     },
684                     maintainAspectRatio: false,
685                     title: {
686                         display: true,

```

```

686         text: "Packets Sent/Received"
687     }
688 }
689 });
690 var protLabels = [];
691 var protData = [];
692 for (var key in data.protocol) {
693     protLabels.push(key);
694     protData.push(data.protocol[key]);
695 }
696 self.protChart = new chart_js__WEBPACK_IMPORTED_MODULE_2__["Chart"]('protChart', {
697     type: 'pie',
698     data: {
699         datasets: [{
700             data: protData,
701             backgroundColor: ["#0074D9", "#FF4136", "#2ECC40", "#FF851B", "#7FDBFF",
702                 "#B10DC9", "#FFDC00", "#001f3f", "#39CCCC", "#01FF70", "#85144b",
703                 "#F012BE", "#3D9970", "#111111", "#AAAAAA", "#CAE1FF", "#00E5EE",
704                 "#00C78C", "#8FBC8F", "#FFD700", "#FFA500", "#FF6347"]
705         }],
706         labels: protLabels
707     }
708 });
709 };
710 HomeComponent.prototype.getData = function () {
711     var _this = this;
712     this.meshService.getPacketData().subscribe(function (data) {
713         var trafficCounts = [];
714         var routeCounts = [];
715         var lengths = [];
716         var prots = [];
717         var protCounts = [];
718         var protMap = new Map();
719         var externalIps = [];
720         var externalIpsCount = [];
721         console.log(data);
722         _this.networkData = data;
723         for (var key in _this.networkData) {
724             for (var prot in _this.networkData[key]["protocol"]) {
725                 if (protMap.has(prot))
726                     protMap.set(prot, protMap.get(prot) + _this.networkData[key]['protocol'][prot]);
727                 else
728                     protMap.set(prot, _this.networkData[key]['protocol'][prot]);
729             }
730             if (!_this.ips.includes(key) && _this.networkData[key].count) {
731                 externalIps.push(key);
732                 externalIpsCount.push(_this.networkData[key].count);
733             }
734         }
735         for (var i = 0; i < _this.ips.length; i++) {
736             if (_this.ips[i] in _this.networkData) {
737                 trafficCounts.push(_this.networkData[_this.ips[i]].count);
738                 lengths.push(_this.networkData[_this.ips[i]].length);
739                 routeCounts.push(_this.networkData[_this.ips[i]].routecount);
740             }
741         }
742     });
743 }

```

```

741         trafficCounts.push(0);
742         lengths.push(0);
743     }
744 }
745 console.log(routeCounts);
746 prots = Array.from(protMap.keys());
747 protCounts = Array.from(protMap.values());
748 var protColors = [];
749 _this.createNetGraph();
750 _this.externalChart = new chart_js__WEBPACK_IMPORTED_MODULE_2__["Chart"]('externals', {
751     type: 'bar',
752     data: {
753         labels: externalIps,
754         datasets: [
755             {
756                 data: externalIpsCount,
757                 backgroundColor: "#1D263B"
758             }
759         ]
760     },
761     options: {
762         legend: {
763             display: false
764         },
765         scales: {
766             xAxes: [{
767                 display: true,
768                 ticks: {
769                     fontSize: 15
770                 }
771             }],
772             yAxes: [{
773                 display: true,
774                 ticks: {
775                     fontSize: 15
776                 }
777             }],
778         },
779         title: {
780             display: true,
781             text: "External Traffic",
782             fontColor: "black",
783             fontSize: 30
784         },
785         gridLines: {
786             color: "black",
787             lineWidth: 10
788         }
789     }
790 });
791 _this.trafficChart = new chart_js__WEBPACK_IMPORTED_MODULE_2__["Chart"]('traffic', {
792     type: 'bar',
793     data: {
794         labels: _this.ips,
795         datasets: [
796             {
797                 label: "Packets",
798                 data: trafficCounts,
799                 backgroundColor: "#70A0AF"

```

```

800         }
801     ]
802 },
803 options: {
804     legend: {
805         display: true
806     },
807     scales: {
808         xAxes: [{
809             display: true,
810             ticks: {
811                 fontSize: 15
812             }
813         }],
814         yAxes: [{
815             display: true,
816             ticks: {
817                 fontSize: 15
818             }
819         }],
820     },
821     title: {
822         display: true,
823         text: "Node Traffic",
824         fontColor: "black",
825         fontSize: 30
826     },
827     gridLines: {
828         color: "black",
829         lineWidth: 10
830     }
831 }
832 });
833 _this.lengthChart = new chart_js__WEBPACK_IMPORTED_MODULE_2__["Chart"]("length", {
834     type: 'bar',
835     data: {
836         labels: _this.ips,
837         datasets: [
838             {
839                 label: "Length",
840                 data: lengths,
841                 backgroundColor: "#4CB963"
842             },
843             {
844                 label: "Route Count",
845                 data: routeCounts,
846                 backgroundColor: "#706993"
847             }
848         ]
849     },
850     options: {
851         legend: {
852             display: true
853         },
854         scales: {
855             xAxes: [{
856                 display: true,
857                 ticks: {
858                     fontSize: 15

```

```

859         }
860     }],
861     yAxes: [{
862         display: true,
863         ticks: {
864             fontSize: 15
865         }
866     }],
867 },
868     title: {
869         display: true,
870         text: "Length of Packets",
871         fontColor: "black",
872         fontSize: 30
873     },
874     gridLines: {
875         color: "black",
876         lineWidth: 10
877     }
878 }
879 });
880 _this.totalProtChart = new chart_js__WEBPACK_IMPORTED_MODULE_2__["Chart"]("protocols", {
881     type: 'pie',
882     data: {
883         labels: prots,
884         datasets: [
885             {
886                 data: protCounts,
887                 backgroundColor: ["#0074D9", "#FF4136", "#2ECC40", "#FF851B", "#7FDBFF", "#B10DC9", "#FFDC00", "#001f3f", "#39CCCC", "#01FF70", "#85144b", "#F012BE", "#3D9970", "#111111", "#AAAAAA"]
888             }
889         ]
890     },
891     options: {
892         title: {
893             display: true,
894             text: "Protocols Used",
895             fontColor: "black",
896             fontSize: 30
897         },
898     }
899 });
900 });
901 };
902 HomeComponent.prototype.sendTrafficControlData = function () {
903     var body = {
904         throttle: this.throttleInput,
905         priority: this.prioritizeinput
906     };
907     this.meshService.sendTrafficControlData(body).subscribe(function (data) {
908         console.log(data);
909     });
910 };
911 HomeComponent.prototype.closeModal = function () {
912     this.viewingIp = null;
913     if (this.packetChart)
914         this.packetChart.destroy();
915     if (this.protChart)

```



```

916         this.protChart.destroy();
917     };
918     __decorate([
919         Object(_angular_core__WEBPACK_IMPORTED_MODULE_0__["ViewChild"])(`graphContainer`),
920         __metadata("design:type", _angular_core__WEBPACK_IMPORTED_MODULE_0__["ElementRef"])
921     ], HomeComponent.prototype, "graph", void 0);
922     HomeComponent = __decorate([
923         Object(_angular_core__WEBPACK_IMPORTED_MODULE_0__["Component"])({
924             selector: 'app-home',
925             template: __webpack_require__(/*! ./home.component.html */ "./src/app/home/home.component.html"),
926             styles: [__webpack_require__(/*! ./home.component.css */ "./src/app/home/home.component.css")]
927         }),
928         __metadata("design:paramtypes", [_mesh_service__WEBPACK_IMPORTED_MODULE_1__["MeshService"]])
929     ], HomeComponent);
930     return HomeComponent;
931 }());
932
933
934
935 /***/ }),
936
937 /***/ "./src/app/mesh.service.ts":
938 /*!*****!\
939  !*** ./src/app/mesh.service.ts ***!
940  \*****/
941  /*! exports provided: MeshService */
942  /***/ (function(module, __webpack_exports__, __webpack_require__) {
943
944    "use strict";
945    __webpack_require__.r(__webpack_exports__);
946    /* harmony export (binding) */ __webpack_require__.d(__webpack_exports__, "MeshService", function()
947      { return MeshService; });
948    /* harmony import */ var _angular_core__WEBPACK_IMPORTED_MODULE_0__ = __webpack_require__(/*!
949      @angular/core */ "./node_modules/@angular/core/fesm5/core.js");
950    /* harmony import */ var _angular_common_http__WEBPACK_IMPORTED_MODULE_1__ = __webpack_require
951      ____(/*! @angular/common/http */ "./node_modules/@angular/common/fesm5/http.js");
952    var __decorate = (undefined && undefined.__decorate) || function (decorators, target, key, desc) {
953      var c = arguments.length, r = c < 3 ? target : desc === null ? desc = Object.
954        getOwnPropertyDescriptor(target, key) : desc, d;
955      if (typeof Reflect === "object" && typeof Reflect.decorate === "function") r = Reflect.decorate
956        (decorators, target, key, desc);
957      else for (var i = decorators.length - 1; i >= 0; i--) if (d = decorators[i]) r = (c < 3 ? d(r)
958        : c > 3 ? d(target, key, r) : d(target, key)) || r;
959      return c > 3 && r && Object.defineProperty(target, key, r), r;
960    };
961    var __metadata = (undefined && undefined.__metadata) || function (k, v) {
962      if (typeof Reflect === "object" && typeof Reflect.metadata === "function") return Reflect.
963        metadata(k, v);
964    };
965
966    var MeshService = /** @class */ (function () {
967      function MeshService(http) {
968        this.http = http;
969      }
970      MeshService.prototype.getPacketData = function () {
971        return this.http.get('http://172.27.0.15:3030/getFile');
972      }
973    }());

```

```

966     };
967     MeshService.prototype.getLinks = function () {
968         return this.http.get('http://172.27.0.15:3030/getArp');
969     };
970     MeshService.prototype.getIPs = function () {
971         return this.http.get('http://172.27.0.15:3030/getIpList');
972     };
973     MeshService.prototype.sendTrafficControlData = function (body) {
974         return this.http.post('http://172.27.0.15:3030/editTrafficControl', body);
975     };
976     MeshService = __decorate([
977         Object(_angular_core__WEBPACK_IMPORTED_MODULE_0__["Injectable"])(
978             {providedIn: 'root'}
979         ),
980         __metadata("design:paramtypes", [_angular_common_http__WEBPACK_IMPORTED_MODULE_1__["HttpClient"]])
981     ], MeshService);
982     return MeshService;
983 }());
984
985
986
987 /***/ }},
988
989 /***/ "./src/environments/environment.ts":
990 /*!*****!*\
991  !*** ./src/environments/environment.ts ***!
992  \*****/
993 /*! exports provided: environment */
994 /***/ (function(module, __webpack_exports__, __webpack_require__) {
995
996     "use strict";
997     __webpack_require__.r(__webpack_exports__);
998     /* harmony export (binding) */ __webpack_require__.d(__webpack_exports__, "environment", function()
999         { return environment; });
1000     // This file can be replaced during build by using the 'fileReplacements' array.
1001     // 'ng build --prod' replaces 'environment.ts' with 'environment.prod.ts'.
1002     // The list of file replacements can be found in 'angular.json'.
1003     var environment = {
1004         production: false
1005     };
1006     /*
1007     * For easier debugging in development mode, you can import the following file
1008     * to ignore zone related error stack frames such as 'zone.run', 'zoneDelegate.invokeTask'.
1009     * This import should be commented out in production mode because it will have a negative impact
1010     * on performance if an error is thrown.
1011     */
1012     // import 'zone.js/dist/zone-error'; // Included with Angular CLI.
1013
1014
1015 /***/ }},
1016
1017 /***/ "./src/main.ts":
1018 /*!*****!*\
1019  !*** ./src/main.ts ***!
1020  \*****/
1021 /*! no exports provided */
1022 /***/ (function(module, __webpack_exports__, __webpack_require__) {

```

```

1023
1024 "use strict";
1025 __webpack_require___.r(__webpack_exports__);
1026 /* harmony import */ var _angular_core__WEBPACK_IMPORTED_MODULE_0__ = __webpack_require__(/*!
    @angular/core */ "./node_modules/@angular/core/fesm5/core.js");
1027 /* harmony import */ var _angular_platform_browser_dynamic__WEBPACK_IMPORTED_MODULE_1__ = __webpack
    _require__(/*! @angular/platform-browser-dynamic */ "./node_modules/@angular/platform-browser-
    dynamic/fesm5/platform-browser-dynamic.js");
1028 /* harmony import */ var _app_app_module__WEBPACK_IMPORTED_MODULE_2__ = __webpack_require__(/*! ./
    app/app.module */ "./src/app/app.module.ts");
1029 /* harmony import */ var _environments_environment__WEBPACK_IMPORTED_MODULE_3__ = __webpack_require
    ____(/*! ./environments/environment */ "./src/environments/environment.ts");
1030
1031
1032
1033
1034 if (_environments_environment__WEBPACK_IMPORTED_MODULE_3__["environment"].production) {
1035     Object(_angular_core__WEBPACK_IMPORTED_MODULE_0__["enableProdMode"])();
1036 }
1037 Object(_angular_platform_browser_dynamic__WEBPACK_IMPORTED_MODULE_1__["platformBrowserDynamic"])().
    bootstrapModule(_app_app_module__WEBPACK_IMPORTED_MODULE_2__["AppModule"])
1038     .catch(function (err) { return console.error(err); });
1039
1040
1041 /***/ }),
1042
1043 /***/ 0:
1044 /*!*****!\
1045     !*** multi ./src/main.ts ***!
1046     \*****/
1047 /*! no static exports found */
1048 /***/ (function(module, exports, __webpack_require__) {
1049
1050 module.exports = __webpack_require__(/*! /Users/Matt/Desktop/Coding/JavaScript/MeshApp/
    MeshAppFrontend/meshFrontend/src/main.ts */"./src/main.ts");
1051
1052
1053 /***/ })
1054
1055 },[[0,"runtime","vendor"]]);
1056 //# sourceMappingURL=main.js.map

```

Bibliography

- [1] goTenna Website, “gotenna.” <https://www.gotenna.com/>.
- [2] J. Tremback, “Faster, cheaper, decentralized internet.” <https://althea.org/>.
- [3] O. Contributors, “Openwisp.” <http://openwisp.org/whatis.html>.
- [4] M. S. Singh and V. Talasila, “A practical evaluation for routing performance of batman-adv and hwmn in a wireless mesh network test-bed,” in *2015 International Conference on Smart Sensors and Systems (IC-SSS)*, pp. 1–6, Dec 2015.
- [5] J. Kim, M. Seo, and S. Lee, “G.o.t.h.a.m. - mesh network management and visualization for batman-adv,” Jan 2017.
- [6] “Advanced traffic control.” https://wiki.archlinux.org/index.php/Advanced_traffic_control.
- [7] p. justin, “In defense of the slim protocol,” Jan 2018.
- [8] A. Rao, E. Dimogerontakis, M. Selimi, A. Ali, L. Navarro, and A. Sathiasaelan, “Blockchain for economically sustainable wireless mesh networks,” 11 2018.
- [9] “Openwisp 2 documentation.” <http://openwisp.io/docs/>.
- [10] M. Jnemann, “802.11s wireless mesh network.” https://mjuenema.github.io/80211s_wireless_mesh/.
- [11] “open80211s.” <https://github.com/o11s/open80211s/wiki/HOWTO>.