The Analysis of Home Network Traffic

EECS 325: Computer Networks
Project #5

Due: December 7, 11:59 PM

The last project of the semester involves conducting measurements of some network facet / behavior and reporting the results. The goal of this project is to move beyond learning about networks from the textbook and towards learning about networks via observations of how they work in practice. This is a student-directed project. Part of the project is arriving at the theme and the details yourself. While there are a few limitations you are free to measure whatever you find most interesting about the network. You may use common tools to help with your investigation (e.g., ping, traceroute).

Jacob Alspaw
Case Western Reserve University
jaa134@case.edu

Table of Contents

1. Introduction

1.1 Project Description

For project 5, I have developed a basic tool called *NetworkSniffer* that helps its users analyze home-network usage statistics. *NetworkSniffer* is a tool that will allow its users to diagnose how a router is being used by its connected clients. Usage is measured in number and size of packets that are transmitted on the network. The tool enables users to collect, track, and then visualize useful information about their currently connected network. *NetworkSniffer* tracks how many devices are utilizing the selected network, which devices are using the network, a general idea of how much data each person is using, what kind of data each person is using, and how much data the collective network is transferring.

All data is processed in real time immediately after it is collected and then graphically displayed back to the user in a neat and formatted fashion. On the home screen, the suspected number of clients and number of transferred IPv4 packets are shown in sliding bar graphs. The last 25 measurements are displayed to the user. The user can then stop the service, at which point a report is generated. All client and packet data collected throughout the sessions is then displayed again in graphs. Above these graphs, users can find statistics about their home network traffic.

*Images of the tooling can be found at the end of this document.

1.2 Project Inspiration and Motivation

The purpose of this project is to conduct measurements of IPv4 network traffic and behavior, and then subsequently report and analyze the results. The development of the *NetworkSniffer* tooling has allowed me as a developer to move beyond learning about networks from the textbook and to make a tool that literally allows users to observe how their home network is functioning in real time.

The inspiration for this project came from a recent YouTube video that caught my eye on a popular text-based forum called Reddit. In the video, a self-designated hacker finds a mesh-network of Raspberry Pis hidden throughout the University of California San Diego library. At first glance, the Pi's seemed to be performing nefarious activities by collecting packets transferred over the school network. It is later determined in the video that these devices were actually property of a company called Waitz that harmlessly reported the real-time "busyness" for locations around UCSD campus.

This YouTube video gave me all the insight I needed to determine my project idea because I also wanted to analyze real-time "busyness" for locations by analyzing a network's traffic and usage statistics. After discussing the project idea with Professor Allman in class and ensuring that the planned theme was suitable for the project, meaning that it followed the project guidelines and limitations, I was sure that analyzing my own personal home network traffic data was a general concept that inspired and motivated me as a developer.

Next semester, I would like to further develop this project and transform it into something even greater like the Waitz service at University of California at San Diego. I plan on discussing the idea with a representative of the campus IT department that could bring a system like Waitz to campus. A personalized service that enables network traffic analysis could help CWRU students plan for or avoid parts of campus when they are busiest. For example, with a service like Waitz at CWRU, students would then be better prepared for visiting the library during midterms and finals or visiting Veale during the spring graduation ceremony.

*YouTube, Reddit, and Waitz links are provided below.

2. Procedure

2.1 Developing the Tools

NetworkSniffer is a simple c++ based application that uses many components of the Qt framework, specifically v5.11. The tool starts by opening a home page that allows the user to start tracking statistics at the push of a button. Once the button is pushed, two additional threads are created. Each thread is responsible for collecting and parsing data in parallel from a distinct networking command. Each command will be discussed in greater detail later. After command execution and parsing is complete, the threads will wait a few seconds before repeating the process again. The original thread will take the parsed data and display the data in an appropriate format to the user.

The service can collect data for as long as the user desires. If the user wishes to end data collection, then the user can click a button to generate a report regarding the collected data. The user must confirm that the session completed because at this point, both threads are stopped and their results are analyzed.

NetworkSniffer leverages two separate linux-based command line utilities to help collect network traffic information. The first tool is called "arp-scan" which provides a simple ARP scanner to users. Arp-Scan can be used to scan the network of a certain interface for alive hosts. It shows the IP address and MAC addresses of all the hosts/nodes found. The second command line utility

that *NetworkSniffer* leverages is called “tcpdump”. Tcpdump is a powerful packet-analyzer that prints out a description of the contents of packets transferred over a network interface. The results of both commands are printed to a time-stamped text file that is analyzed after the command completes. The functions that both command line utilities use to read and write packets require root privilege, so it is essential that both programs are granted SUID root before running the *NetworkSniffer* application.

At any point during its function, *NetworkSniffer* employs great error handling. A valid connection to the internet must be established before the service can begin. If there is not a valid connection, then the application will show an error page. Determining if the system running *NetworkSniffer* has a valid connection is done by sending a simple HTTP Get request to the Google homepage. If the request returns without errors, then the application allows users to proceed. If any system command fails, then the application will also show errors. Lastly, requests to open files are also safely wrapped in error handling code. All application errors are logged in Stderr.

2.2 Gathering Data

Gathering data was tricky and severely limited the times at which I could work on this project. After my discussion with Professor Allman, it was clear that this project would only be permitted if data was collected on my own personal network or my family’s home network in which consent was given by each family member. I agree with Professor Allman’s assessment of the situation. I would also recommend only running this application on a network in which the user is the network operator. The leveraged command line utilities, “arp-scan” and “tcpdump”, are considered promiscuous operations on non-personal networks by network operators as they are often misused for nefarious purposes. On non-personal networks, these command are essentially stealing access to private data to which consent has not been expressly given.

To gather the data, I left my machine running the application for a significant period of time. On multiple occasions, the duration reached upwards of about 20,000 seconds or about five and a half hours. Each sample taken is saved to a text file that is soon after processed by the *NetworkSniffer* application. I made sure to leave the application running when my family was leaving our house, arriving to our house, or sleeping. Collecting samples at different times of day allowed me to diversify my results and findings discussed later in this document.

The “arp-scan” command that the application uses to collect data is “arp-scan -l -g -q > output.txt”. The command line utility generate addresses from the network interface configuration (“-l”). It uses the network interface IP address and network mask to generate the list of target host addresses. The list will include the network and broadcast addresses, so an

interface address of 10.0.0.1 with netmask 255.255.255.0 would generate 256 target hosts from 10.0.0.0 to 10.0.0.255 inclusive. Duplicate packets are ignored ("-g") and the minimal output is displayed ("-q"). The results of the command are then saved into a text file.

The "tcpdump" command that the application uses to collect data is "timeout 10 tcpdump ip -v -t -q -n -N > output.txt". The command will then be run for ten seconds before it is stopped. During this time, IPv4 packet information will be collected and then saved into a text file. IPv6 packets are filtered out by the "ip" option. The command specifies to use verbose mode ("-v") which displays the transport protocol for the packet, but then hide more in-depth protocol information ("-q") that this application does not need to process. The timestamp of each packet is then removed ("-t"). Lastly, the packets should display IP addresses and port numbers instead of domain and service names ("-n") and not print the domain name qualification of host names ("-N").

*A link to the data can be found below.

2.3 Analyzing Data

From the data that *NetworkSniffer* has collected with the help of the command line utilities, we are able to parse the results into many helpful statistics. As seen in the images below, the report section shows the statistics that we have collected as a summary of the application's session. The statistics are as follows:

Client Info	**IPv4 Traffic Analysis**
Minimum number of clients to respond to ARP requests.	The number of TCP packets processed by the command.
Maximum number of clients to respond to ARP requests.	The number of UDP packets processed by the command.
The number of times the command was run to determine results.	The number of other packets processed by the command.
The average number of clients to respond to the ARP requests.	The average length of a TCP packet.
The distinct number of IP addresses linked to an ARP response.	The average length of a UDP packet.
The distinct number of MAC addresses linked to an ARP response.	The source IP address associated with the highest number of packets.

The distinct number of clients to respond to ARP requests.	

* A client being defined as a distinct MAC address <-> IP address pairing.

3. Results

3.1 Home Networks and Promiscuity

From our results, the fact that we were able to collect any in the first place, tells us a great deal about my home network security and layout. For the commands to work, the network interface controller on the device that runs the *NetworkSniffer* application must have enabled promiscuous mode. The mode tells the controller to pass all traffic it receives to the central processing unit rather than passing only the frames that the controller is specifically programmed to receive.

In all IEEE 802 networks, ethernet frames include a destination MAC address. If a node is in non-promiscuous mode, the network interface controller receives a frame and drops it unless the frame's destination address matches the network interface controller's MAC address. However, in promiscuous mode the network interface controller skips checking the destination MAC address and instead allows all frames through. This gives a promiscuous node the ability to read frames, and in our case packets intended for other machines or network devices.

Promiscuous mode is often used for packet sniffing, which is essentially taking a copy of information intended for others on the network. Many operating systems require superuser privileges to enable this mode. This is why it was important to enable SUID root on both "arp-scan" and "tcpdump" because they require promiscuous mode to function.

It is not easy for home networks to combat promiscuous mode and it's often difficult even for large corporations to combat. Because we can receive the ethernet frames from all of the devices connected to the home router, we can conclude that our home network does not employ many, if any, software or hardware security measures against promiscuous nodes. One way to protect against a promiscuous node is to use switches instead of hubs. Switches will intelligently route traffic only to those ports that need it, while a hub broadcasts traffic to all ports. Another solution, that I am sure my home network is not using, is to use an end-to-end encryption protocol like IPsec.

3.2 ARP Requests and Reliability

For the results of this project, the number of clients connected to the home network is solely determined by the results of the "arp-scan" command. As mentioned earlier, "arp-scan" can be used to scan the network of a certain interface for alive hosts. It shows the IP address and MAC addresses of all the hosts/nodes found. It does so by using Address Resolution Protocol that maps and IP address to a MAC address.
The command finds connected IPv4 devices by sending broadcast ARP requests to every IP address on the subnet and using UDP as a response transport protocol. IPv4 connected devices must respond if they are to be communicating using IPv4 ethernet. One ARP packet is sent for each for each target host, with the target protocol address set to the IP address of the host running the *NetworkSniffer* application. If a host does not respond, then the ARP packet will be re-sent once more. The tool only displays any responses that are received.

From this information, we can gather that many ARP packets on the network are being lost. The results of continuously running the "arp-scan" command show in our graphs that the number of targets to respond successfully is rarely consistent even after one retry attempt. Over a trial run of the application during a brief five minute period, a successful response by each device is unlikely evident by the majority of values in the trend graph sitting beneath the dotted maxima line. The results can only ever show us a general number of connected clients.

On very few occasions, there are brief periods in the results that show no clients responded to ARP requests. This reveals an interesting situation that may shine light on exactly where a packet is being lost. For no hosts to respond seems as if packets where being lost on the side of the machine running the *NetworkSniffer* application perhaps because the initial broadcast ARP request was not being distributed to the network successfully. The situation where some hosts would reply, but not all indicates that at least some ARP requests were answered and the machine running the application was successful in sending packets to the network while the receiving machines were not. An extended period of no responses to the ARP requests reveals that the machine running the application was likely removed from the network.

3.3 Packet Protocols

From our results conducted throughout the week, we can determine that my household sends and receives a significantly greater number of TCP packets than UDP packets. This is shown on the report page under the Traffic Analysis section. In nearly every sample run of the application, TCP was shown to be the most used transfer protocol by the household. This would indicate that users of my home network are more heavily using applications where loss cannot be permitted.

Because TCP is a connection based protocol, we can infer many of the trends in the report's IPv4 traffic graph. At times, there are significant spikes in network usage which indicates significantly more bandwidth usage by the network. This is likely the result of a TCP connection being established and large amounts of data being transferred as would happen during a heavy download. I confirmed this hypothesis myself by downloading large files upwards of 200MB and then observed the spikes in the resulting IPv4 traffic trend graph.

TCP and UDP are not the only transfer protocols being used on my network. Beyond TCP and UDP, there is at least one other protocol being leveraged by a device. IGMP packets were seen many times throughout running my application. Internet Group Management Protocol is used by hosts and adjacent routers on IPv4 networks to establish "multicast group memberships". These packets were likely sent from my smart TV that allows for streaming over WiFi.

Depending on the time of day, it is also evident that TCP and UDP traffic varies heavily. UDP seems to see a relative spike at night. This is likely a cause of clients using streaming services like Netflix and YouTube before bed. TCP saw a rise in packet flow around dusk, likely when my brother and I were visiting websites while doing homework for school.

3.4 Clients, Usage, and Time of Day

Even though the ARP protocol is not always successful, it still reveals plenty of information about IPv4 devices on a network. It can show us how many devices are connected to a network. After collecting enough samples, it is apparent that the number of devices connected to the network varies throughout the day when people arrive to and leave from a household. As shown in the client trend graph, on many occasions the approximate number of connected devices increases and decreases. A general increase in the connected devices trend shows someone new connecting to the network and their device responding with ARP packets. A general decrease shows a device disconnecting from the network and no longer responding to ARP requests.

My family is a normal family household. On a workday, both of my parents are out of the house, my brother and I (usually) are at school. The trends as shown by using the *NetworkSniffer* application shows that a majority of devices are connected and actively using the network before bed and in the morning when everyone is home. During mid day, the network has much fewer devices connected to it because everyone and their phones specifically are removed from the network. It is during this midday period that we see the least usage of the network and least number of connected devices. It is at times of day when people are present at the household that we see a greater trend in the number of people connected to the network and far more usage.

3.5 Bandwidth Hogs

Part of the reporting process is keeping a table that maps source IP addresses to the number of packets sent. Doing this allows us to establish who is using the most bandwidth over the network. We call this person a bandwidth hog. As it turns out from the results of collecting data, there were many IP addresses that were very popular amongst our network activity. The report data shows that there are many IP addresses associated with relatively few packets compared to the most popular packet sender. This indicates that one or more source IP address is much busier sending packets than others. One such example is that my smart tv is sending many packets throughout a day when its in use, whereas my Amazon Alexa Echo Dot makes only a few requests per day. The result is my television sending significantly more packets over our network than my Amazon Alexa Echo Dot.

3.6 IP Address Reassignment

When looking at the report statistics, I purposefully included a few measurements as part of a test to see how my router was handling DHCP when devices disconnected from the network and rejoined later in the day. Dynamic Host Configuration Protocol automatically assigns a device-specific IP address to each device on your home router's network, which ensures that no two devices will share an IP address and thus encounter connection errors.

My personal home router has DHCP enabled. The results show that this must be true. Some of the figures will display a value for distinct IP addresses that is greater than the value for distinct MAC addresses. The value for distinct MAC addresses is also less than the number of distinct clients, a client being defined as a MAC address and IP address pair. Because the number of distinct MAC addresses does not match the other two values and is in fact smaller by one, it is apparent that devices are being re-assigned IP addresses over the course of a day. One device must have have been automatically assigned a new and different IP address when it reconnected to the network.

3.7 Possible Network Flooding

Part of this programming assignment is understanding that I am analyzing many packets that are ARP requests coming from the machine running the *NetworkSniffer* application. I chose to include these packets in the final analysis because my machine is a node connected to the network and, afterall, the ARP packets are transferred along the network.

I have concluded that every few seconds I am asking each of my home devices to repeatedly send UDP packets in response to ARP requests. I could observe this by running my application in tandem with another instance of a terminal running “timeout 10 tcpdump -vvv”. The results

show a substantial number of ARP related packets being sent over the network that likely made up a majority of UDP packets that the “tcpdump” command had processed.

If a machine were to run many *NetworkSniffer* applications at once, then I predict that this would greatly flood the network in a short amount of time. Network flooding is a denial of service attack that is designed to bring a network or service down by flooding it with large amounts of traffic. If there are many applications performing ARP requests at the same time, then the number of UDP packets being transferred over the network would multiply quickly. Since my home network did not seem to suffer on the day that I collected data for multiple hours at a time, I would expect that the number of running *NetworkSniffer* applications would have to be very large to flood the network.

4. Helpful Links

- Original Reddit Post:
  https://www.reddit.com/r/hacking/comments/9rm9r6/my_roommate_found_a_bunch_of_these_hidden_behind/

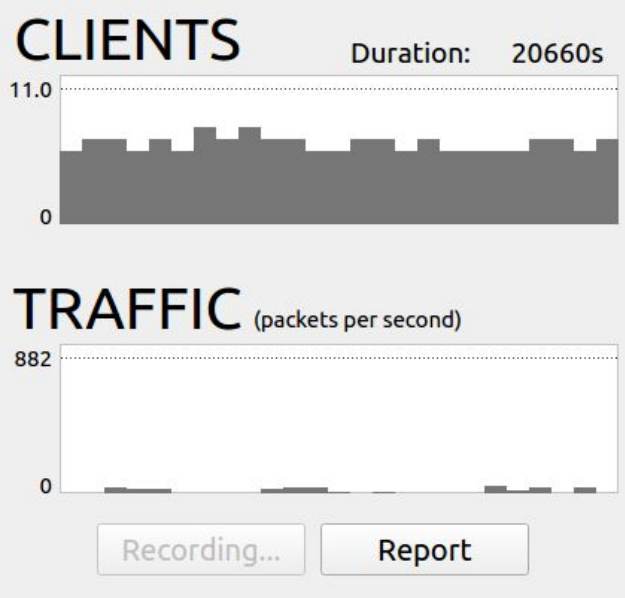
- Analysis YouTube Video
  https://www.youtube.com/watch?v=UeAKTjx_eKA

- Waitz Website
  https://www.ucsdwaitz.com/

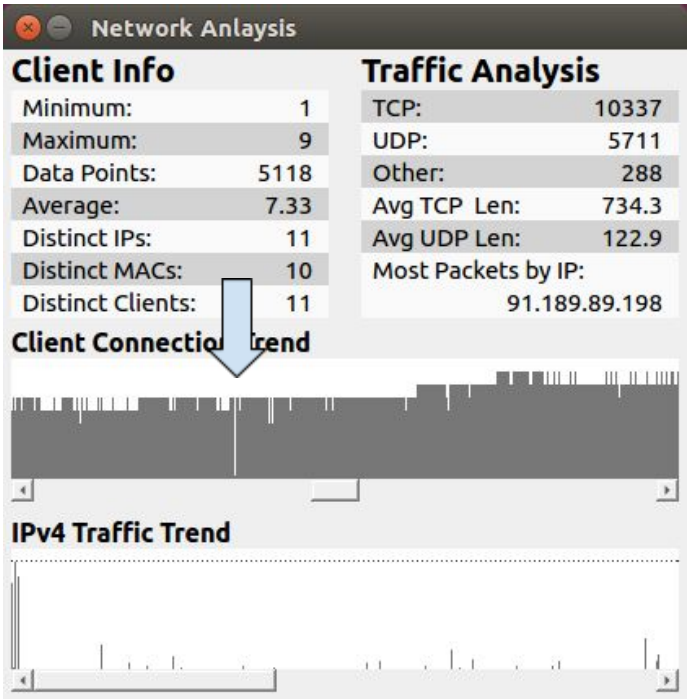
- Project Data
  https://drive.google.com/drive/folders/1NVRX4D6PUwfeojJF0IIIyYddpWTt-W0f?usp=sharing

5. Figures

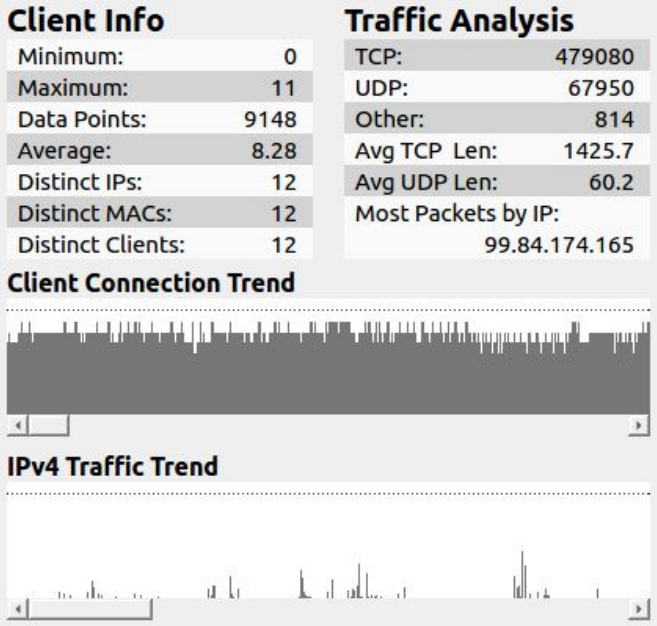
Basic home page.
Collecting data for 20,000 seconds.



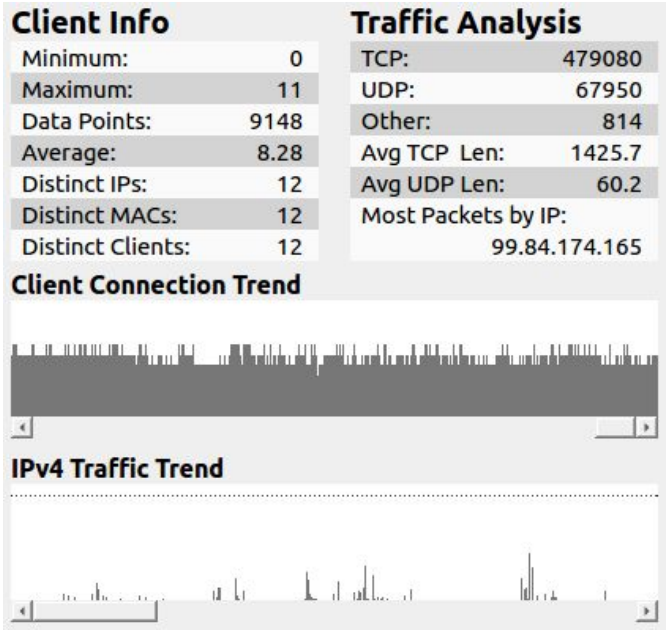
No ARP requests being answered.



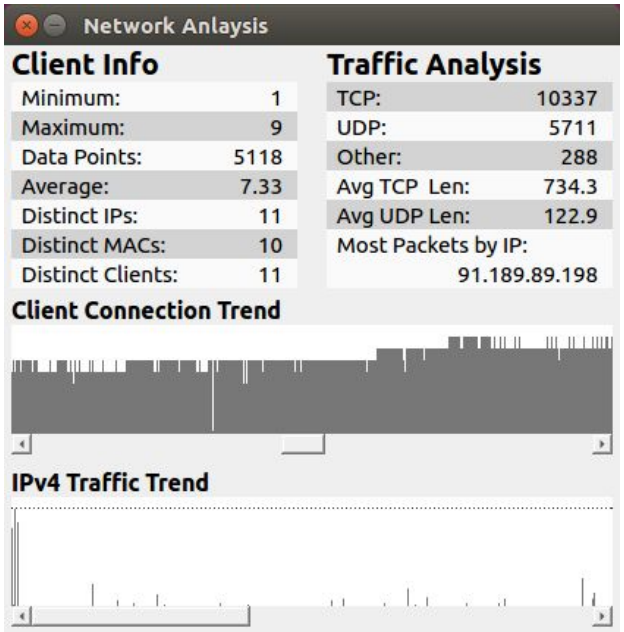
Many Clients



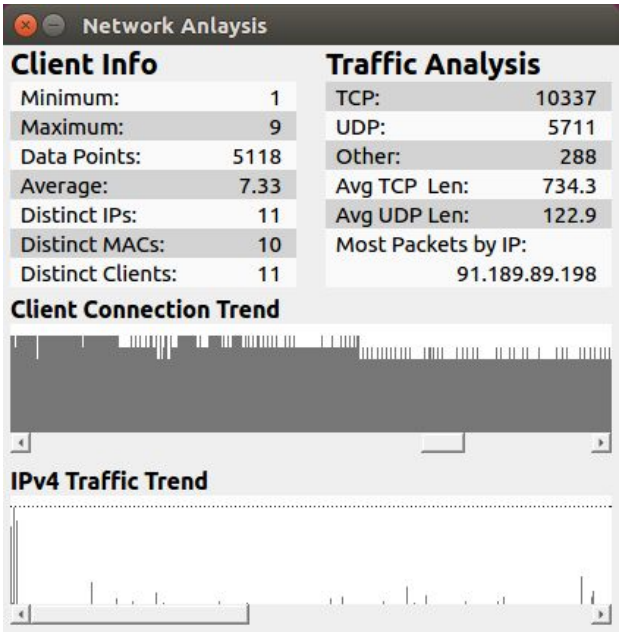
Few Clients



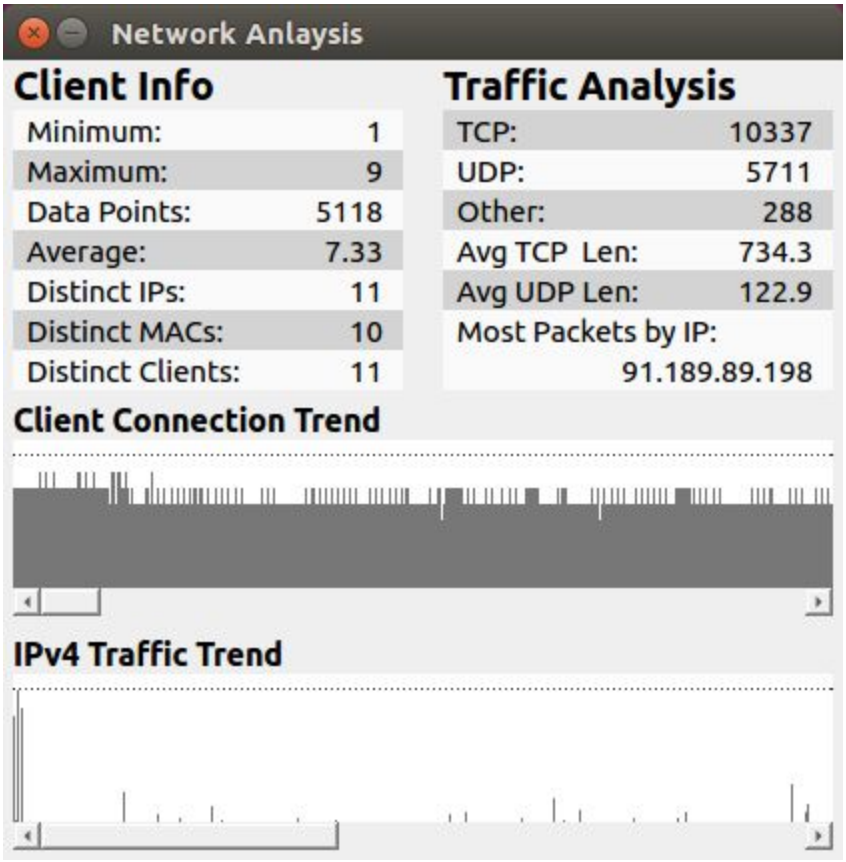
Increasing Client Trend



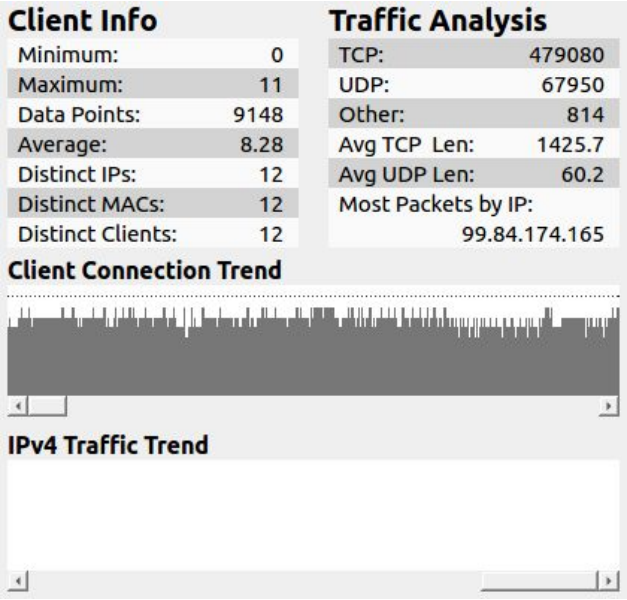
Decreasing Client Trend



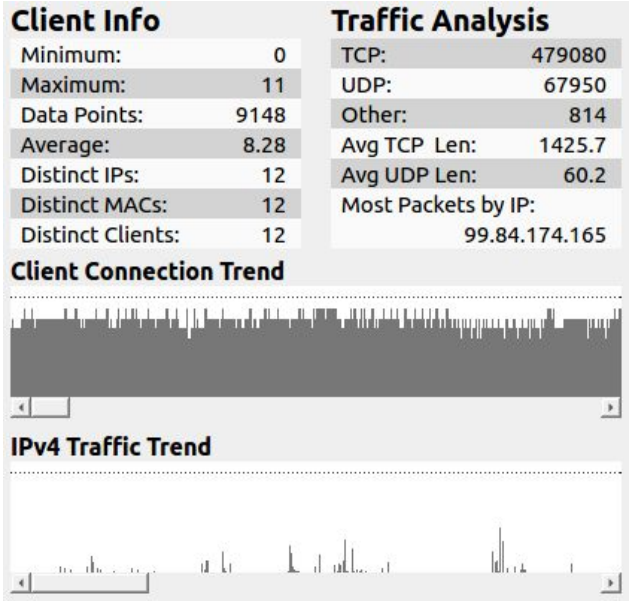
Stable Client Trend



No Network Traffic



Sporadic Network Traffic



Network Usage Spike