**Introduction:**

This documentation will provide a guide to replicating the experiments of

“Experimental Evaluation of the Age ofInformation via Emulation”, a paper that is attached in the following github link: <https://github.com/Johnmancini30/CORE_Research>

**Downloads:**

To run these experiments, you will need to download, CORE, EMANE, and Python3.

To download CORE: <http://coreemu.github.io/core/install.html>

To download EMANE: <https://github.com/adjacentlink/emane/wiki/Install>

To download Python3: <https://www.python.org/downloads/>

Using “pip install <program>” in the command line, you should also download the python libraries:

1. pyautogui
   1. https://pyautogui.readthedocs.io/en/latest/
2. matplotlib
   1. https://matplotlib.org/
3. Scipy - follow website instructions for downloading this package
   1. https://www.scipy.org/install.html

**Experimentation:**

**Creating First Network:**

The first step is to create your first virtual network. See the attached video “creating-first-network.mp4”. This video will show you how to create a 2 router network connected over a WLAN using EMANE’s RFPIPE model, with delay and jitter set to .1 seconds. Note that I set all nodes to be on the subnet 24.

**Creating First Traffic Flow:**

Now you will need to create your first traffic flow. See the attached video “creating-first-traffic-flow.mp4”. This video will show you how to start the virtual network, direct the mgen log output to a place of your choosing, and create a traffic flow with Poisson arrival distribution of 10 packets/second of size 1000 bytes. We then start the flow and stop it, and then look at the output to make sure it worked.

**Automation:**

At this point you know how to manually create a virtual network, as well as a configure and run a traffic flow. For most experimentation purposes, you will want to collect a lot of data. To accommodate this, a bot is supplied that will automate the process of data collection.

The bot will run by executing the script “bot.py” in <https://github.com/Johnmancini30/CORE_Research/blob/master/source/bot.py>

**Configuring the bot:**

This bot clicks based on specified pixels, so first you will need to find where on your screen certain pixel values correspond to. Create the same virtual network you created earlier, and create a traffic flow between each router. Then watch “configuring-bot.mp4”. You will need to install pyautogui to run this bot which can be installed with the command line “pip install pyautogui”. Follow along and input your own screens pixel values for various buttons into the bot.py program.

Lastly, change the values of the rest of the global variables to values that you want, they are commented so you will know what they refer to.

**Running the bot:**

Now that the global variables are changed to what you want them to be, we can run the bot. Watch “running-bot.mp4” to see how this will work. In this video I only want to run 2 flows per parameter, where the parameters are 5 packets per second and 10 packets per second. I save them in a folder called data on my Desktop.

**Data Parsing:**

With the help of the bot, you have now been able to collect as much data as you need. I have collected 5 traffic flows per parameter, where the parameters are lambda = .3, .5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9.5, 9.7, and on average 100 packets are sent per run. Therefore, I have 12 directores, each with a file traffic0.log, … , traffic 4.log. We will now run them through the parser to generate a corresponding age file for them. These files are in the directory “/home/jm/Desktop/CORE\_Research/data/poisson/” on my computer. Go to the main method and specify the instructions and the directory. The code left there should make it obvious how to use this.

The directory name and the instructions are passed to the create\_files function. This function calls parse\_file which maps each traffici.log file to a latencyi.txt file which contains the latency, reception, generation, and sequence of each packet for the traffic flow in traffici.log. Then, the write\_age\_file function is called which creates a file containing average age, average peak age, average latency, and average interarrival time for each traffic flow.

Watch “parsing.mp4” to see an example of that being done for the directory I specified above. The code for this is in the parser.py file

**Analysis:**

Now we will analyze the data we have just parsed. We will be analyzing the agei.txt files per parameter by averaging the values in the agei.txt file per parameter, and then plotting them against their parameter.

The functions in analysis.py correspond to different experiments conducted throughout the paper.

1. The plot\_age function plots the average age vs. average latency vs. average interarrival time.
2. The plot\_average\_age\_distributions functions plots the average age from the emulation and theoretical results for poisson and periodic
3. The plot\_avg\_age\_vs\_avg\_peak\_age\_mm1 plots the average age and average peak age for emulated results vs theoretical results for poisson arrival distribution
4. The plot\_avg\_age\_vs\_avg\_peak\_age\_dm1 plots the average age and average peak age for emulated results vs theoretical results for periodic arrival distribution

Watch “analysis.mp4” for an example of this for the same data we parsed in the Data Parsing section.

**Conclusion:**

From this documentation you have learned to

1. Create a virtual network and a traffic flow, and run the traffic flow on the network.
2. Configure and run a bot that will do data collection for you.
3. Parse the data to get valuable information from it, like the average age of information.
4. Analyze the data to create visualizations similar to those found in the paper.

**Appendix:**

1. All code for this project is in <https://github.com/Johnmancini30/CORE_Research>
2. The paper we are replicating is “Experimental Evaluation of the Age of Information via Emulation”. It is the in github repo
3. The average age and average peak age formula come from the paper “Real-Time Status: How Often Should One Update?” *Sanjit Kaul, Roy Yates, Marco Gruteser*
4. The MGEN user guide is in the github repo.