Measuring Starlink Latency from RIPE Atlas

CHRIS GU, CHRIS YUAN, ETHAN STANLEY, DEVAN FLORES, and CHARLIE MUELLER

The Starlink Network is a collection of satellites in orbit around Earth. The satellites orbit much lower than different artificial satellites and communicate with user terminals on the ground. Our job was to use a set of RIPE Atlas probes connected to the Starlink Network to run measurements on the network. More specifically, we needed to measure latency from user terminals to a destination on the internet across the world.

ACM Reference Format:
Chris Gu, Chris Yuan, Ethan Stanley, Devan Flores, and Charlie Mueller. 2023. Measuring Starlink Latency from RIPE Atlas. 1, 1 (May 2023), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In this project, we took measurements over the Starlink satellite network. The main starlink network is comprised of a series of sattelites that hover a few hundreds of kilometers over the earth. Starlink also consists of ground stations on Earth, which serve as a connection to the Internet's fiber infrastructure. In this way different user terminals can send data to each other via the satellites. In this project we measured the latency from user terminals to a site on the Internet.

The company RIPE Atlas provides a set of probes that can run and collect measurements on a given network. These probes are given to volunteers who connect a probe to their network. We interfaced and measured the set of RIPE probes connected to the Starlink network through a Starlink user terminal. Each probe has an ID number and some other helpful information such as the country it is located in. The measurement we are interested in is ping. Ping is the time for a user to send packets to a destination, and then the time needed to receive a response from the destination. Using ping we can get a sense of the latency of a network since ping measures RTT (round trip time). Our goal for this project was to measure the RTT time from Starlink user terminals to google.com over the course of a day using a set of RIPE probes.

2 CREDIT CALCULATION

Each measurement on the RIPE Atlas platform costs credits. Cost for ping measurements is calculated based on the total number of measurements and the amount and size of packets per ping.

$$\text{Unit Cost} = N * (int(S/1500) + 1) \tag{1}$$

Where N is the number of packets and S is the size in octets. After doing some reading on the RIPE Atlas site we decided to send two packets for each ping measurement, giving us two RTT's for each measurement. We believed that by averaging these results we would get less variable data since after testing some of the pings we saw high variance in the RTT for different packets of the same ping. The project required us to send a ping from a probe every 15 minutes for the entirety of 24 hours. This gives us 96 pings per probe, and 192 packets per probe over the day. As will be explained below, we ended up finding 56 RIPE probes connected to the Starlink network. 192 * 56

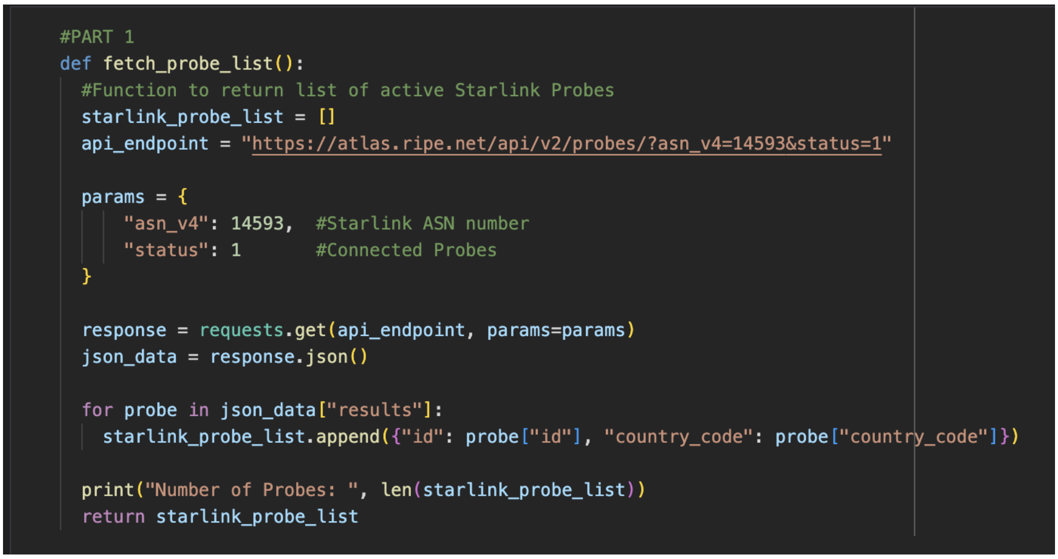
Authors' address: Chris Gu; Chris Yuan; Ethan Stanley; Devan Flores; Charlie Mueller.

XXXX-XXXX/2023/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

gave us a final total of 10,752 packets, or 10,752 credits that we needed from the course staff to get the data.

3 FINDING PROBES

As explained above, to take these measurements we needed to find a all the RIPE Atlas probes that are connected to Starlink user terminals. Each probe has an ID number as specified by the RIPE Atlas API, which also offered a handful of functions helpful in finding probes and creating measurements. One particular API function we used was GET /api/v2/probes/, which returns a list of RIPE Atlas probes. In the API call we added in a parameter asnv4=14593, which causes the API to only return probes connected to the system with ASN = 14593. We found out that ASN 14593 is corresponds to the Starlink network through an online search. This enabled to us to use that API call to get a list of RIPE Atlas probes connected to the Starlink network. We then also used a second parameter in the API call, status=1, which caused the API to only return probes that are active. We then called the API, and turned the response into JSON. This allowed us to iterate through the returned list of probes and create a new list containing only the probe ID and country codes for each probe. This gave us what we needed to begin measuring: a list of all probe IDs corresponding to RIPE Atlas probes that are active and connected to the Starlink network. After running everything, we had a list of 56 RIPE probes. Below is a snippet of the code we wrote to find the needed probes.



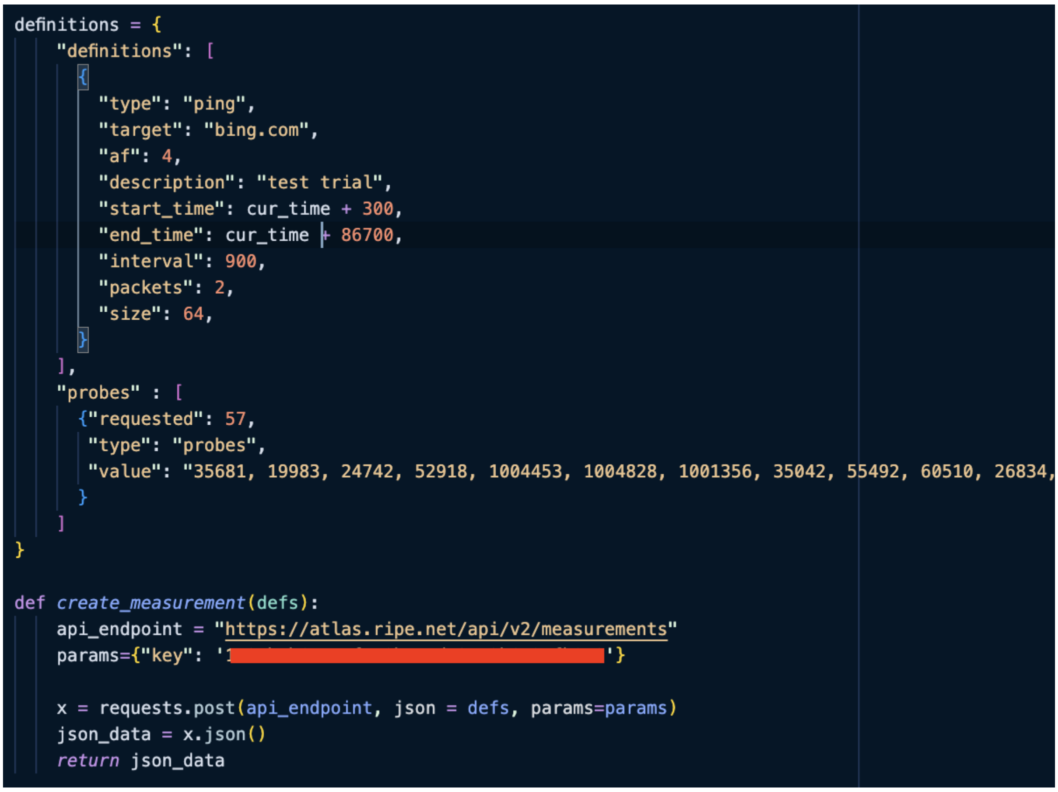
```python
#PART 1
def fetch_probe_list():
    #Function to return list of active Starlink Probes
    starlink_probe_list = []
    api_endpoint = "https://atlas.ripe.net/api/v2/probes/?asn_v4=14593&status=1"

    params = {
        "asn_v4": 14593,  #Starlink ASN number
        "status": 1       #Connected Probes
    }

    response = requests.get(api_endpoint, params=params)
    json_data = response.json()

    for probe in json_data["results"]:
        starlink_probe_list.append({"id": probe["id"], "country_code": probe["country_code"]})

    print("Number of Probes: ", len(starlink_probe_list))
    return starlink_probe_list
```

Fig. 1. Code used to get a list of RIPE Atlas Probes connected to Starlink.

4 CREATING MEASUREMENTS AND GETTING DATA

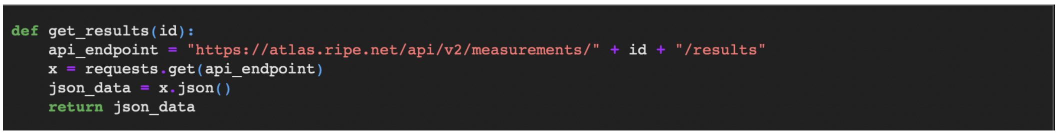
To create a measurement, we can send a post request to https://atlas.ripe.net/api/v2/measurements with the specifications of our measurement, using the Starlink probes retrieved from GET /api/v2/probes/. Finally, we sent this POST request with our an API key generated by our RIPE Atlas account.

Retrieving results just requires the measurement ID which is returned as a result after posting a new measurement.



```
definitions = {
    "definitions": [
        {
            "type": "ping",
            "target": "bing.com",
            "af": 4,
            "description": "test trial",
            "start_time": cur_time + 300,
            "end_time": cur_time + 86700,
            "interval": 900,
            "packets": 2,
            "size": 64,
        }
    ],
    "probes" : [
        {"requested": 57,
         "type": "probes",
         "value": "35681, 19983, 24742, 52918, 1004453, 1004828, 1001356, 35042, 55492, 60510, 26834,
        }
    ]
}

def create_measurement(defs):
    api_endpoint = "https://atlas.ripe.net/api/v2/measurements"
    params={"key": '[redacted]'}

    x = requests.post(api_endpoint, json = defs, params=params)
    json_data = x.json()
    return json_data
```

Fig. 2. Code used to create a ping measurement



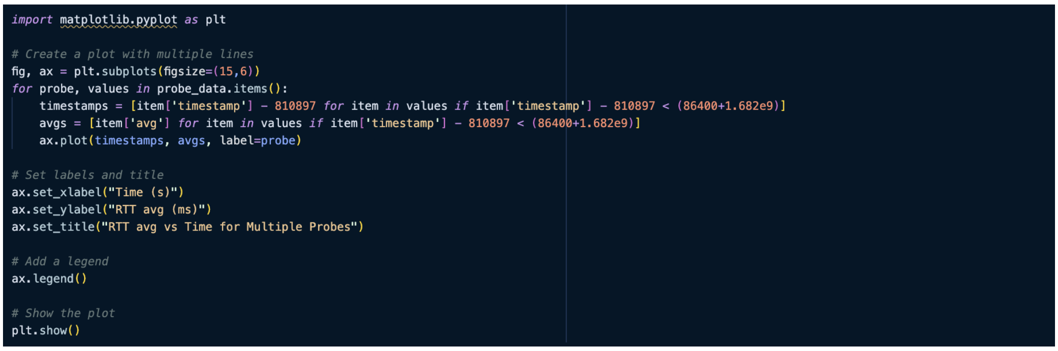
```
def get_results(id):
    api_endpoint = "https://atlas.ripe.net/api/v2/measurements/" + id + "/results"
    x = requests.get(api_endpoint)
    json_data = x.json()
    return json_data
```

Fig. 3. Code used to retrieve a measurement.

5 PROCESSING DATA AND PLOTTING

We processed the data to remove outliers using standard deviation of response times based on the probe. We plotted the data based on location of the probe as well as probe ID. Because Starlink heavily depends on ground station, plotting based on location will give more insightful results. Probes that are in the same approximate area also experience similar weather conditions and make connections to a close if not the same satellite. These factors can affect the internet connection and therefore should be account for in plotting.

We used the python library matplotlib to generate plots. Timestamps of our data is subtracted by 810897 to zero the charts. All plots are round-trip time average over UTC time.



```python
import matplotlib.pyplot as plt

# Create a plot with multiple lines
fig, ax = plt.subplots(figsize=(15,6))
for probe, values in probe_data.items():
    timestamps = [item['timestamp'] - 810897 for item in values if item['timestamp'] - 810897 < (86400+1.682e9)]
    avgs = [item['avg'] for item in values if item['timestamp'] - 810897 < (86400+1.682e9)]
    ax.plot(timestamps, avgs, label=probe)

# Set labels and title
ax.set_xlabel("Time (s)")
ax.set_ylabel("RTT avg (ms)")
ax.set_title("RTT avg vs Time for Multiple Probes")

# Add a legend
ax.legend()

# Show the plot
plt.show()
```

Fig. 4. Code for plot generation

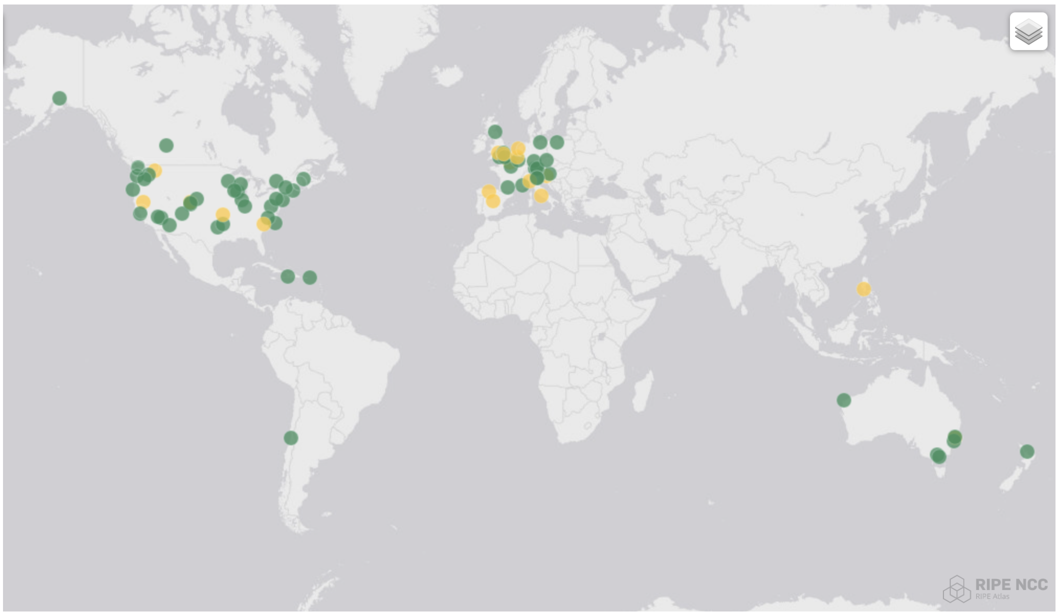


Fig. 5. Starlink probes overlayed onto World Map

6 ANALYZING RESULTS

Figures 7 through 9 display the average round trip time (RTT) ping of various probes over the course of our 24-hour collection period. We will use these graphs to see how distance to ground stations effect ping.

In Figure 7 we compare the RTT of Alaskan probe 61113 and Chilean Probe 26834. The Alaskan probe experiences large ping with high variance across the board, while the Chilean probe has consistently low RTT. According to figure 5, there is a ground station located near the Chilean probe while the Alaskan probe is a good distance from the nearest station. This ground station distance discrepancy likely accounts for the large ping in Alaska.

Figure 8 contains the RTT averages for all RIPE Atlas probes located in Australia. Probe 52918 consistently performs the worst across the 24-hour period. Looking at the ground stations in figure

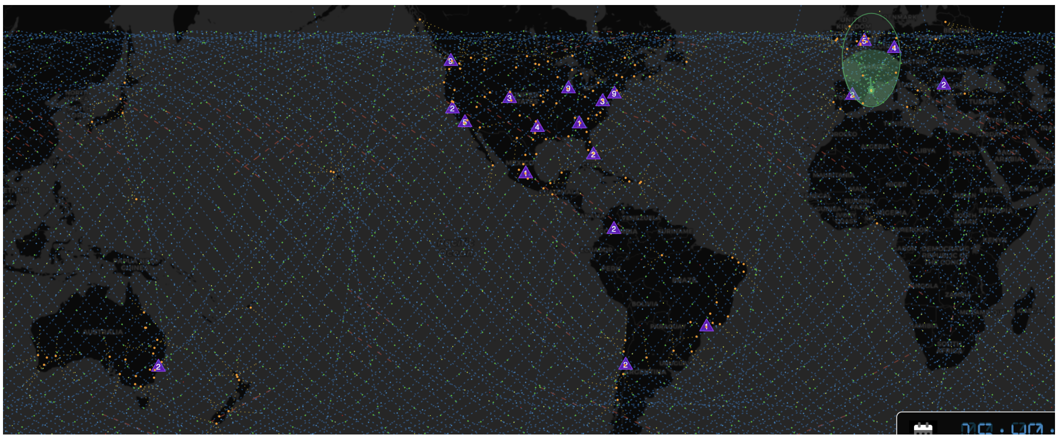


Fig. 6. Purple Dots represent ground stations

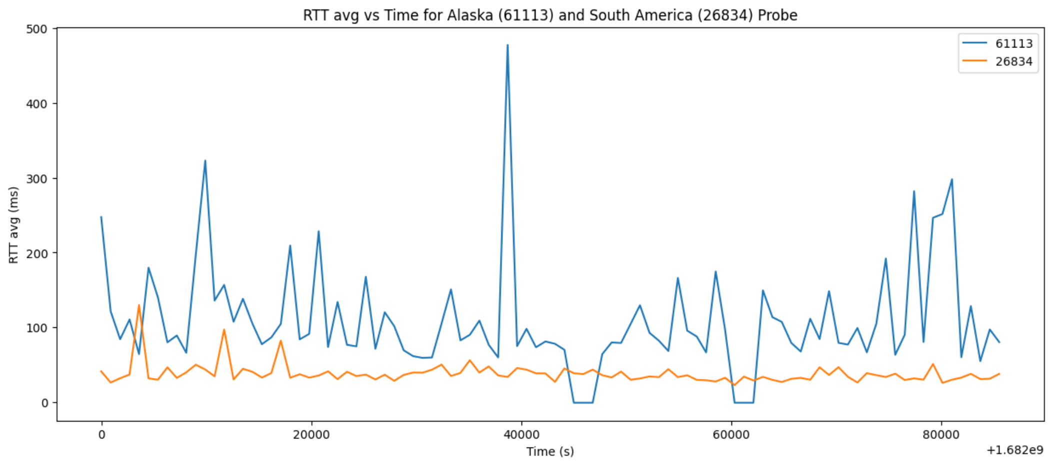


Fig. 7. RTT Avg over Time for Alaska (61113) and South America (26834) probes

5, we see that Western Australian probe 52918, is far from the nearest ground station in Eastern Australia. This could explain the high RTT numbers.

In Figure 9, we plot the RTT averages of American RIPE Atlas probes, by urban density. We do not see a clear disparity between the urban and rural groups, although there are a couple rural outlier probes with highly variant RTT. Figure 5 shows that the Starlink ground stations are equally dispersed across the continental United States. This explains why rural and urban areas see similar RTT performances.

Distance to ground station is a very important factor when it comes to Round Trip Time ping.

7 POTENTIAL IMPLICATIONS

Ultimately, our research implies that Starlink is meant as an alternative to standard broadband connections, instead of a replacement. In most scenarios, Starlink performs optimally and gives users a stable and reliable connection. However, there are several outliers that show significant issues.

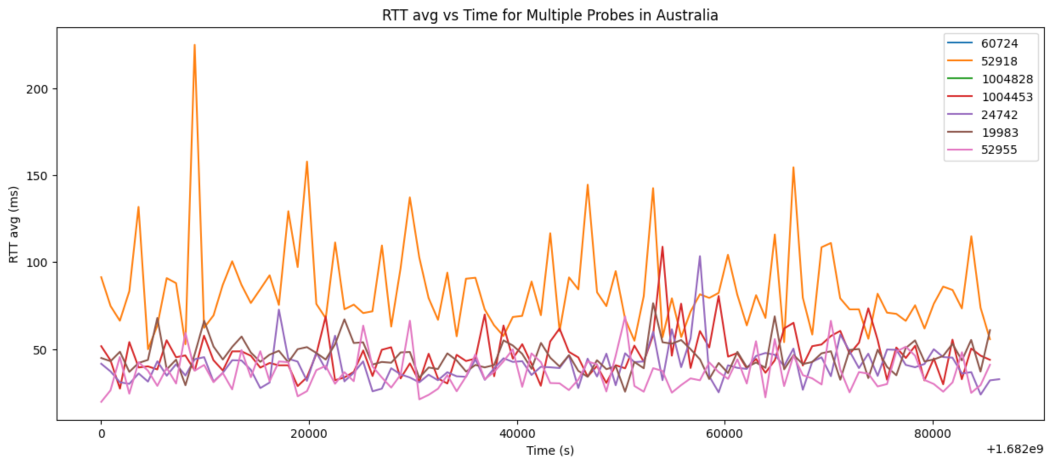


Fig. 8. RTT Avg over Time for Australian probes

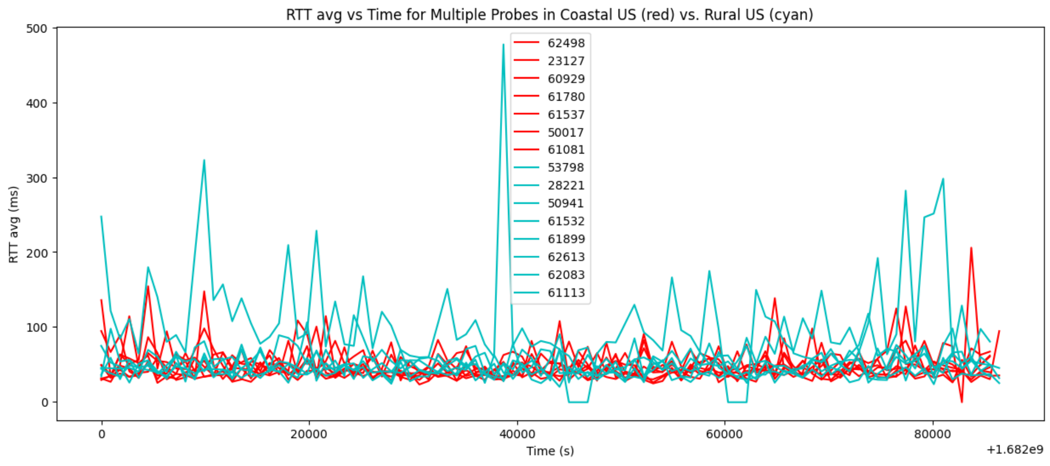


Fig. 9. RTT Avg over Time for Coastal vs Rural America probes

We believe this is due to the very nature of satellite connections and the various interruptions that can happen when transmitting over satellite. In practice, many Starlink customers complain about the unreliable nature of the technology. Everyday objects such as "trees, birds, and buildings" can cause major interruptions in signal [1]. Starlink themselves say that the technology was designed for areas "where connectivity has been unreliable or completely unavailable" [2]. Although Starlink is a promising technology, we are very far away from completely replacing broadband.

REFERENCES

[1] Nilay Patel. 2021. Starlink review: broadband dreams fall to Earth. (14 May 2021). Retrieved May 9, 2023 from https://www.theverge.com/22435030/starlink-satellite-internet-spacex-review
[2] Starlink. 2023. *Roam*. Retrieved May 9, 2023 from https://www.starlink.com/roam