Starlink Performance Comparison

November 9, 2021

This document was created by the research group ROADMAP-5G at the Carinthia University of Applied Sciences (CUAS) in October 2021. It is an extension to a previously published report by the same group and focuses on the changes that happened since then. The used Starlink dish and modem were again kindly provided by Stereo Media.

Christoph Uran (c.uran@fh-kaernten.at), Kurt Horvath (k.horvath@fh-kaernten.at), Helmut Wöllik (h.woellik@fh-kaernten.at), Valentin Egger (v.egger@fh-kaernten.at)

1 Measurement Setup

The setup was exactly the same as it was in the previous report from September 2021. Please refer to this document for more details.

2 Library Import

The computations done here need different Python libraries, which are imported first. Also, the plotting libraries have to be configured accordingly. The needed libraries are (in alphabetical order):

- `datetime`: for handling dates and times
- `helpers`: a self-written helper library for data preprocessing
- `IPython.display`: library for correctly displaying Pandas DataFrames
- `json`: for handling JSON (JavaScript Object Notation) data
- `matplotlib.pyplot`: for plotting graphs
- `pandas`: for handling tabular data
- `re`: a library for using regular expressions
- `seaborn`: an advanced plotting library
- `statistics`: a library for statistical operations on data
- `time`: for handling time object

[1]:
```
# library for handling JSON
import json
# plotting libraries
import matplotlib.pyplot as plt
import seaborn as sns
# self-written helper library for data preprocessing
import helpers
# library for statistical operations on data
import statistics
```

```
# libraries for handling dates and times
from datetime import datetime, timedelta
import time
# libraries for handling regular expressions
import re
# library for handling tabular data
import pandas as pd
# library for correctly displaying Pandas DataFrames
from IPython.display import display, HTML
# configuration of the size of the plotted figures
plt.rcParams['figure.figsize'] = [12, 6]
```

3 Data Preparation

For this report, the data from June 2021 as well as the data from September 2021 have to be imported. This includes the data generated at `iperf3` tests as well as `ping` tests. All of this data is being stored in appropriate data structures.

[2]:
```
# read, load, and extract data from all the iperf files

# June, Download, Client-Side
file = open('../2021-06/starlink_iperf_tests/iperf-client-download-20210614.
 ↪log')
logdata = json.load(file)
file.close()
jun_dl_timestamps_clientside, jun_dl_throughputs_clientside,␣
 ↪jun_dl_retransmits_clientside, jun_dl_cwnd_clientside = helpers.
 ↪extract_ts_and_throughput(logdata, unit='M')

# June, Upload, Server-Side
file = open('../2021-06/starlink_iperf_tests/
 ↪iperf3-server-test-from-client-to-server-20210614.log')
logdata = json.load(file)
file.close()
jun_ul_timestamps_serverside, jun_ul_throughputs_serverside,␣
 ↪jun_ul_retransmits_serverside, jun_ul_cwnd_serverside = helpers.
 ↪extract_ts_and_throughput(logdata, unit='M')

# September, Download, Client-Side
file = open('starlink_iperf_tests/iperf-client-download-20211001.log')
logdata = json.load(file)
file.close()
sep_dl_timestamps_clientside, sep_dl_throughputs_clientside,␣
 ↪sep_dl_retransmits_clientside, sep_dl_cwnd_clientside = helpers.
 ↪extract_ts_and_throughput(logdata, unit='M')
```

```
# September, Upload, Server-Side
file = open('starlink_iperf_tests/iperf-client-upload-20210929.log')
logdata = json.load(file)
file.close()
sep_ul_timestamps_serverside, sep_ul_throughputs_serverside,␣
 ↪sep_ul_retransmits_serverside, sep_ul_cwnd_serverside = helpers.
 ↪extract_ts_and_throughput(logdata, unit='M')

# read, load, and extract data from all the ping files

# June
jun_all_timestamps, jun_all_status, jun_all_latencies, jun_unsuccessful_ts,␣
 ↪jun_colors, jun_working_windows, jun_not_working_windows,␣
 ↪jun_latency_after_not_working = helpers.extract_latency_values('../2021-06/
 ↪starlinktest_with_load_20210602_to_20210609.log')
# September
sep_all_timestamps, sep_all_status, sep_all_latencies, sep_unsuccessful_ts,␣
 ↪sep_colors, sep_working_windows, sep_not_working_windows,␣
 ↪sep_latency_after_not_working = helpers.extract_latency_values('september21/
 ↪starlinktest_20092025_1.txt')
```

```
Skipping line: PING 194.232.104.3 (194.232.104.3) 56(84) bytes of data.

Skipping line: PING 194.232.104.3 (194.232.104.3) 56(84) bytes of data.
```

As can be seen from the above output, only the first line of the ping tests of June and September respectively have been skipped because they don't contain any important data.

4 Throughput

This section compares the achieved throughputs in upload and download respectively between June and September. It has been tested using a server hosted in the Frankfurt datacenter of the Google Cloud.

[3]:
```
# get data into dictionary
tp = {'Month': ['June', 'June', 'September', 'September'],
      'Direction': ['Up', 'Down', 'Up', 'Down'],
      'Average': [statistics.mean(jun_ul_throughputs_serverside), statistics.
 ↪mean(jun_dl_throughputs_clientside),
                  statistics.mean(sep_ul_throughputs_serverside), statistics.
 ↪mean(sep_dl_throughputs_clientside)],
      'Median': [statistics.median(jun_ul_throughputs_serverside), statistics.
 ↪median(jun_dl_throughputs_clientside),
                 statistics.median(sep_ul_throughputs_serverside), statistics.
 ↪median(sep_dl_throughputs_clientside)],
```

```
      'Maximum': [max(jun_ul_throughputs_serverside),␣
 ↪max(jun_dl_throughputs_clientside),
                  max(sep_ul_throughputs_serverside),␣
 ↪max(sep_dl_throughputs_clientside)]}
# create DataFrame out of dictionary
tp_df = pd.DataFrame(data=tp)
# format the numbers
tp_df['Average'] = tp_df['Average'].map('{:,.0f} Mbit/s'.format)
tp_df['Median'] = tp_df['Median'].map('{:,.0f} Mbit/s'.format)
tp_df['Maximum'] = tp_df['Maximum'].map('{:,.0f} Mbit/s'.format)
# print the DataFrame
display(HTML(tp_df.to_html(index=False)))
```

```
<IPython.core.display.HTML object>
```

Table 1: Comparison of throughputs achieved in June and September.

It can be seen from Table 1 that the maximum achievable throughput has been reduced by about 15% from June to September. Furthermore, the median of the achieved throughputs has even been reduced by approximately 25%. This suggests that there have been some systematic changes inside the Starlink setup.

[16]:
```
# plot definitions
fig, axes = plt.subplots(1, 2, sharex=True, sharey=True)
fig.suptitle('Comparison of Download Throughputs')
axes[0].set_title('June')
axes[1].set_title('September')

# defining start and end time, getting the respective indices and processing␣
 ↪the data
jun_start_time = '2021-06-14 08:00:00'
jun_end_time = '2021-06-14 18:00:00'
jun_start_index, jun_end_index = helpers.
 ↪get_indices_by_time(jun_dl_timestamps_clientside, jun_start_time,␣
 ↪jun_end_time)
jun_boxes, jun_times, jun_unsuccessfuls = helpers.
 ↪generate_latency_boxes(jun_dl_timestamps_clientside,␣
 ↪jun_dl_throughputs_clientside, jun_start_index, jun_end_index)

# plotting the data
sns.set(context='notebook', style='whitegrid')
sns.boxplot(ax=axes[0], data=jun_boxes, showfliers=False)
axes[0].set(xlabel='Hour', ylabel='Throughput (MBit/s)')

# defining start and end time, getting the respective indices and processing␣
 ↪the data
sep_start_time = '2021-10-02 08:00:00'
sep_end_time = '2021-10-02 16:00:00'
```

```
sep_start_index, sep_end_index = helpers.
 ↪get_indices_by_time(sep_dl_timestamps_clientside, sep_start_time,␣
 ↪sep_end_time)
sep_boxes, sep_times, sep_unsuccessfuls = helpers.
 ↪generate_latency_boxes(sep_dl_timestamps_clientside,␣
 ↪sep_dl_throughputs_clientside, sep_start_index, sep_end_index)

# plotting the data
#sns.set(context='notebook', style='whitegrid')
sns.boxplot(ax=axes[1], data=sep_boxes, showfliers=False)
axes[1].set(xlabel='Hour', ylabel='Throughput (MBit/s)')
_ = axes[1].set(xlabel='Hour')
```

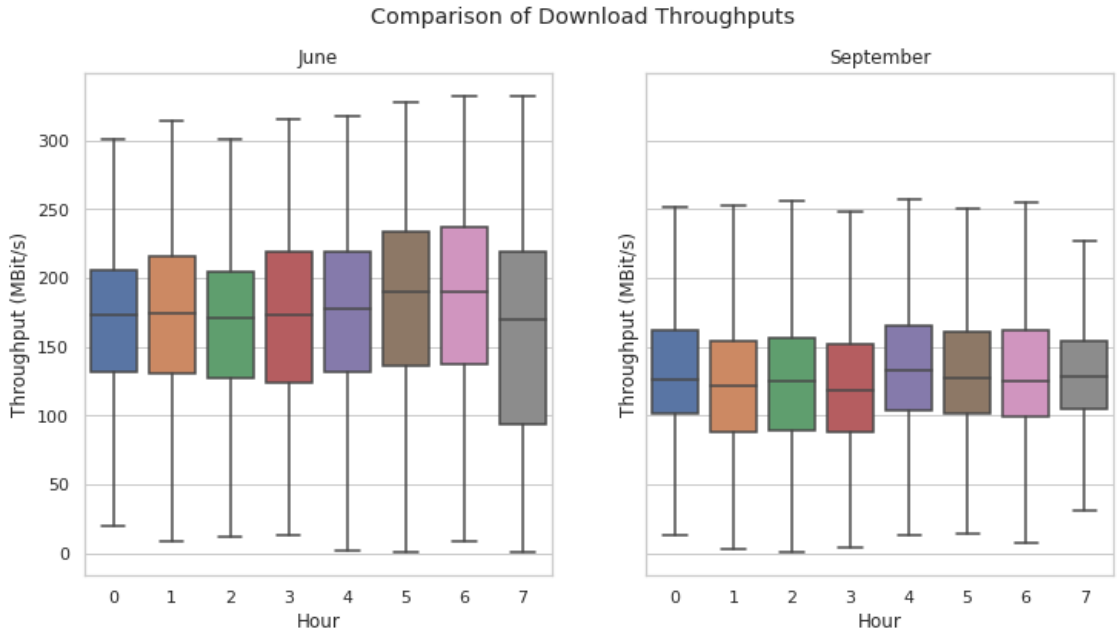

Figure 1: Comparison of achieved download throughputs.

[17]:
```
# plot definitions
fig, axes = plt.subplots(1, 2, sharex=True, sharey=True)
fig.suptitle('Comparison of Upload Throughputs')
axes[0].set_title('June')
axes[1].set_title('September')

# defining start and end time, getting the respective indices and processing␣
 ↪the data
jun_start_time = '2021-06-14 16:00:00'
jun_end_time = '2021-06-15 06:00:00'
```

```
jun_start_index, jun_end_index = helpers.
 ↪get_indices_by_time(jun_ul_timestamps_serverside, jun_start_time,␣
 ↪jun_end_time)
jun_boxes, jun_times, jun_unsuccessfuls = helpers.
 ↪generate_latency_boxes(jun_ul_timestamps_serverside,␣
 ↪jun_ul_throughputs_serverside, jun_start_index, jun_end_index)

# plotting the data
sns.set(context='notebook', style='whitegrid')
sns.boxplot(ax=axes[0], data=jun_boxes, showfliers=False)
axes[0].set(xlabel='Hour', ylabel='Throughput (MBit/s)')

# defining start and end time, getting the respective indices and processing␣
 ↪the data
sep_start_time = '2021-09-30 16:00:00'
sep_end_time = '2021-10-01 06:00:00'
sep_start_index, sep_end_index = helpers.
 ↪get_indices_by_time(sep_ul_timestamps_serverside, sep_start_time,␣
 ↪sep_end_time)
sep_boxes, sep_times, sep_unsuccessfuls = helpers.
 ↪generate_latency_boxes(sep_ul_timestamps_serverside,␣
 ↪sep_ul_throughputs_serverside, sep_start_index, sep_end_index)

# plotting the data
sns.boxplot(ax=axes[1], data=sep_boxes, showfliers=False)
axes[1].set(xlabel='Hour', ylabel='Throughput (MBit/s)')
_ = axes[1].set(xlabel='Hour')
```

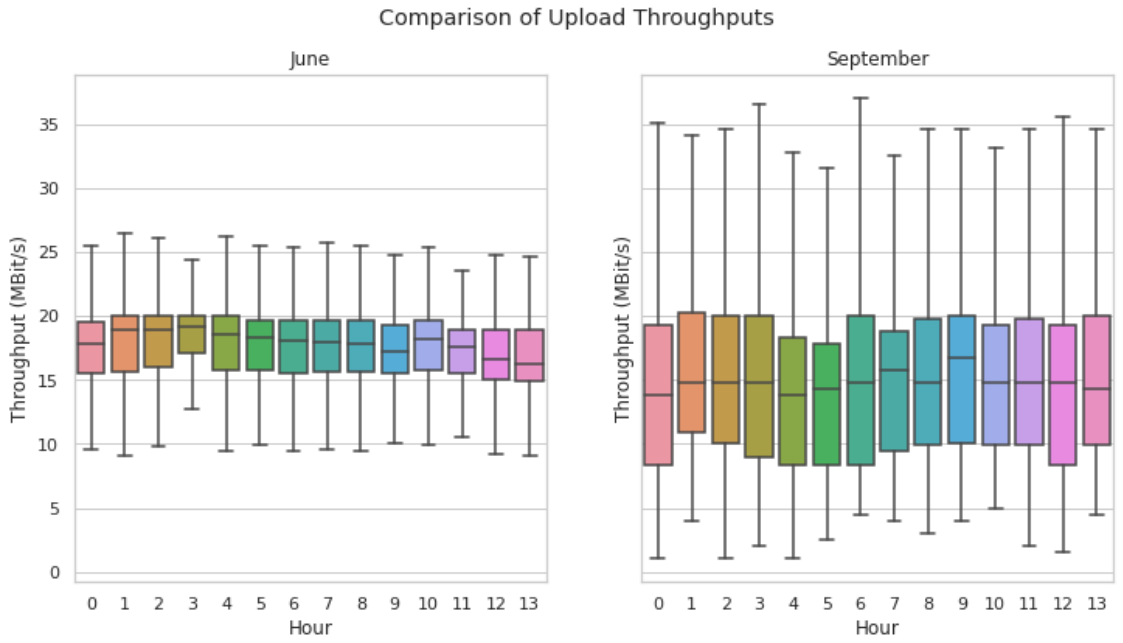

Figure 2: Comparison of achieved upload throughputs.

All of the plots in Figures 1 and 2 show an inner consistency, i.e. there are now signs of faults during the measurements which might influence the interpreted results. It is therefore interesting to see that the spread of the upload throughput in September has doubled compared to that in June. In contrast, the spread of the download throughput has decreased in September, albeit not that significantly. The reason for that could not be found with the available data.

4.1 Interpretation

As can be seen from the table and the plots above, the achieved throughputs have gone down significantly in the three months between the measurements. This could be due to several reasons:

- The number of Starlink users has likely increased in the time from June to September. Due to the suspected higher number of simultaneous users, the available data rate is lower for an individual user.
- There might have been some optimizations inside the system that lower the throughput in order to achieve higher reliability. This would fit with the observations described later in this document.
- It could also be some other event (e.g. other tests or massive load caused by other Starlink users at the same time as our tests) that influenced our measurements.

The most likely reason is that the system has been optimized to achieve higher reliability.

5 Latency

This section compares the achieved latency to a web server of an Austrian TV broadcaster (orf.at) via Starlink between June and September. This server is located in Vienna.

[21]:
```
# get the data
jun_all_working_latencies = [l for l in jun_all_latencies if l != 0]
sep_all_working_latencies = [l for l in sep_all_latencies if l != 0]
# get data into dictionary
lat = {'Month': ['June', 'September'],
       'Minimum': [min(jun_all_working_latencies),␣
 ↪min(sep_all_working_latencies)],
       'Average': [statistics.mean(jun_all_working_latencies), statistics.
 ↪mean(sep_all_working_latencies)],
       'Median': [statistics.median(jun_all_working_latencies), statistics.
 ↪median(sep_all_working_latencies)],
       'Maximum': [max(jun_all_working_latencies),␣
 ↪max(sep_all_working_latencies)]}
# create DataFrame out of dictionary
lat_df = pd.DataFrame(data=lat)
# format the numbers
lat_df['Minimum'] = lat_df['Minimum'].map('{:,.0f} ms'.format)
```

```
lat_df['Average'] = lat_df['Average'].map('{:,.0f} ms'.format)
lat_df['Median'] = lat_df['Median'].map('{:,.0f} ms'.format)
lat_df['Maximum'] = lat_df['Maximum'].map('{:,.0f} ms'.format)
# print the DataFrame
display(HTML(lat_df.to_html(index=False)))
```

```
<IPython.core.display.HTML object>
```

Table 2: Comparison of latencies achieved in June and September.

Table 2 shows that, while the minimum and the maximum measured latencies stayed within similar magnitudes, the average and the median latencies measured in September increased by about 50% compared to the ones measured in June. This is obviously a significant change.

[15]:
```
lat_cap = 150
plt.hist(jun_all_working_latencies, bins=lat_cap*100, density=True,␣
 ↪histtype='step', cumulative=True, label='June', linewidth=2)
plt.hist(sep_all_working_latencies, bins=lat_cap*100, density=True,␣
 ↪histtype='step', cumulative=True, label='September', linewidth=2)
plt.title('Cumulative Histogram of the Latencies capped at %s ms' % lat_cap)
plt.xlabel('Latency (ms)')
plt.ylabel('Frequency')
plt.xlim([40, lat_cap])
plt.ylim([0.5, 1.03])
plt.grid(True)
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)
plt.legend(loc='lower right')
plt.show()
```

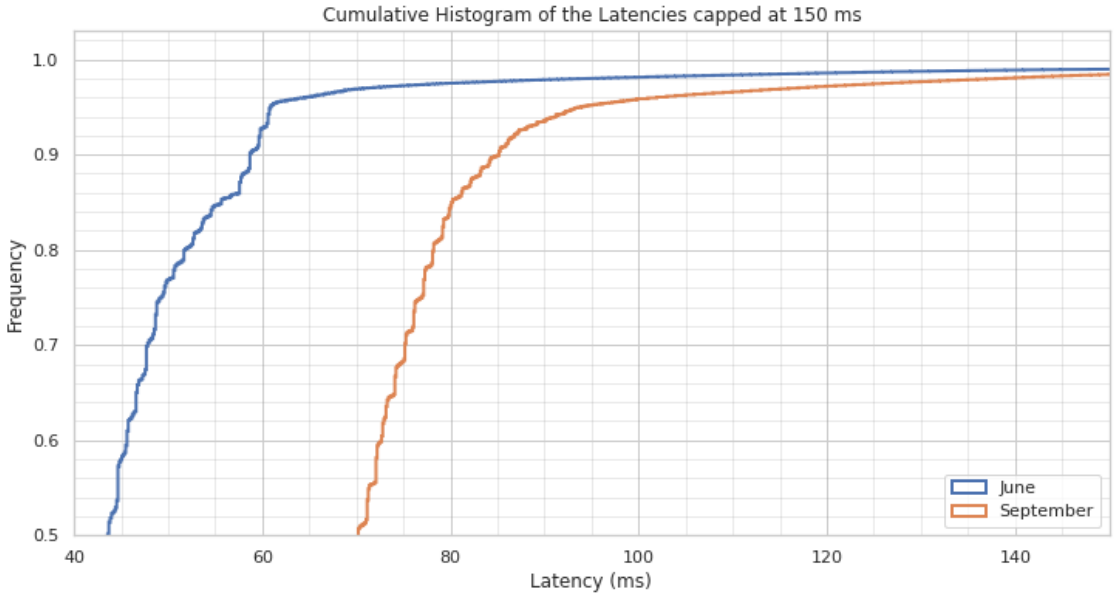

Figure 3: Histogram of achieved latencies.

The result from Table 2 can be even better seen with the help of Figure 3. It shows that nearly 98% of the measured latencies in June were under 80 ms while in September, only 85% of the latencies stayed under that threshold. A similar relation can be observed throughout the plot in Figure 3.

5.1 Interpretation

Similar to the throughput measurements, all signs indicate that there have been some systematic changes in Starlink that increase reliability (see below) at the expense of throughput and latency.

6 Reliability

As already mentioned in the paragraphs above, the achievable throughputs and latencies seem to have worsened since the previous tests in June. The following analysis will show, that this correlates with a simultaneous significant improvement in reliability.

6.1 General Information

[18]:
```
# crunching the numbers
# June
jun_all_pings = len(jun_all_timestamps)
jun_failed_pings = len(jun_unsuccessful_ts)
jun_time_working = sum(jun_working_windows, timedelta())
jun_time_not_working = sum(jun_not_working_windows, timedelta())
# September
sep_all_pings = len(sep_all_timestamps)
sep_failed_pings = len(sep_unsuccessful_ts)
sep_time_working = sum(sep_working_windows, timedelta())
sep_time_not_working = sum(sep_not_working_windows, timedelta())

# get data into dictionary
ping = {'Month': ['June', 'September'],
#       'Start': [jun_all_timestamps[0], sep_all_timestamps[0]],
#       'End': [jun_all_timestamps[-1], sep_all_timestamps[-1]],
       'Pings': [jun_all_pings, sep_all_pings],
       'Fails': [jun_failed_pings, sep_failed_pings],
       'Loss': [(jun_failed_pings / jun_all_pings) * 100, (sep_failed_pings /␣
 ↪sep_all_pings) * 100],
       'Working': [jun_time_working, sep_time_working],
       'Not Working': [jun_time_not_working, sep_time_not_working],
       'Reliability': [100-((jun_time_not_working / (jun_time_working +␣
 ↪jun_time_not_working)) * 100), 100-((sep_time_not_working /␣
 ↪(sep_time_working + sep_time_not_working)) * 100)]}

# create DataFrame out of dictionary
```

```
ping_df = pd.DataFrame(data=ping)

# format the numbers
#ping_df['Start'] = ping_df['Start'].dt.strftime('%m-%d %H:%M')
#ping_df['End'] = ping_df['End'].dt.strftime('%m-%d %H:%M')
ping_df['Working'] = ping_df['Working'].round('1s')
ping_df['Not Working'] = ping_df['Not Working'].round('1s')
ping_df['Loss'] = ping_df['Loss'].map('{:,.1f}%'.format)
ping_df['Reliability'] = ping_df['Reliability'].map('{:,.1f}%'.format)

display(HTML(ping_df.to_html(index=False)))
```

```
<IPython.core.display.HTML object>
```

Table 3: Analysis of the reliabilities experienced in June and September.

Table 3 shows that there have been by far fewer lost pings in September than there have been in June. The *Reliability* (*1 - time_not_working / time_total*) has increased from 97.6% to 99.8%, which is a massive improvement.

6.2 Distribution of Outage Durations

[9]:
```
condense_to = 31
jun_sec_bins_not_working_windows = [td.total_seconds() if td.total_seconds() <␣
 ↪condense_to else condense_to for td in jun_not_working_windows]
sep_sec_bins_not_working_windows = [td.total_seconds() if td.total_seconds() <␣
 ↪condense_to else condense_to for td in sep_not_working_windows]
plt.hist([jun_sec_bins_not_working_windows, sep_sec_bins_not_working_windows],␣
 ↪bins=condense_to, label=['June', 'September'], density=True)
plt.title('Histogram of the Duration of Periods without Connectivity condensed␣
 ↪to %s bins' % condense_to)
plt.xlabel('Duration (s)')
plt.ylabel('Frequency')
plt.grid(True)
plt.legend(loc='upper right')
plt.show()
```

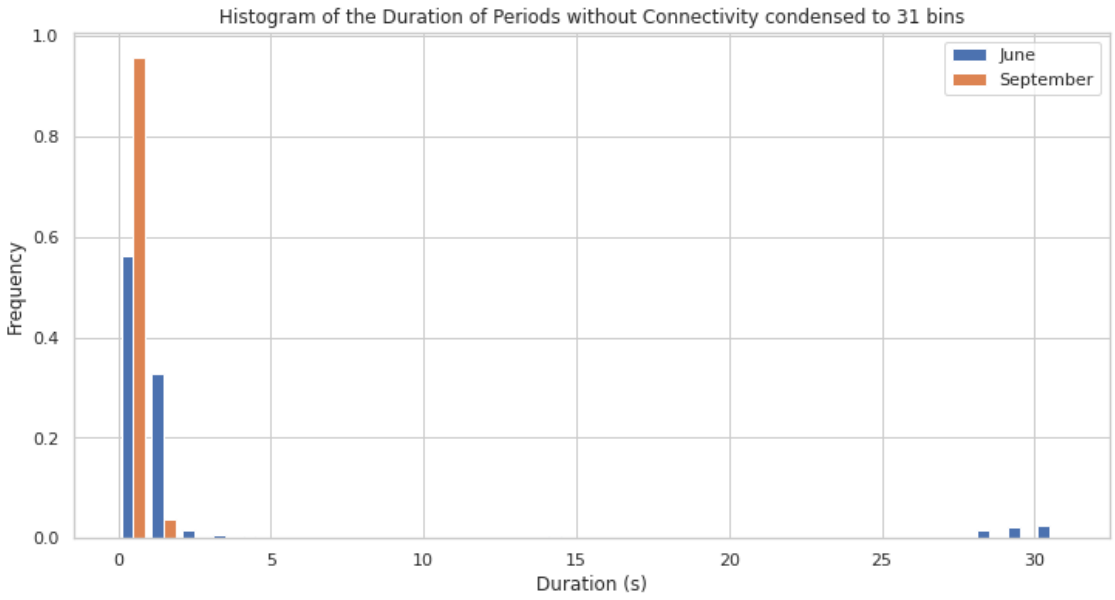

Figure 4: Histogram of the duration of periods without connectivity.

Furthermore, as Figure 4 shows, the system has been modified in a way that the few outages that still occur tend to be shorter than they have been in June. This also makes the system much more usable.

6.3 Continuously Connected Periods

[10]:
```
condense_to = 65

jun_sec_bins_working_windows = [td.total_seconds() if td.total_seconds() <=␣
 ↪condense_to-1 else condense_to for td in jun_working_windows]
sep_sec_bins_working_windows = [td.total_seconds() if td.total_seconds() <=␣
 ↪condense_to-1 else condense_to for td in sep_working_windows]
plt.hist([jun_sec_bins_working_windows, sep_sec_bins_working_windows],␣
 ↪bins=condense_to, label=['June', 'September'], density=True)
plt.title('Histogram of the Duration of continuously connected Periods␣
 ↪condensed to %s bins' % condense_to)
plt.xlabel('Duration (s)')
plt.ylabel('Frequency')
plt.grid(True)
plt.legend(loc='upper right')
plt.show()
```

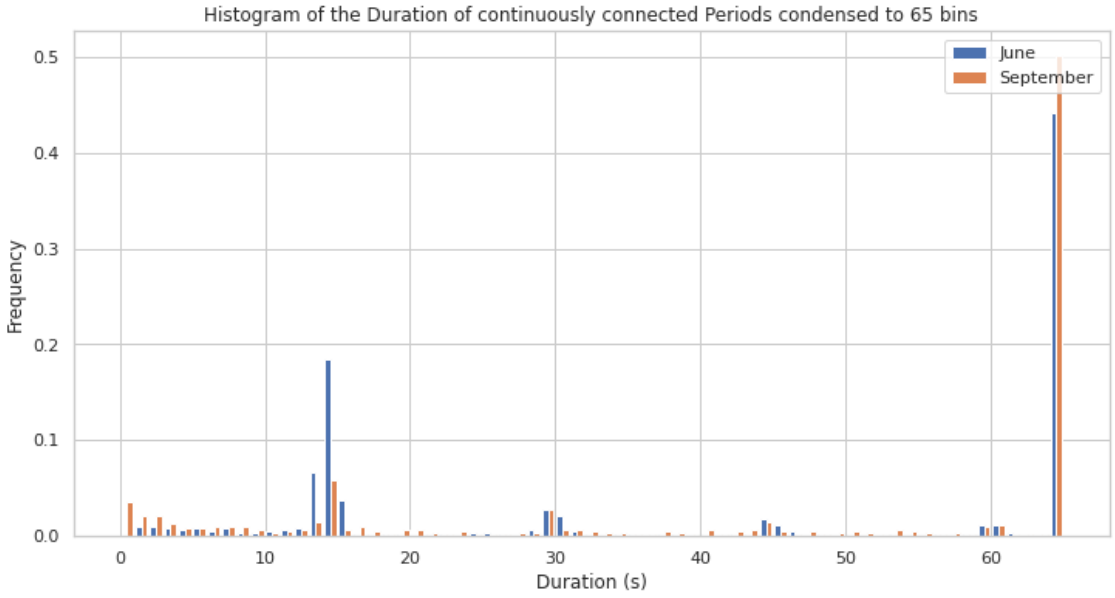

Figure 5: Histogram of the duration of continuously connected periods.

In our previous report, we discovered that there are "jumps" in the latencies, which occur *exactly* at multiples of 15 seconds (15, 30, 45, …). Due to this fact, we assumed that outages likely occur immediately after a period of connectivity that lasted a multiple of 15 seconds, which could also be verified. As can be seen in Figure 5, this phenomenon is still visible at our measurements in September, albeit to a lesser degree. As of now, we can not explain this behavior without more detailed information about the internals of Starlink.

7 Conclusion

From the gathered data and a comparison to the results of June it can be said that Starlink has improved massively in terms of reliability. However, this was done at the expense of achievable throughput and latency. Nevertheless, this is a smart decision by the developers, because an unreliable Internet connection is by far more annoying than lower throughput and higher latency. Especially for use cases that depend on a stable Internet connection, such as video streaming and virtual meetings, this is a massive improvement of the system.

In summary, the most important new findings from September are as follows.

- The throughput for download has decreased on average by 21%.
- The throughput for upload has decreased on average by 12%.
- The latency has increased on average by 32%.
- The reliability (no package loss) has increased from 97.6% to 99.8%.

It occurs that between the first and second measurement series, efforts have been made to increase the reliability of the network. It is very likely that the reduced throughput and increased latency are the consequences of this improved network reliability.