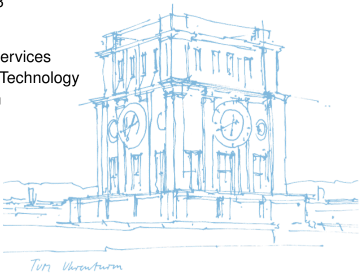
Development of a Framework for Retrieval of Parameters of the Starlink Dish

Final talk for the IDP by

Roberto Castellotti
advised by Leander Seidlitz, Johannes Zirngibl

Saturday 25th November, 2023

Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich



What is Starlink?

- Starlink Low Earth Orbiting (LEO) Satellite Constellation
- Brings Internet connection to remote areas
- More than 4000 Satellites with plans to launch more
- End users have a Dish that connect to satellites in sight
- Performance is higher compared to Geostationary Satellites (GEOSAT) based connections
- Average Latency is 35ms and Download Bandwidth is >100 MBps

Starlink 101

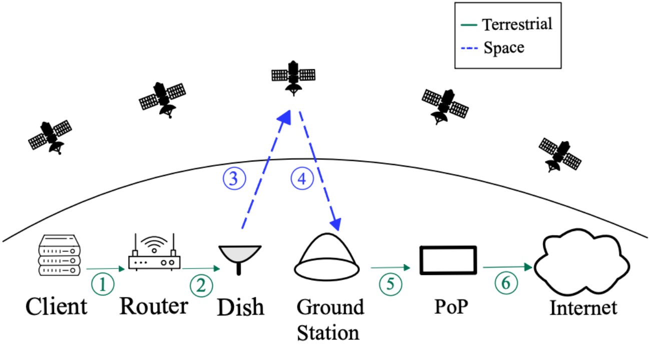


Figure 1: Starlink in a nutshell (ignoring ISL), from [1]

Our Dish

Figure 2: Our Starlink Dish

Our Work

- Understanding Routing Decisions
- Visualize Visible Satellites and patterns in their apperance
- Physical Layer Influcences on Performance
- Documenting gRPC API
- Retrieval of Obstruction Maps
- Satellite Handovers detection based on Obstruction Maps
- Correlation between Satellite Handovers and Bandwidth Drops

Understanding Routing Decisions

- retrieved ip address blocks from major cloud providers (aws,azure,oracle), as we know their position [1]
- chose 5 geographically sparse targets around the globe, i.e for aws:
  - `ap-northeast-2` Asia Pacific (Seoul)
  - `us-east-1` US East (N. Virginia)
  - `ap-south-1` Asia Pacific (Mumbai)
  - `sa-east-1` South America (São Paulo)
  - `me-south-1` Middle East (Bahrain)
- tracerouted the targets over several days

[1] the fact we know the position doesn't really the last hop will be exactly in that area (little infomation around what happens inside datacenters), but it is a good enough approximation traceroute to that geographic area

Understanding Routing Decisions

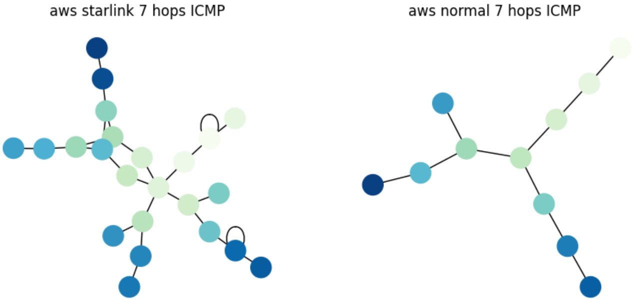


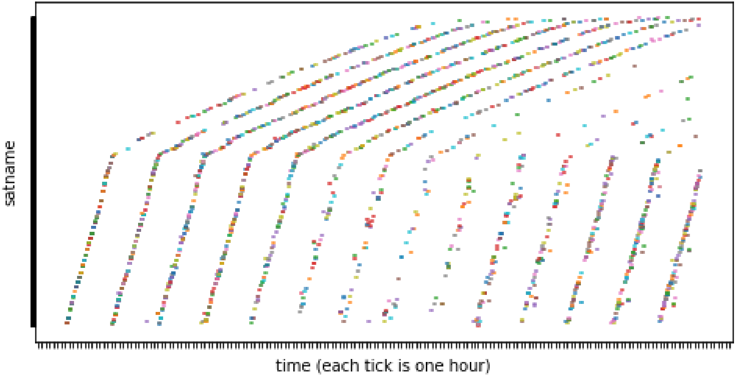
Figure 3: First 7 hops of traceroutes to 5 AWS datacenters using ICMP

Visualize Visible Satellites

- from celestrak.org we can download a list of Starlink's satellites TLEs
- A two-line element set (TLE) is a data format encoding a list of orbital elements of an Earth-orbiting object for a given point in time, the epoch.
- Using a suitable prediction formula, the state (position and velocity) at any point in the past or future can be estimated to some accuracy. (from wikipedia.org)
- wrote a Python script to calculate visible satellites[2]
- Gathered information about satellites position and proceeded to visualize how often we see a satellite

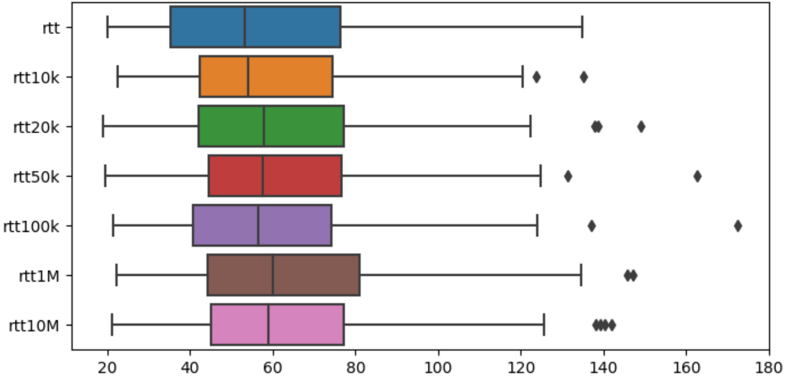
[2] https://gitlab.lrz.de/netintum/teaching/tumi8-theses/idp-castellotti/-/blob/main/common.py?ref_type=heads#L132

Visualizing Patterns in Visible Satellites



Pysical Layer Influences on Performance

- sent packets with iPerf to create traffic on the interface
- downloded Debian ISOs from 5 different mirrors (neutralize upload speed differences)



The gRPC api

- the dish exposes a gRPC api with server reflection, "runtime construction of requests without having stub information precompiled into the client." [3]
- 55 "methods" are available, most of them don't work, we have 2 categories of errors: `Uninmplemented`, `PermissionDenied` and a couple of some other specific errors
- most interesting working methods:
  - `reboot`
  - `get_status`
  - `get_obstruction_map`
- all methods: `https://gist.github.com/rcastellotti/e20630366dfeaeada6cc2680f562f6ac`

[3] https://github.com/grpc/grpc/blob/master/doc/server-reflection.md

the `dish_get_obstruction_map` Endpoint

- the `dish_get_obstruction_map` endpoint seems interesting
- an Obstruction Map captures the position where the dish has seen satellites up to that moment
- designed to provide a way to report whether the dish position is optimal
- following the approach described by Izhikevich et al. [1] we retrieve maps
- rebooting the dish clears the Obstruction Map
- polling the endpoint frequently enough allows us to detect satellite handovers
- we start by saving maps every second, we then proceed to visualize them

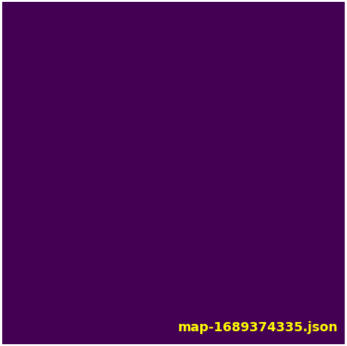
Querying the `dish_get_obstruction_map` Endpoint

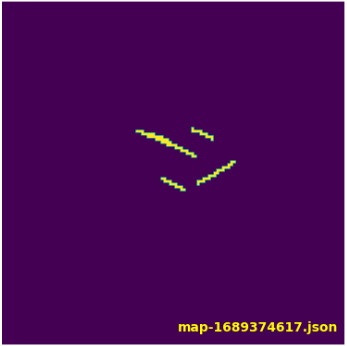

```
"apiVersion":"9",
"dishGetObstructionMap":{
    "minElevationDeg":10.0,
    "numCols":123,
    "numRows":123,
    "snr":[-1.0,-1.0,-1.0,-1.0,...,1.0,1.0,-1.0,-1.0]
}
```

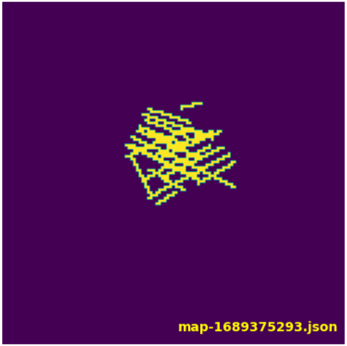
Visualizing a single Obstruction Map

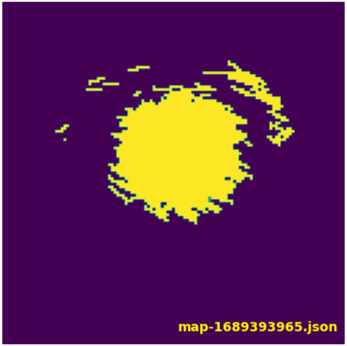
```
import json
import numpy as np
import matplotlib.pyplot as plt

map = json.load(open("1692089163.json"))
map = map["dishGetObstructionMap"]["snr"]
map = np.array(map).reshape(123, 123) # a 123*123 matrix
plt.imshow(map)
plt.show()
```


map-1689374335.json


map-1689374617.json


map-1689375293.json


map-1689393965.json

Detecting Handovers Algoritmically

- going through visualizations frame by frame is not feasible
- interpet obstruction maps as matrices
- "1" means a satellite was seen in that position, "-1" means no satellite was detected
- iterate through matrices two by two to detect handovers
- sum matrices
- check if "0" value is "near" (inside a 3*3 matrix)
  - if it is "near" no handover was performed
  - if it is in complete separate position an handover must have been performed

Obstruction Maps as Matrices (No Handover)

$$\begin{bmatrix} -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} + \begin{bmatrix} -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 2 & -2 & -2 \\ -2 & -2 & -2 & 2 & -2 \\ -2 & -2 & -2 & -2 & 0 \\ -2 & -2 & -2 & -2 & -2 \\ -2 & -2 & -2 & -2 & -2 \end{bmatrix}$$

Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich

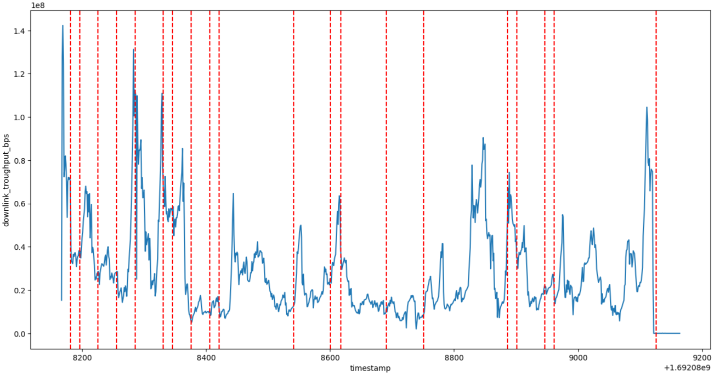
Obstruction Maps as Matrices (Handover)

$$\begin{bmatrix} -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} + \begin{bmatrix} -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 2 & -2 & -2 \\ -2 & -2 & -2 & 2 & -2 \\ -2 & -2 & -2 & -2 & -2 \\ -2 & -2 & -2 & -2 & -2 \\ 0 & -2 & -2 & -2 & -2 \end{bmatrix}$$

Correlation between Satellite Handovers and Bandwidth Drops

- we have an algorithm to detect handovers
- we run in parallel 2 scripts
  - 1st: retrieves an obstruction map every second
  - 2nd: gathers Bandwidth data
- we visualize Bandwidth data and satellite handovers

Correlation between Satellite Handovers and Bandwidth Drops



Bibliography

[1] L. Izhikevich, M. Tran, K. Izhikevich, G. Akiwate, and Z. Durumeric.
Democratizing leo satellite network measurement, 2023.