Control-Plane-Triggered Traffic Performance Degradation Detection Final Report

Mingwei Zhang, Ph.D., BGPKIT

Background

Network Diagnostic Tool (NDT) is a network performance tool that measures the characteristics of a TCP connection under heavy load. The various RTT values and other performance metrics defined in NDT results (e.g. defined in ndt7 protocol) can reveal the network performance of an NDT client toward the defined NDT servers. By establishing normal performance profiles for clients (or client-server pairs), we can learn about the expected network performance and incidences of performance degradation.
Internet traffic performance degradation, traffic slowdown, or complete outages happen for different reasons. One particular group of traffic degradation that we are interested in exploring is the control-plane-triggered degradation. Knowing if and which control-plane behavior triggers a data-plane performance degradation allows network operators to better improve performance and potentially prevent future incidences.

Overview

During this fellowship, I worked on improving toolings for data analysis using M-Lab's NDT data and BGP data processing. Specifically, I have finished the following tasks:

- Familiarized me with M-Lab's BigQuery data storage and querying process
- Conducted multiple case studies on the basic NDT statistics.
- Developed time-series data machine learning process using Dart library to predict NDT test data volume and detect potential anomalies.
- Conducted longitude study on the overall measurement statistics of NDT5 and NDT7 data.
- Developed Python binding of BGPKIT BGP data processing tool for using it on Google Colab platform and NDT BigQuery data.
- Developed a prototype SDK to provide an abstraction over Google BigQuery's SQL interface and NDT data organization, allowing users to easily pull desired data without writing SQL statements.

In the rest of this report, I will describe the major pieces of my work and provide links to code, scripts, and additional documentation.

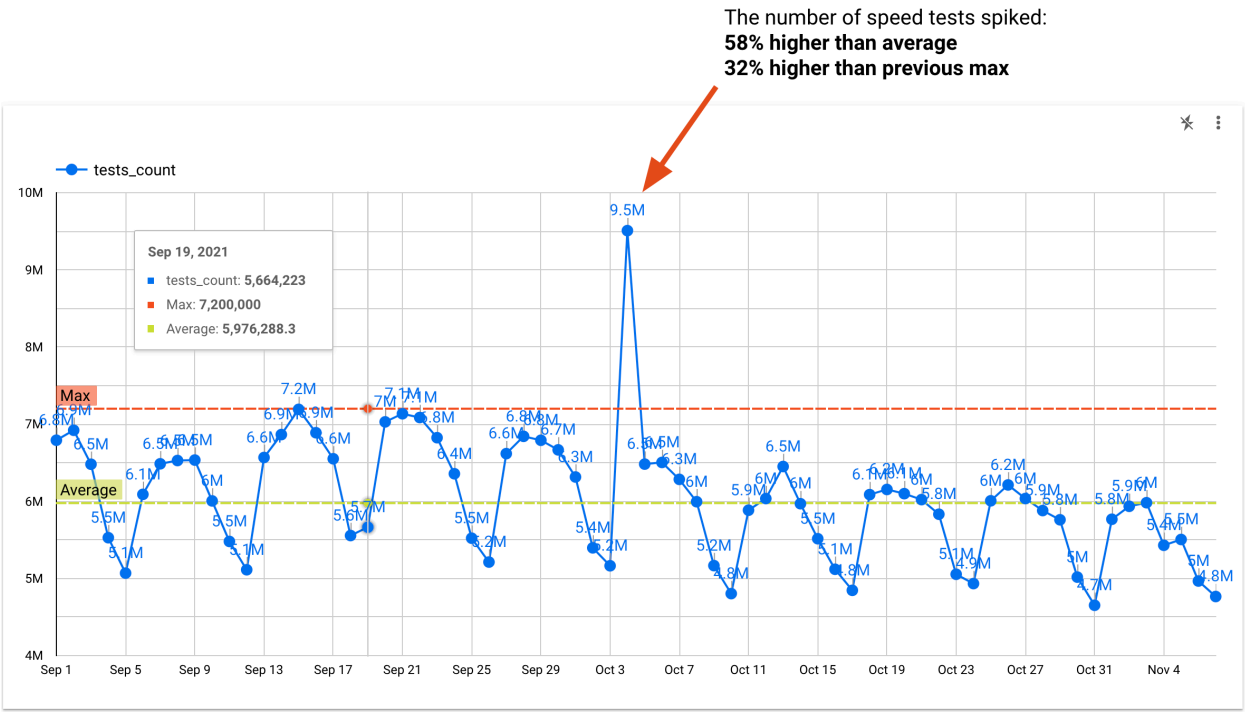
NDT Data Analysis and Discovery

From a very high level, the overall NDT data volume has been increasing stably for the past few years. After examining multiple relatively large-scale Internet routing events (reported by multiple media outlets), I found that most of the events do not impact the overall NDT data volume in a significant way. This can be attributed partly to the sparse nature of the data points and also the robustness/self-healing feature of the Internet routing. The only significant event that impacted the overall volume of the NDT data was the 2021 Facebook outage.

Facebook 2021 Outage Event Analysis

After collecting NDT tests daily counts metrics, I discovered that during the day of the large-scale Facebook outage event in 2021, there was an abnormally amount of speed test requests collected by the MLab platform. I also built a visualization showing the event, shown below.

https://datastudio.google.com/reporting/f7f6db6c-00f5-45f9-94f5-904eab94ffd2/page/4FJkC



The increase in NDT test volume has multiple indications.

1. There was a significant disruption of Facebook services globally instead of regional disruption.
2. There were also no significant Internet outages, as the test requests completed fine during the Facebook outage.

3. Facebook users associate the Facebook outage with Internet connection speed issues and thus reached out to NDT speed tests to confirm their suspicions.

Further analysis of the test results during the outage period revealed no significant changes in terms of network performance, which showed that the Facebook outage was localized to Facebook and its company's services only, not an Internet-level disruption.

Ukraine Network Stableness in Early 2022

Besides NDT data volume, I also examined the detailed measurement metrics over more recent routing events, such as Cogent disconnecting multiple Russian networks in IXPs. However, I could not detect significant changes in terms of MinRTT or throughput during the event time. This could mean that the network is routed around the disconnected points without causing a significant impact on the traffic speed.

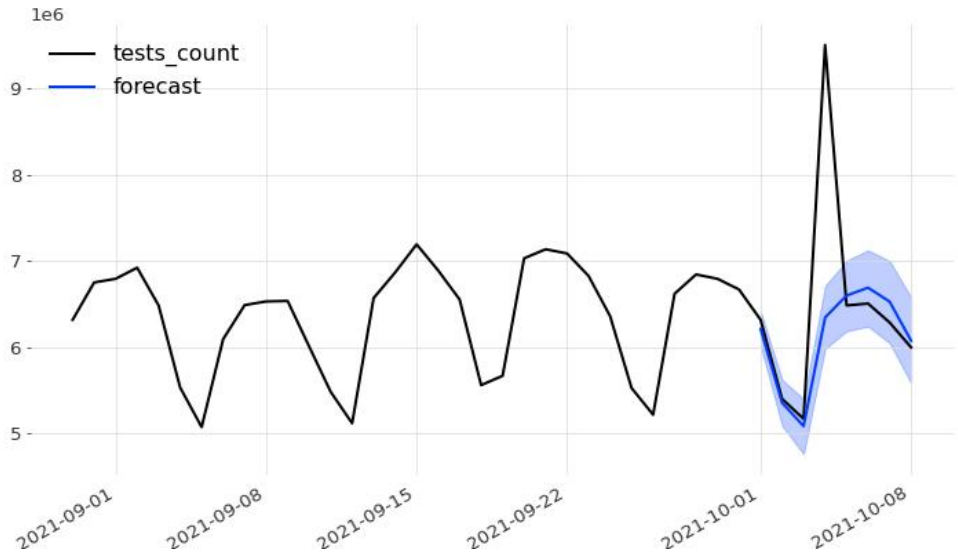
I have also made some more analysis on BGP data and found that although some routing changes caused periods of outages, the changes did not affect the NDT speed test as the tests were smart enough to find topologically nearby servers. For more details, please check out my post on a routing-trigger outage:

https://blog.bgpkit.com/2022-05-01-khersontelecom-connectivity-change/

Time-series machine learning

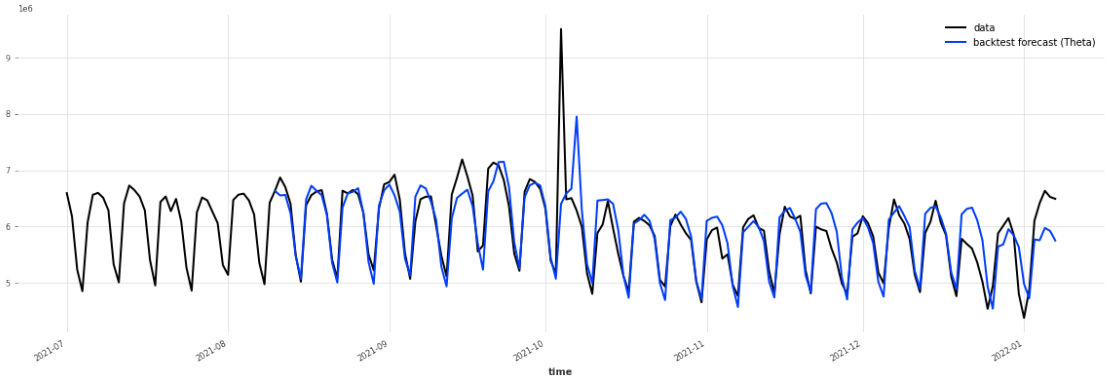
Since NDT data over time is inherently time-series data and shows day-night patterns, it is interesting to look further into if we can make predictions on NDT data volume and discover abnormal volumes automatically.

Using Darts time-series analysis library, I built a machine learning model applicable to daily NDT test counts time series data. The plot below shows a prediction (forecast) plot for the Facebook 2021 outage event period. The one month period before the event is used for training,

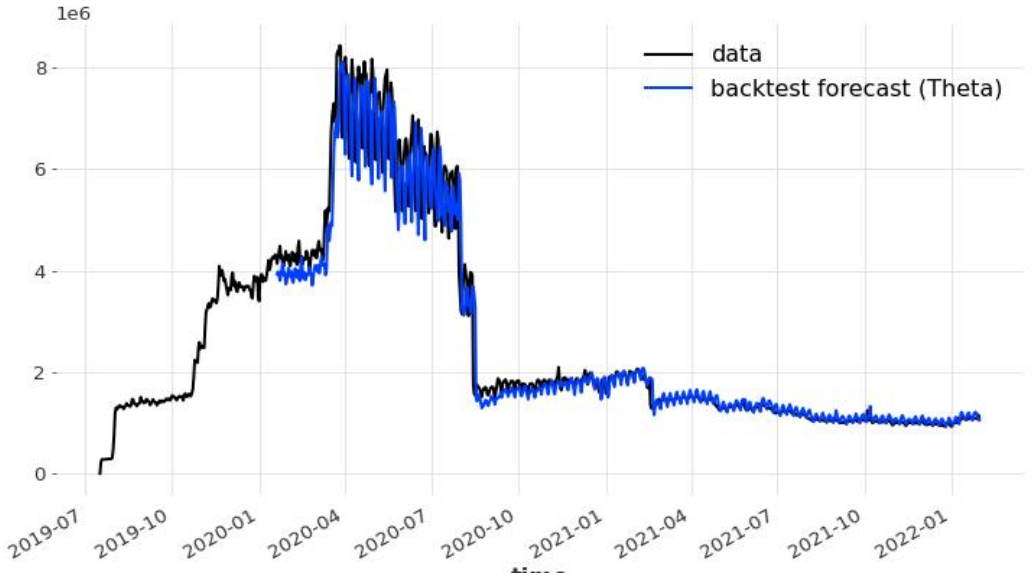


and I used Dart to make predictions on the week of the event. We can clearly see that the event day's data fell outside the prediction range significantly.

I also built a moving-window version of the prediction pipeline where the program uses the previous month's data for training and make a one-day prediction every day continuously. Applying this model for a longer period and we have the following result graph. We can see that the Facebook event again stroke as an outlier, which also disrupted the following day's prediction.



This model can be utilized for a much longer period as well. The following graph shows a continuous prediction graph on top of the real data for the past three years.



NDTGuide Python Package

NDTGuide is a Python package that aims to provide easy-to-use, quick access to MeasurementLab's (MLab) NDT measurement data on Google BigQuery. NDTGuide provides an abstract layer around the Google BigQuery interface and MLab's data schema. At its core, it provides a growing number of functions that translate user intentions into BigQuery SQL statements.

This library is intended to work only on the Google Colab platform. The library is hosted under MIT license at https://github.com/bgpkit/ndtguide. The package is also available on PyPI at https://pypi.org/project/ndtguide/.

Usage

The full example code of this usage guide is available here.

Install the ndtguide Package on Google Colab

Execute the following code block at the very beginning of your Google Colab script to install and import the package.

```
!pip install ndtguide==0.1.1
from ndtguide import NDTGuide
```

Login with Google Account (Required Step)

Since the MLab's data access is tied to individual Google accounts, it is required to first login to Google first when first running the script. NDTGuide provided a wrapper .login() function that interactively prompt user to login.

```
guide = NDTGuide()
guide.login()
```

Gather Daily NDT Stats

One of the most used queries is to pull out daily aggregated statistics of the measurement data to gather some overview of certain clients/servers/regions. `NDTGuide` provides the `.sql_daily_aggregate(...)` function to generate sql statements for this task.

It accepts the following required parameters:

- `table_name`: currently supports `ndt7` or `ndt5`

- `date_start` and `date_end`: start and end date, in the format of `YYYY-mm-dd`
- `aggr_func`: aggregation function, currently supports `avg`, `max`, `min`

Additional filters for NDT clients and servers include:

- `client_asn` and `server_asn`: autonomous number of the client/server
- `client_cidr` and `server_cidr`: IP block (CIDR) of the client/server
- `client_country` and `server_country`: two-letter country code of the client/server

The query should return results of the following data:

- mean throughput
- minimum RTT
- packet loss rate

The following example queries the average measurements from `ndt7` table between `2022-02-01` to `2022-02-10` for clients located in Ukraine.

```
sql = guide.sql_daily_aggregate("ndt7", "2022-02-01",
"2022-02-10", "avg", client_country="ua")
guide.exec_sql(sql)
```

	avg_throughput	avg_rtt	avg_lossrate	date
0	49.453097	52.033121	0.042338	2022-02-01
1	50.410214	53.451678	0.051775	2022-02-02
2	52.047963	52.372124	0.043772	2022-02-03
3	54.767157	55.180171	0.041808	2022-02-04
4	47.897135	58.383362	0.029880	2022-02-05
5	75.044735	59.411379	0.041825	2022-02-06
6	91.473595	49.754529	0.048684	2022-02-07
7	58.433473	48.761221	0.041670	2022-02-08
8	105.985296	56.598006	0.034676	2022-02-09
9	64.017782	57.256041	0.045156	2022-02-10

Gather Clients and Servers

NDTGuide provides function to generate queries look for

- clients that users servers in certain network
- servers that the clients in certain network uses

These functions allow users to quickly locate relevant clients/servers for any interested networks.

For example, the following query gathers all the NDT servers any clients from AS3216 used during a one week period:

```
sql = guide.sql_get_servers("ndt7", "2022-01-01",
"2022-01-07", "3216")
print(sql)
df = guide.exec_sql(sql)
df
```

```
     SELECT distinct server.Site, server.Machine, server.Network.ASNumber, server.Network.ASName, server.Network.CIDR,
server.Geo.CountryCode, server.Geo.City
     FROM `measurement-lab.ndt.ndt7`
     WHERE date>='2022-01-01' and date<='2022-01-07'  and client.Network.ASNumber=3216
```

	Site	Machine	ASNumber	ASName	CIDR	CountryCode	City
0	beg01	mlab1	13004	Serbian Open Exchange DOO	188.120.127.0/26	RS	Belgrade
1	arn03	mlab1	3356	Level 3 Parent, LLC	213.242.86.64/26	SE	Stockholm
2	beg01	mlab2	13004	Serbian Open Exchange DOO	188.120.127.0/26	RS	Belgrade
3	arn04	mlab2	1299	Telia Company AB	62.115.225.128/26	SE	Stockholm
4	arn05	mlab3	3257	GTT Communications Inc.	77.67.119.64/26	SE	Stockholm
5	arn03	mlab2	3356	Level 3 Parent, LLC	213.242.86.64/26	SE	Stockholm
6	beg01	mlab3	13004	Serbian Open Exchange DOO	188.120.127.0/26	RS	Belgrade
7	arn02	mlab3	1273	Vodafone Group PLC	195.89.146.192/26	SE	Stockholm
8	hnd02	mlab1	2518	BIGLOBE Inc.	210.151.179.128/26	JP	Tokyo
9	arn06	mlab3	6453	TATA COMMUNICATIONS (AMERICA) INC	193.142.125.64/26	SE	Stockholm
10	arn04	mlab1	1299	Telia Company AB	62.115.225.128/26	SE	Stockholm
11	arn06	mlab2	6453	TATA COMMUNICATIONS (AMERICA) INC	193.142.125.64/26	SE	Stockholm
12	arn04	mlab3	1299	Telia Company AB	62.115.225.128/26	SE	Stockholm
13	hnd04	mlab1	5580	GTT Netherlands B.V.	64.235.255.128/26	JP	Tokyo
14	arn05	mlab1	3257	GTT Communications Inc.	77.67.119.64/26	SE	Stockholm
15	arn02	mlab1	1273	Vodafone Group PLC	195.89.146.192/26	SE	Stockholm
16	arn02	mlab2	1273	Vodafone Group PLC	195.89.146.192/26	SE	Stockholm
17	hnd03	mlab1	2516	KDDI Corporation	111.109.1.64/26	JP	Tokyo
18	hnd04	mlab3	5580	GTT Netherlands B.V.	64.235.255.128/26	JP	Tokyo
19	arn05	mlab2	3257	GTT Communications Inc.	77.67.119.64/26	SE	Stockholm
20	hnd03	mlab3	2516	KDDI Corporation	111.109.1.64/26	JP	Tokyo
21	arn06	mlab1	6453	TATA COMMUNICATIONS (AMERICA) INC	193.142.125.64/26	SE	Stockholm
22	arn03	mlab3	3356	Level 3 Parent, LLC	213.242.86.64/26	SE	Stockholm

Customizable Queries

NDTGuide provides a .get_schema() function to provide a selected useful schema to help with manually constructing BigQuery queries.

```
guide.get_schema()
{'a': {'CongestionControl': 'string',
  'LossRate': 'float',
  'MeanThroughputMbps': 'float',
  'MinRTT': 'float',
  'TestTime': 'TimeStamp',
  'UUID': 'string'},
 'client': {'Geo': {'City': 'string',
   'ContinentCode': 'string',
   'CountryCode': 'string',
   'CountryName': 'string'},
  'Network': {'ASName': 'string', 'ASNumber': 'integer',
'CIDR': 'string'}},
 'date': 'date',
```

```
 'id': 'string',
 'server': {'Geo': {'City': 'string',
   'ContinentCode': 'string',
   'CountryCode': 'string',
   'CountryName': 'string'},
  'Machine': 'string',
  'Network': {'ASName': 'string', 'ASNumber': 'integer',
'CIDR': 'string'},
  'Site': 'string'}}
```

The customized queries can be passed into the same `.exec_sql(sql)` function, similar to other built-in functions.

PyBGPKIT Python Package

Apart from data analysis and tooling for the NDT data, in this project, I also worked extensively on improving the control-plane data analysis by porting the Rust-based BGPKIT data processing tool to Python. The main benefit of the port is that the software is now usable together with all the analytical tools on Google Colab or any Python scripts/notebooks.

The source code is fully open-source under the MIT license and is available at https://github.com/bgpkit/pybgpkit.

Example usages

The following code snippet parses a remote MRT file and filter out all messages but the ones from two specified peers.

```
import bgpkit
parser = bgpkit.Parser(url="https://spaces.bgpkit.org/parser/update-example",
                       filters={"peer_ips": "185.1.8.65,2001:7f8:73:0:3:fa4:0:1"})
elems = parser.parse_all()
assert len(elems) == 4227
```

The following code snippet searches and fetches the meta information of MRT files on 2021-10-20.

```
import bgpkit
broker = bgpkit.Broker()
items = broker.query(ts_start="1634693400", ts_end="2021-10-20T01:30:00")
for item in items:
    print(item)
print(len(items))
assert len(items) == 58
```

The following example shows a query that asks for all the validated ROA payload for RIPE NCC on the date of 2018-01-01.

```
import bgpkit
roas = bgpkit.Roas()
data = roas.query(debug=True, asn=3333, date="2018-01-01")
for entry in data:
```

```
    print(entry)
assert len(data) == 10
```

Usages on Google Colab

Here are a few event analysis conducted using PyBGPKIT directly on Google Colab.

1. Simple usage examples: https://colab.research.google.com/drive/1AuNnzT43LYAZNnp1muhJTy0rvv3qbCX6?usp=sharing
2. Japan earthquake triggering routing changes: https://colab.research.google.com/drive/1TeC6OUIkB8YQZ7qW08acQADJWmaT87Dk?usp=sharing
3. Twitter prefix hijacked by Rusian Network RTCOMM: https://colab.research.google.com/drive/1HPBQXkN2IvjYmPztCQv8dsrUYyoSXqpj?usp=sharing
4. Cryptocurrency exchange point hijacked by attackers: https://colab.research.google.com/drive/1juuTgMGbGG7MGKvZJys44bzZAsTlnQjZ?usp=sharing