New methods of anomaly detection in PerfSONAR data

James Zhang, Ilija Vukotic

September 13, 2017

Overview

- Goals and Challenges
- Performance measurements from perfSONAR
- Factors influencing throughput
- Related work
- Collecting the data
- Anomaly detection
  - Simulated data
  - Actual data

Goals

Use machine learning algorithms to analyze network performance data collected by perfSONAR to identify significant anomalies in the network performance.

Challenge

Due to the network mesh size, it is very difficult and time consuming for someone to comb through this data looking for issues, sort them according to their seriousness, pinpoint causes, assign responsibility to fix.

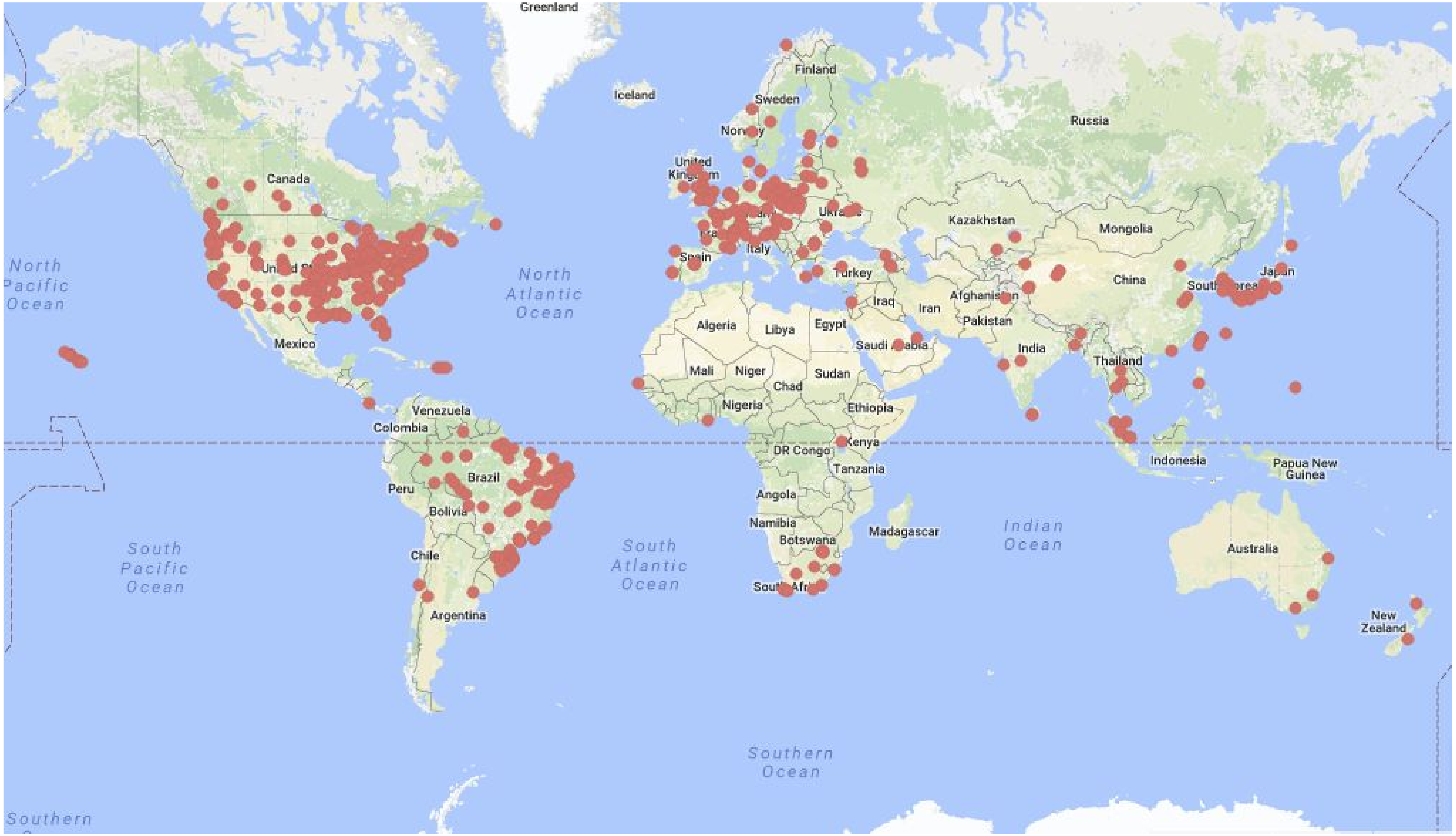
Performance measurement with perfSONAR

perfSONAR is a network measurement toolkit that monitors and stores network performance data between pairs of endpoints ("**links**")

We are interested in 4 of the Network performance measurements:

1) **packet loss rate** - percentage of lost packets over the total transferred packets in 1 minute intervals (actually measured at 10Hz)
2) **one way delay** - measures delay (in ms) separately for each direction of a path
3) **Throughput** - measures the amount of data that can be transferred over a period of time (25 seconds)
4) **Traceroute** - finds the path between the source and destination and the transition time

While PerfSONAR is installed on thousands of servers, we are specially interested in ones that are part of the **WLCG and OSG meshes**.


Greenland
Iceland
Finland
Sweden
Norway
Russia
United Kingdom
Canada
Kazakhstan
Mongolia
Ukraine
North Pacific Ocean
United States
North Atlantic Ocean
Spain
Italy
Turkey
China
Japan
South Korea
Afghanistan
Iraq
Iran
Pakistan
Algeria
Libya
Egypt
Saudi Arabia
India
Mexico
Mali
Niger
Sudan
Chad
Thailand
Nigeria
Ethiopia
Venezuela
Colombia
Kenya
DR Congo
Indonesia
Papua New Guinea
Tanzania
Peru
Brazil
Angola
Bolivia
Namibia
Madagascar
Indian Ocean
Botswana
South Pacific Ocean
Chile
South Atlantic Ocean
Australia
South Africa
New Zealand
Argentina
Southern Ocean
Southern Ocean

Factors influencing throughput

Most frequent reasons for decreased network performance are:

- Full connectivity disruption - all packets are lost.
- A device on the path is close to saturation - increase in one way delay as packets spend more time in device’s buffers.
- A device on the path is saturated - packets are lost as device’s buffers overflow, delays increase.
- Routing changes - large variance in one way delays, asymmetrical paths, more time to reorder packets.
- Dirty fiber, other problems with optics - increase in packet loss rate

Why is packet loss so important?

Packet loss very strongly limits maximal possible bandwidth for high latency links.

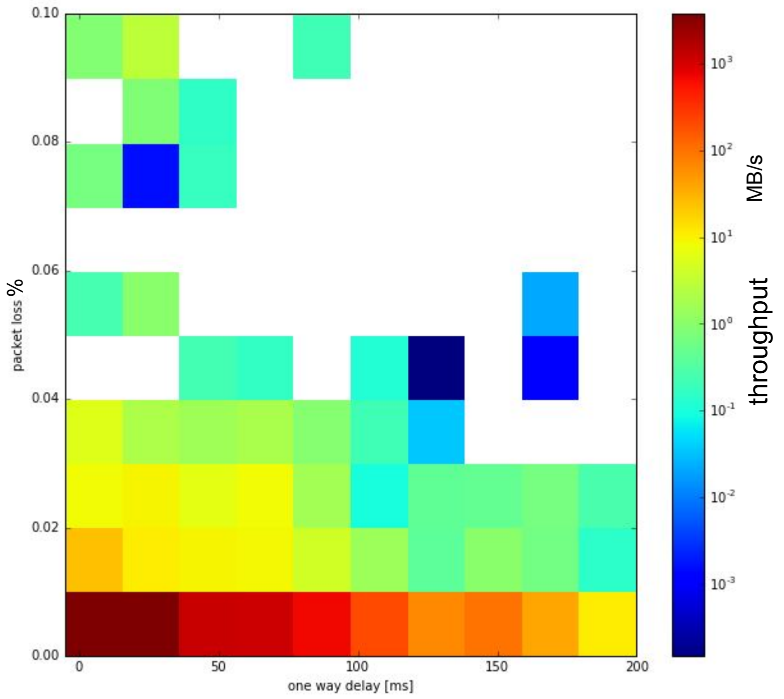
From Mathis et al. Rate < (MSS/RTT)*(1 / sqrt(PL))

MSS - maximum segment size

RTT - round trip time

PL - packet loss

It gives no predictive value for packet loss lower than our sensitivity.



More details: https://en.wikipedia.org/wiki/Bandwidth-delay_product
http://fasterdata.es.net/ and presentations by Jason Zurawski

Previous attempts

First idea was to use Support Vector Machine (SVM) as one class classifier. Suffers from the course of dimensionality as we usually have 60+ features to consider per site.

Next we tried BASIC - Zhou Fan implementation of empirical bayesian analysis of simultaneous changepoints in multiple time series (https://arxiv.org/abs/1508.01280, github). While giving good results it is not fast enough for our data scale.

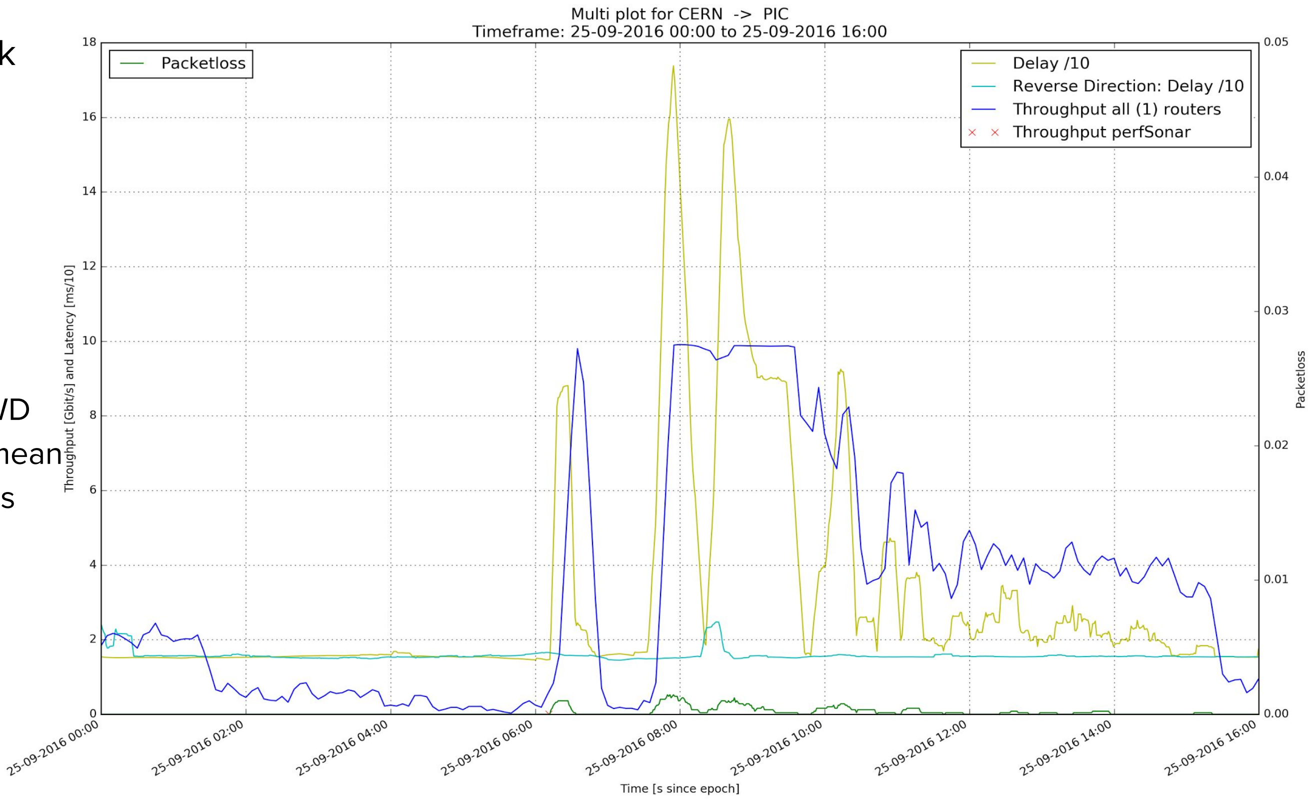
Related work - continued

Hendrik Borras and Marian Babik tried to detect high utilization periods on LHCONE links.

Both delay and packet loss very sensitive to high utilization.

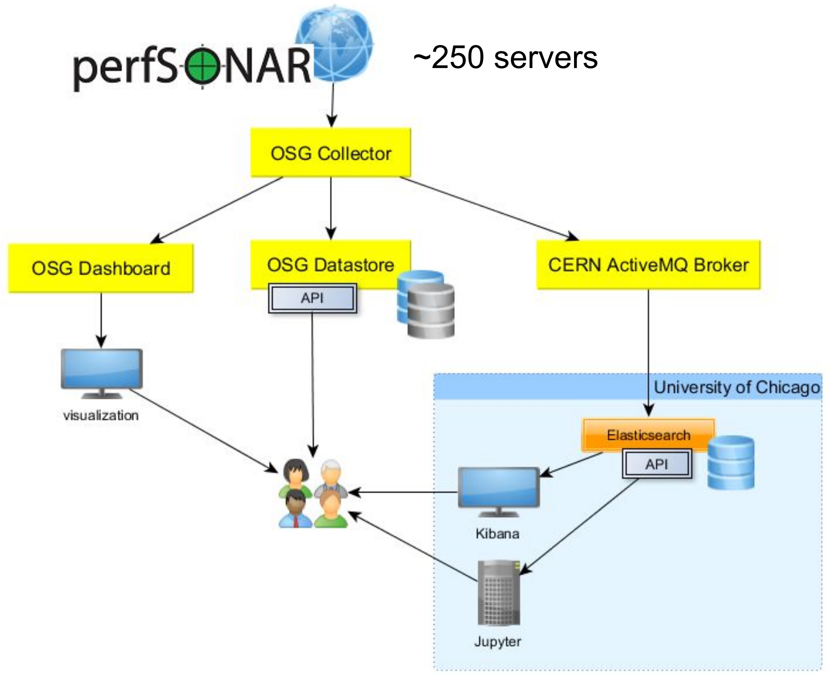
Recurrent neural network:

- Inputs - 15 previous PL and OWD measurements (15 min span), mean and variance values of previous one week.
- 3 layers, one output, rectified linear unit activation
- Trained to predict utilization between 70 and 100%



Collecting the data

- PerfSONAR collects network performance data and sends it to a messaging service (ActiveMQ at CERN)
- Python collector subscribes to AMQ, enrichies the data, sends it to Elasticsearch server at UC
- Data is indexed in Elasticsearch
- Data analysis on a co-located Jupyter cluster querying ES.



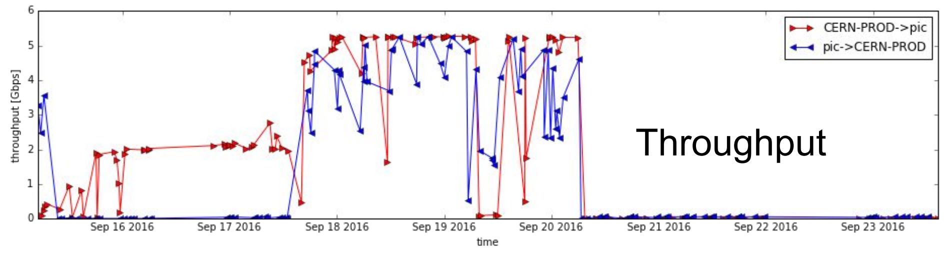
Collecting the data

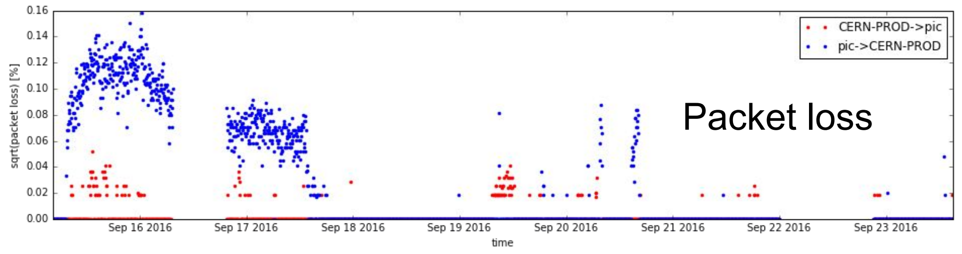
We collect ~8.3GB of data in ~16M documents per day.

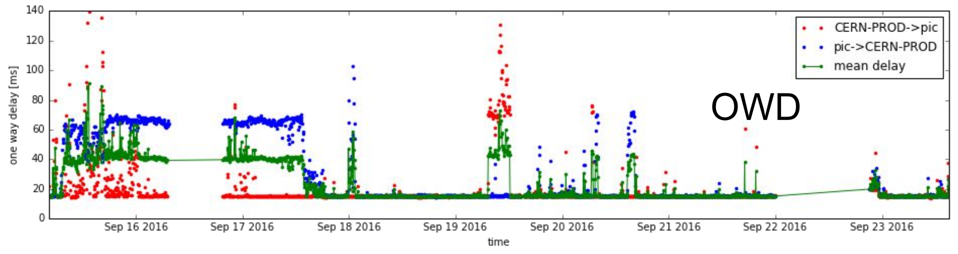
PerfSONAR data rates dashboard

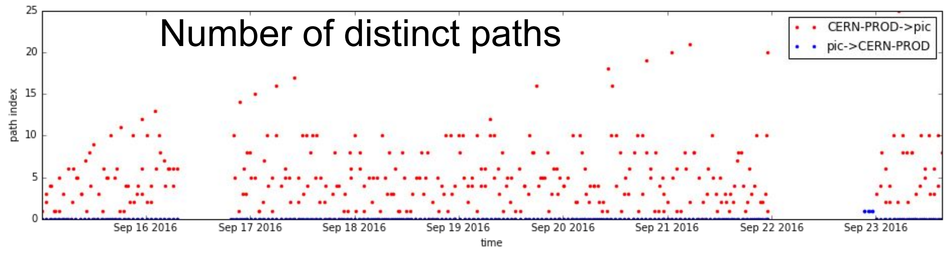
MWT2 inter-site mesh dashboard

Measurement type	Sources		Destinations		Links
	sites	instances	sites	instances	
Packet loss rate	73	196	79	205	9041
One way delay	73	196	78	205	8971
Throughput	75	236	74	236	8933
Traceroute	83	233	93	361	7456









General considerations

An ideal anomaly method would:

- Naturally combine disparate data features and even data sources
  - All the links to/from site
  - Packet loss, OWDs, paths, throughputs, FTS measurements, ...
- Give information on what features (combinations of features) are causing the anomaly
  - We want to know who to alert
- Have tunable sensitivity
  - We are not interested in short duration (order of hour) flukes as no action can be taken at time scales shorter than that.
- Be sufficiently performant at our scale

Following promising results on a similar problem we decided to implement two new methods:

- Boosted decision trees
- Simple feed forward neural network

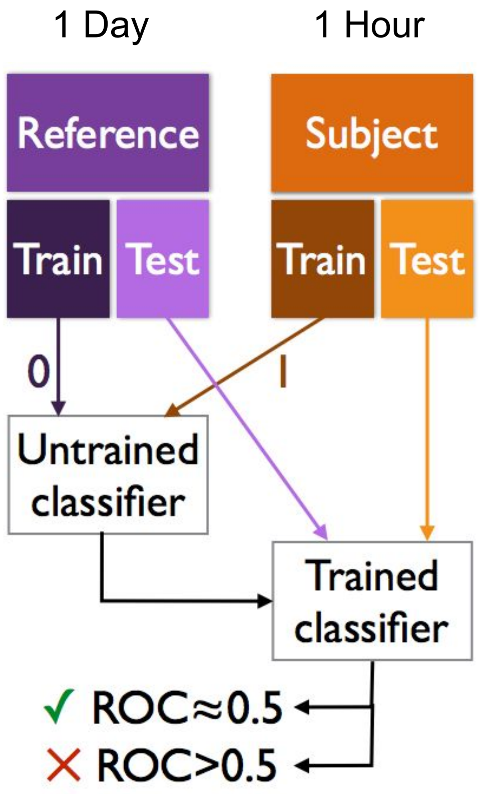
Approach for Detecting Anomalies

Objective is to train a model to be able to distinguish the data from two periods of time. If it is unable to distinguish the two periods of time, then there was no significant change between the two. If it is able to distinguish the two periods of time, then something happened to cause a change.

Reference data is the data from one day (1st period of time). Subject data is the data from the hour following that day (2nd period of time).
e.g. Reference data is flagged with 0 and subject data is flagged with 1.

70% of the data from both the reference and subject periods are combined and used to train a BDT to separate flags. Training effectiveness was then tested on the remaining 30% of the data.

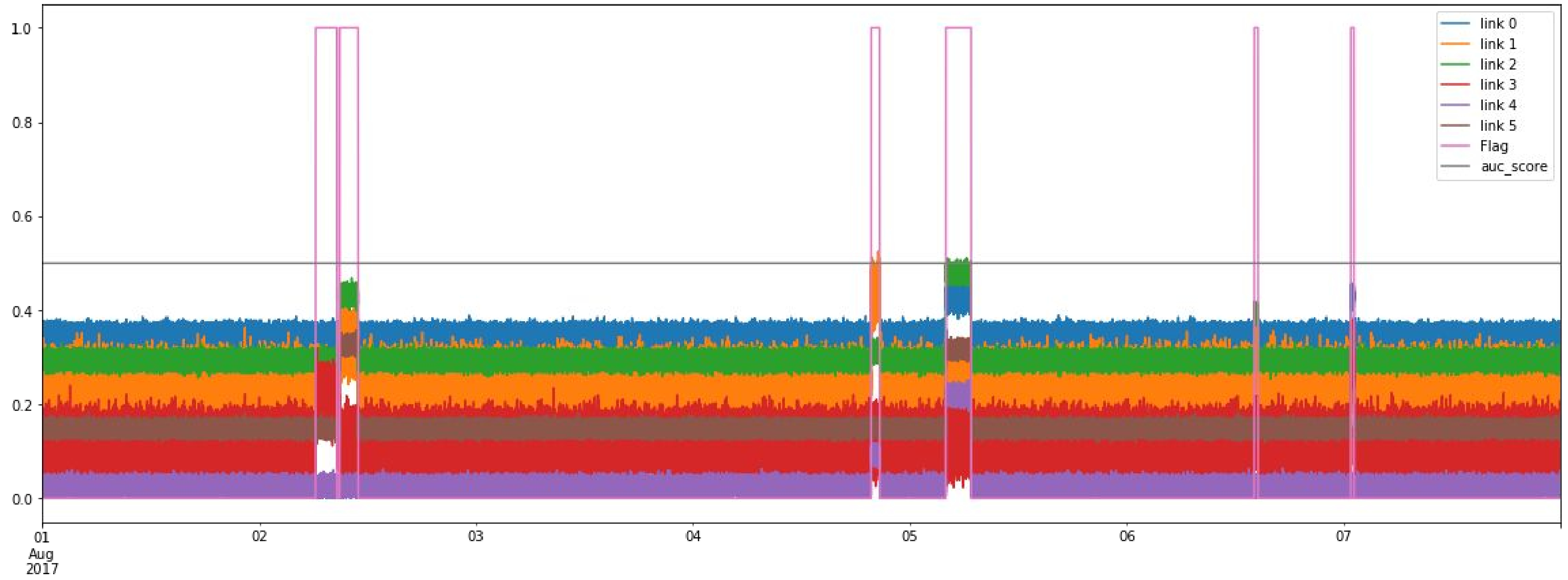
An AUC score is determined, and if the score is above the threshold of 0.55, then the result is determined to be an anomaly



Simulated Data

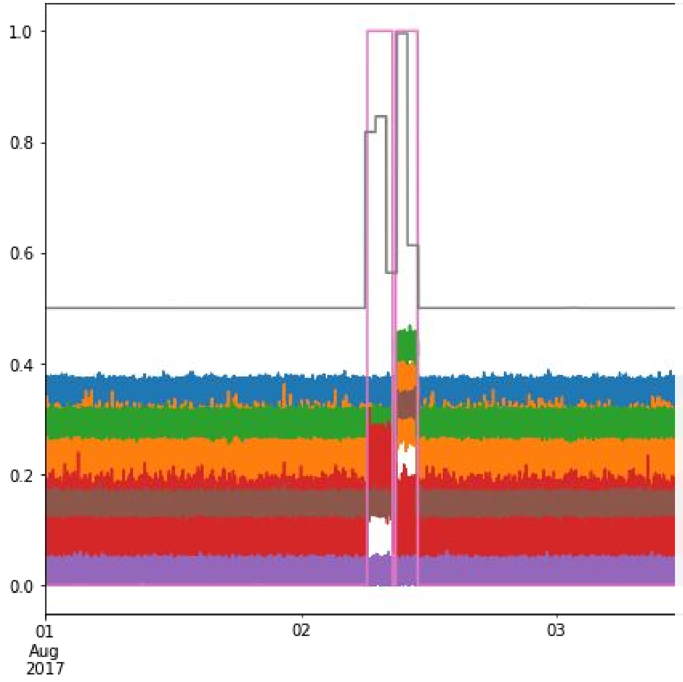
Simulation of 6 timeseries with anomalies.

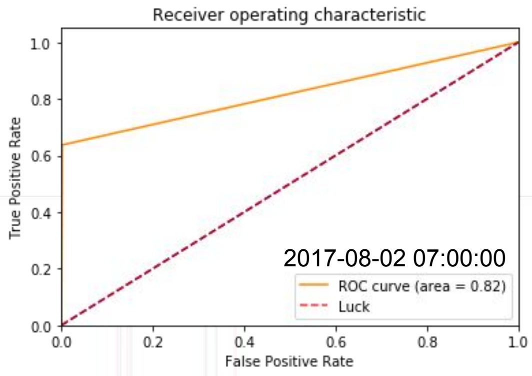
```
2017-08-02 06:13:02  2017-08-02 08:36:03  affected: [3]
2017-08-02 08:52:35  2017-08-02 10:57:00 affected: [2, 5, 1]
2017-08-04 19:42:17  2017-08-04 20:39:28 affected: [0, 4, 2, 1]
2017-08-05 03:59:13  2017-08-05 06:47:32 affected: [2, 5, 4, 0]
2017-08-06 14:06:20  2017-08-06 14:32:08 affected: [1, 2, 3]
2017-08-07 00:48:02  2017-08-07 01:09:34 affected: [0, 5, 2, 1, 3]
```



Anomaly threshold

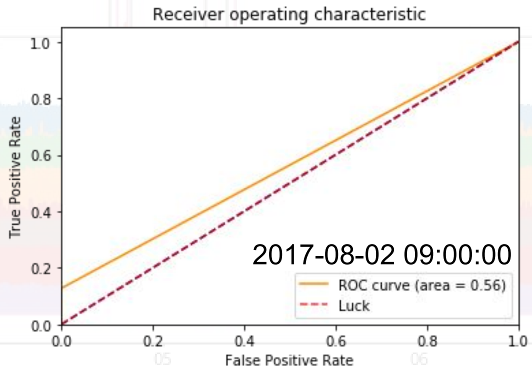
AUC scores : [Code]()





Feature importance:

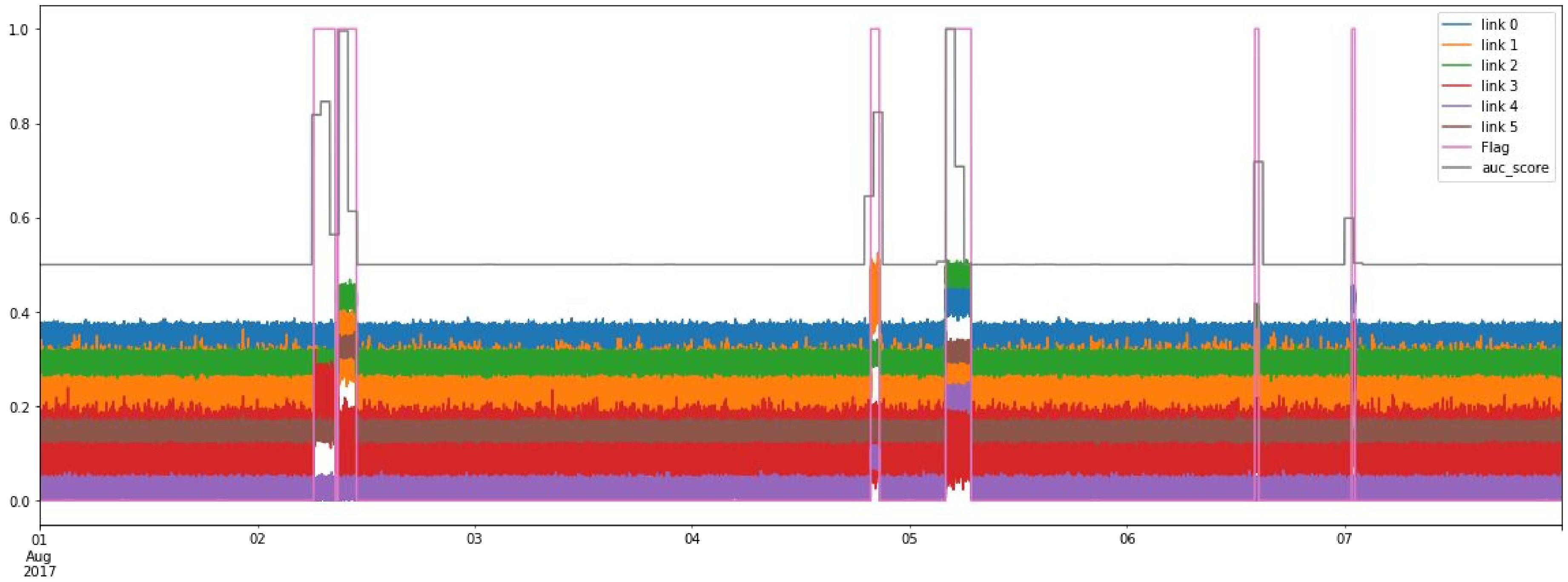
Link 3	0.36
Link 1	0.24
Link 0	0.12
Link 2	0.06
Link 4	0.06
Link 5	0.16



Feature importance:

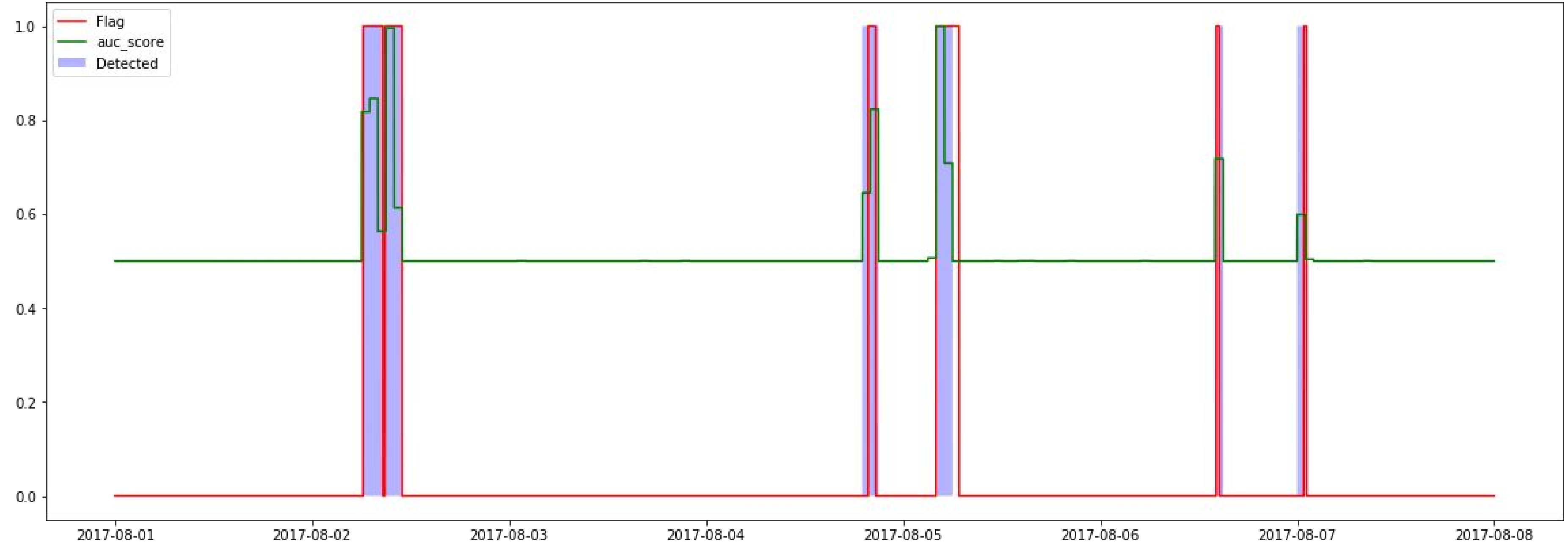
Link 3	0.34
Link 1	0.16
Link 2	0.20
Link 0	0.14
Link 4	0.08
Link 5	0.08

Adaboost (50 stumps) 24h ref, 1h subject



Results (Anomaly detection on Simulated Data)

On this data cut on AUC > 0.55 gives good results.

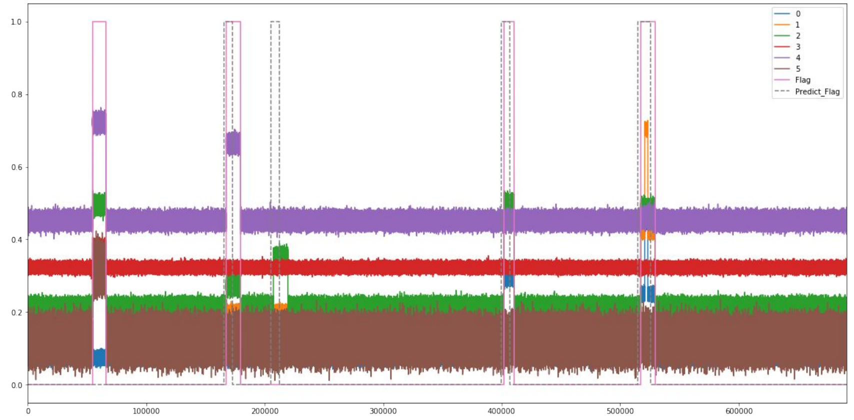


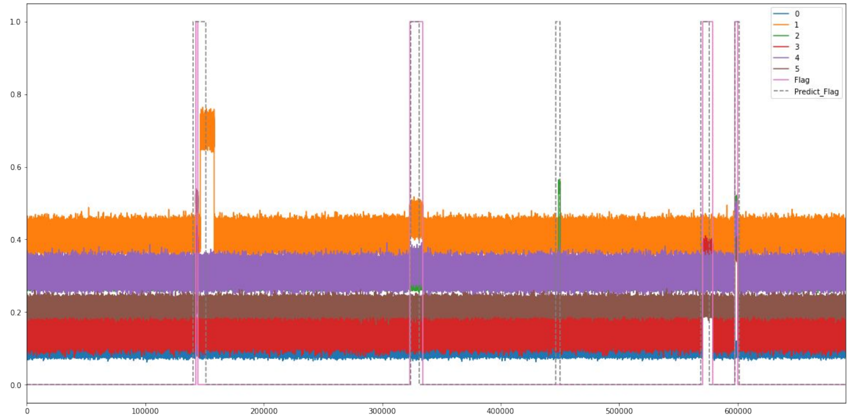
Using Decision Stumps vs Decision Trees (depth: 6)

Stumps Run Time: 674.511009 seconds ≈ 11.24 minutes

Tree Run Time: 2458.262419 seconds ≈ 40.97 minutes

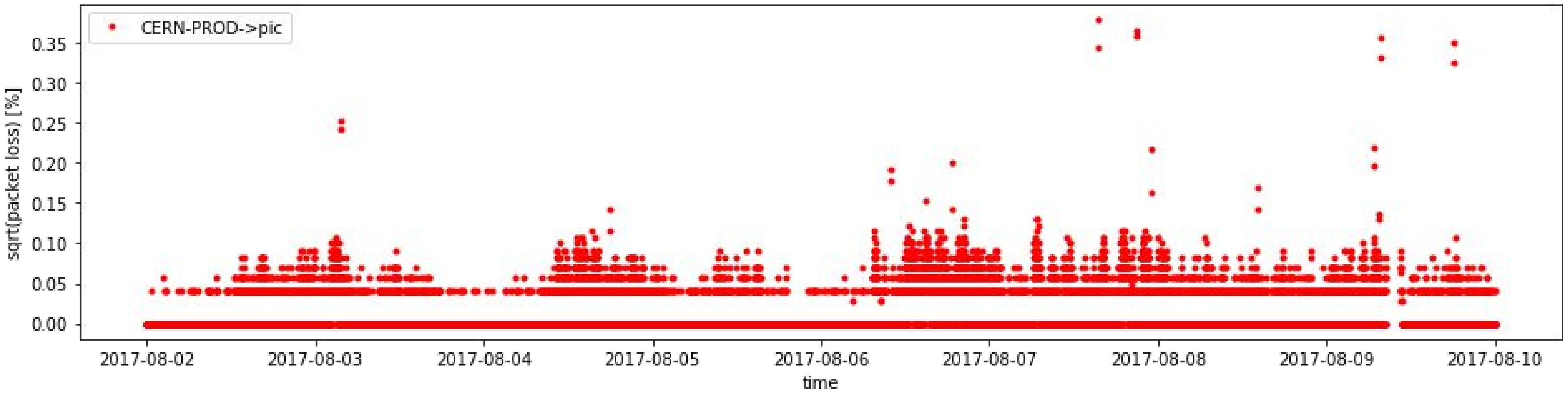
Results virtually the same: Stumps (left) vs. Trees (right)



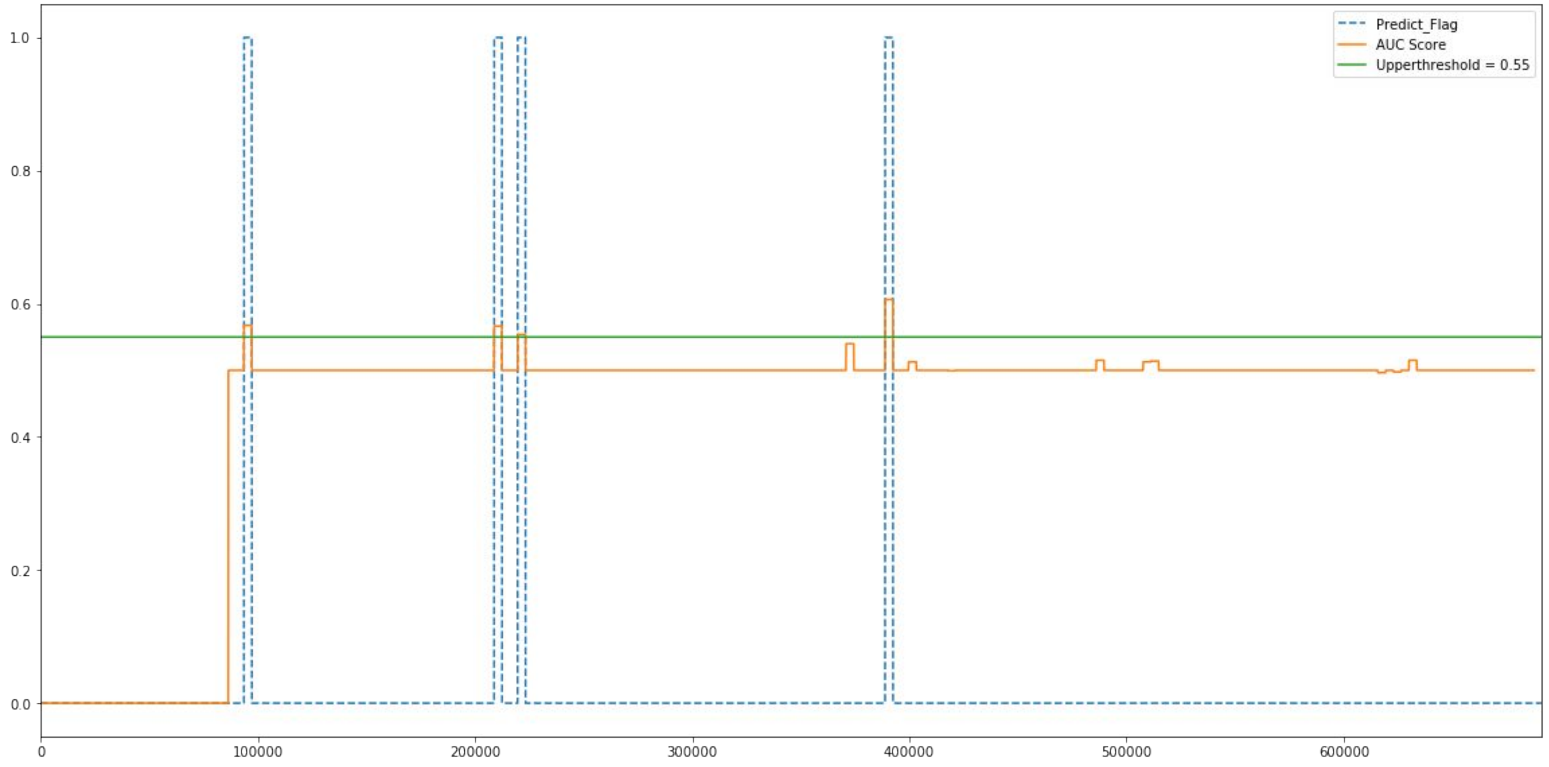


Real Data

Packet loss rate gathered between 8/02/2017 - 08/10/2017: data collected every minute: link

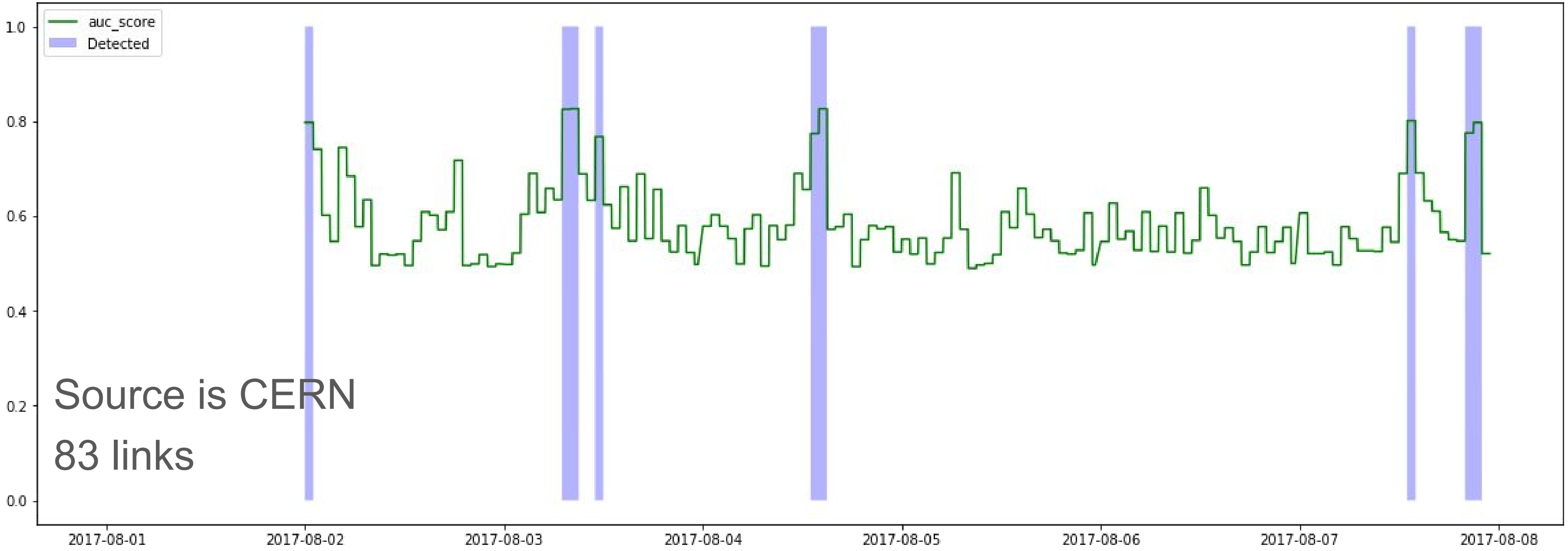


Actual Data Anomaly detection - single link



All links from one site

There are always some anomalies.
Task should be redefined from “anomaly detection” to “most significant anomaly detection”.



Source is CERN
83 links

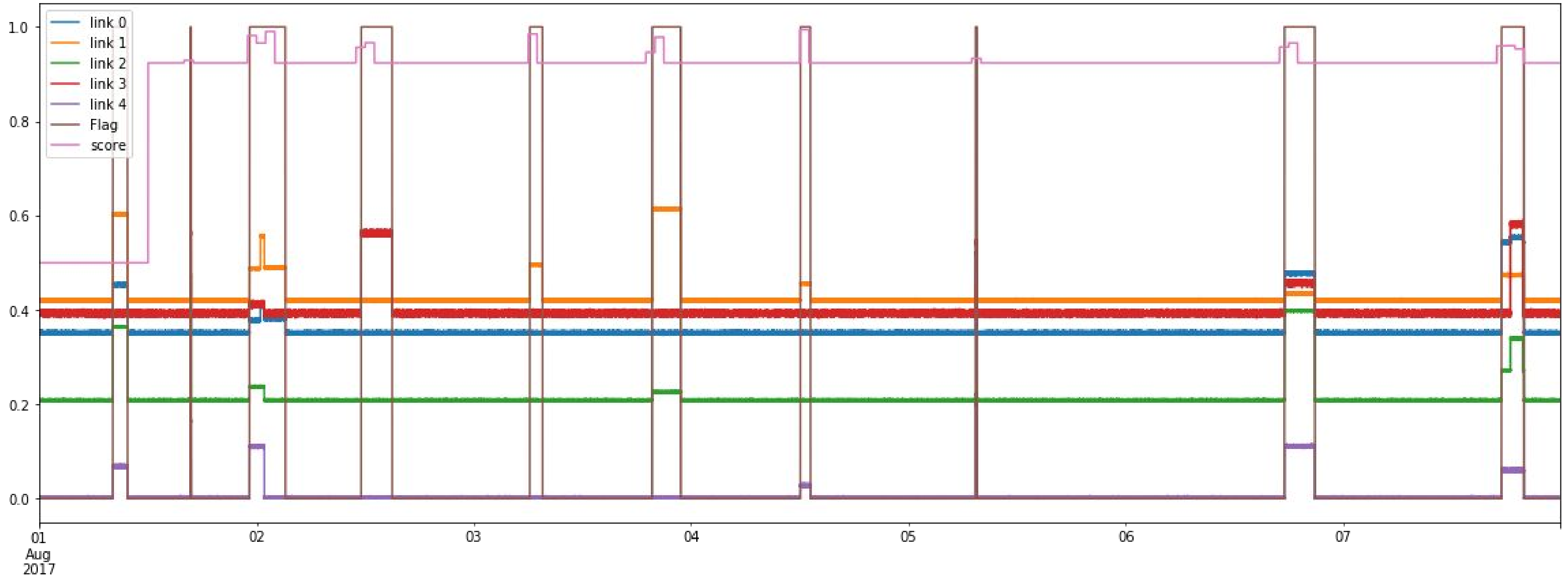
NN approach

Try using simple feed forward NN in the same way as BDT.

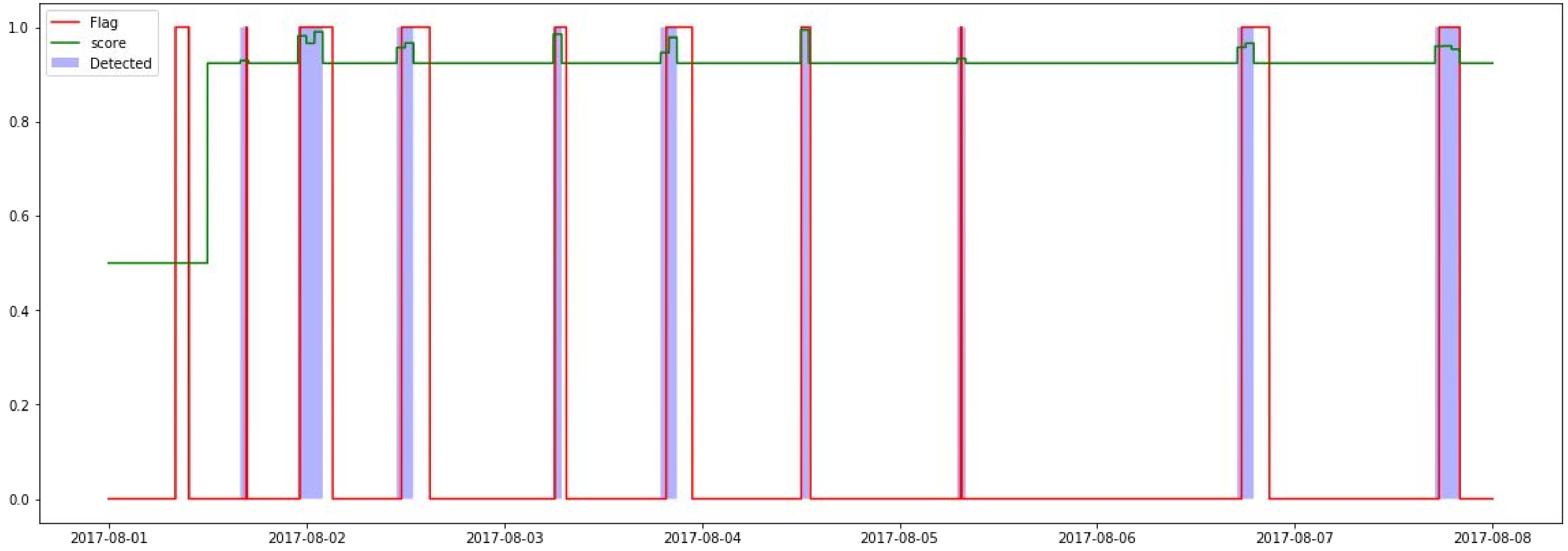
2 hidden layers with 2 * n_series relu neurons, 1 sigmoid output, no dropout (for now).

100 epochs/128 batch size on 1 Tesla K20 takes ~20 seconds per test period.

NN on simulated data



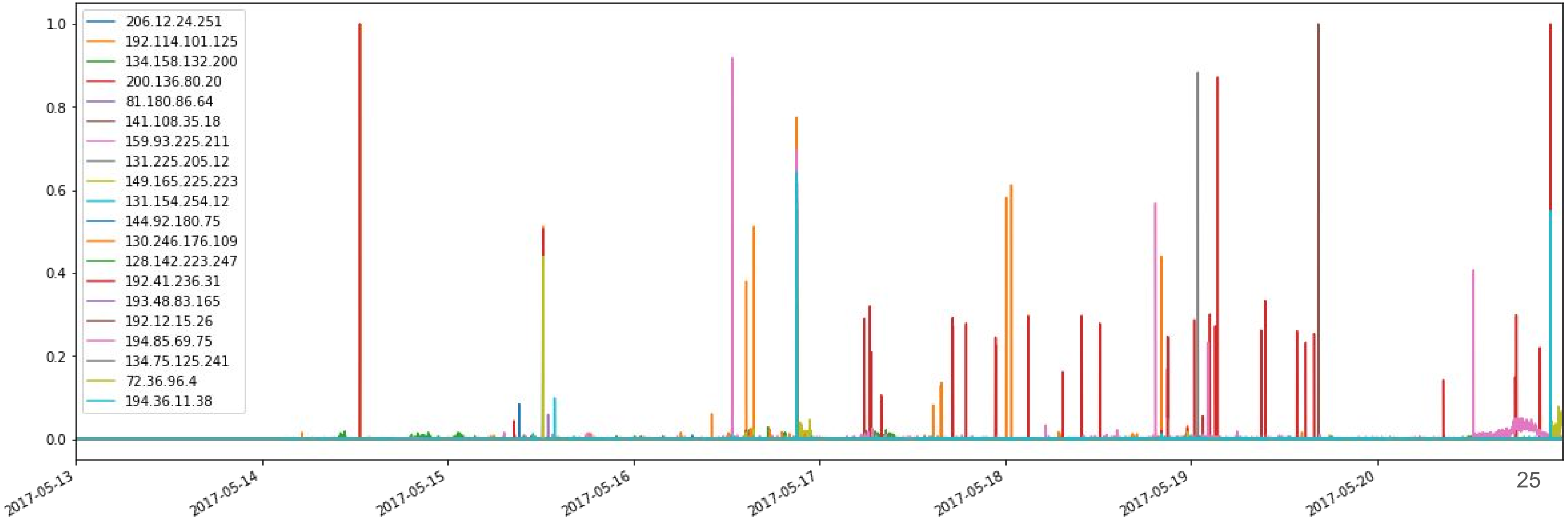
NN on simulated data



NN on actual data

20 links where PIC is source.

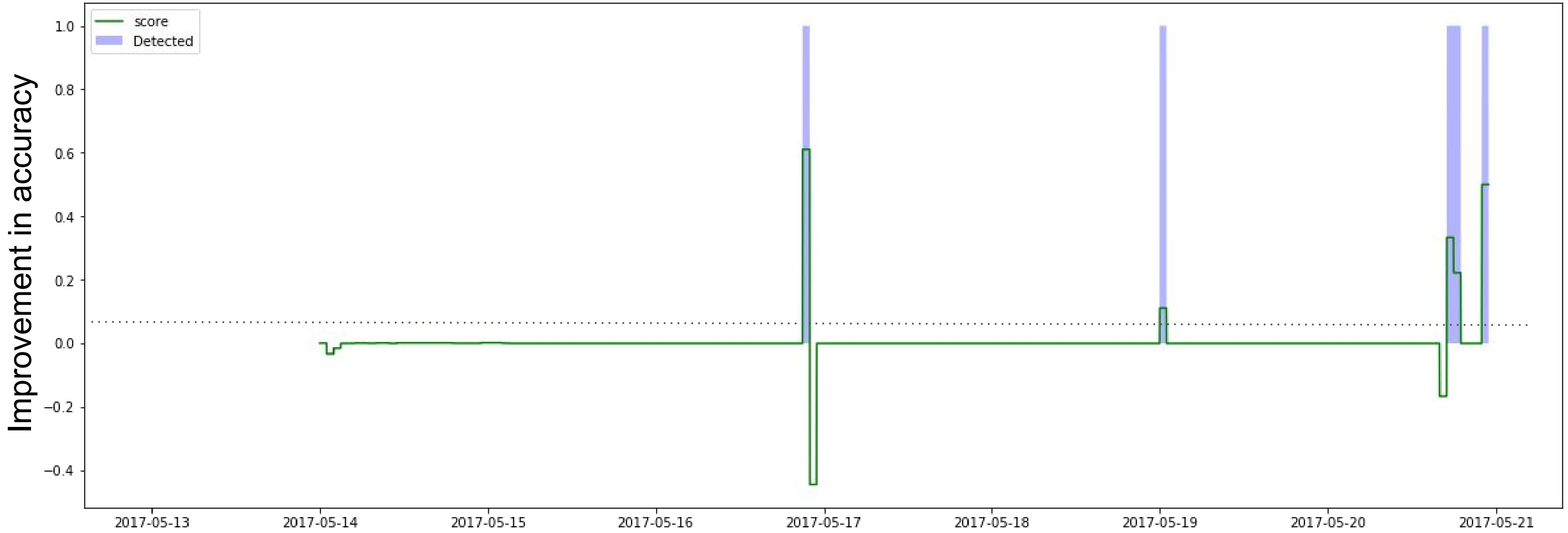
24h reference, 1h subject periods.



NN on actual data

Chance gives 96% accuracy.

5% improvement in accuracy (96.2%) considered anomaly.



Next Steps

Create a test alert service for several sites.

By hand annotate anomalies of 10 links in one 8 day period. Rather difficult thing to do even on so limited set.

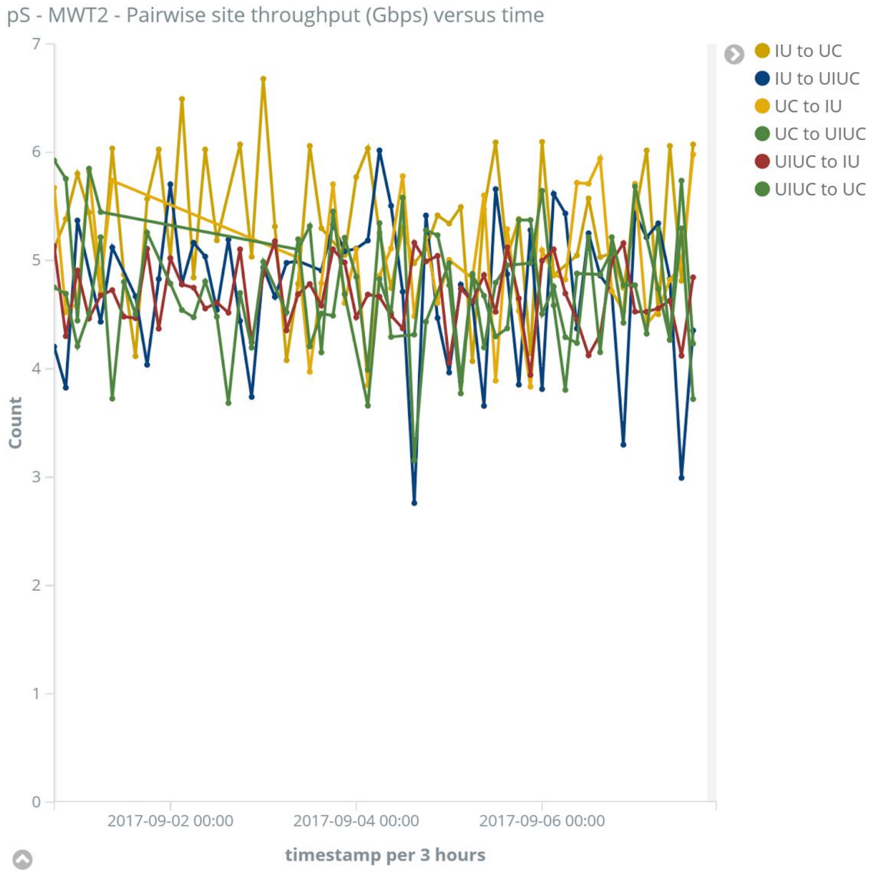
Retest on all links from one site also in 8 day period. Tune sensitivity on 6 months period.

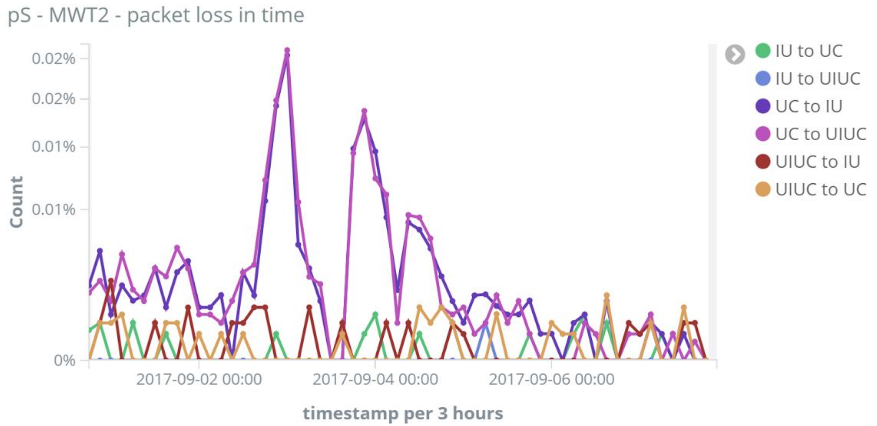
Add delay info. Retest.

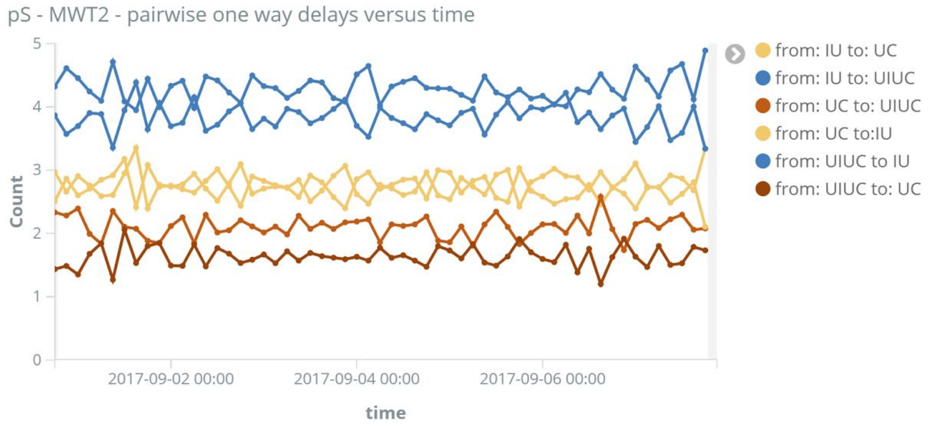
Create production size/strength alerts service

Spares

MWT2 PerfSONAR metrics



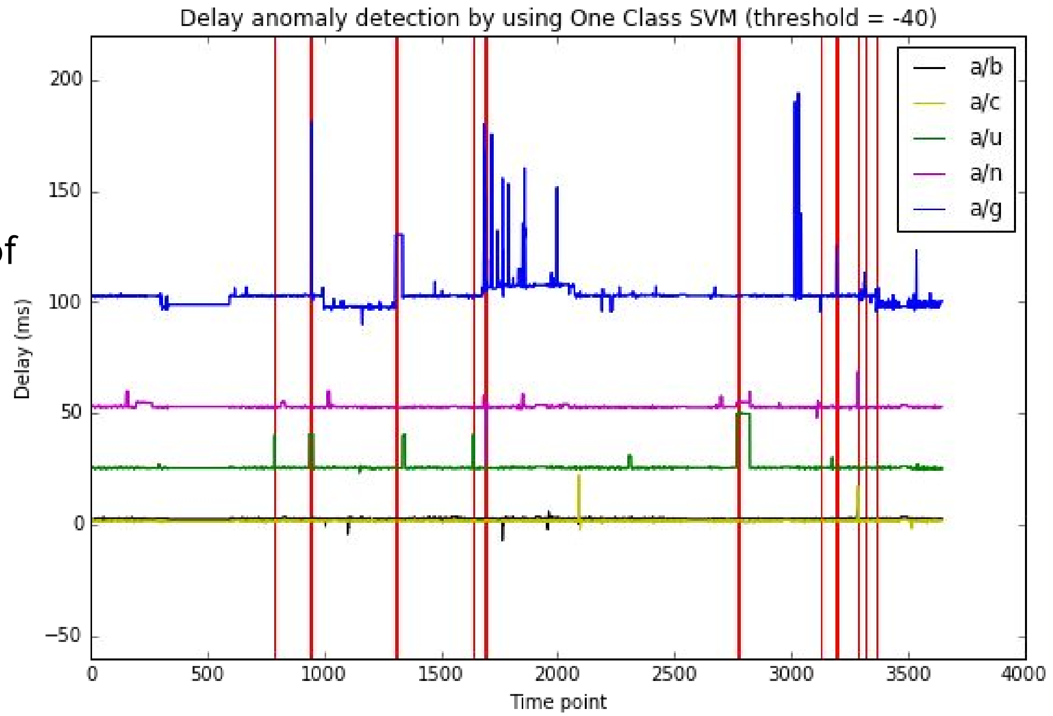




SVM for anomaly detection

First try to detect routing changes from delays:

- One class SVC
- Requires identification of the "default" good route, annotation of periods to train on. Difficult for long paths.
- Sensitive to clock synchronization issues and high utilization effects
- Adding traceroute measurements complicated by equal-cost multi-path routing (multi-IP devices, failover routing), and frequent ICMP packet drops.

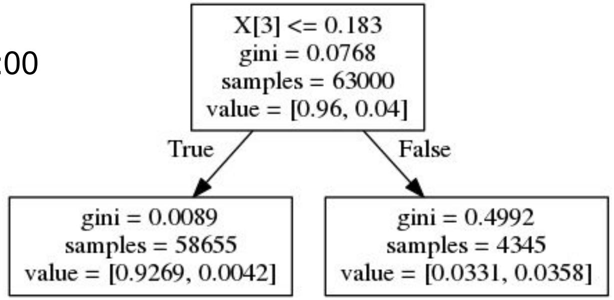


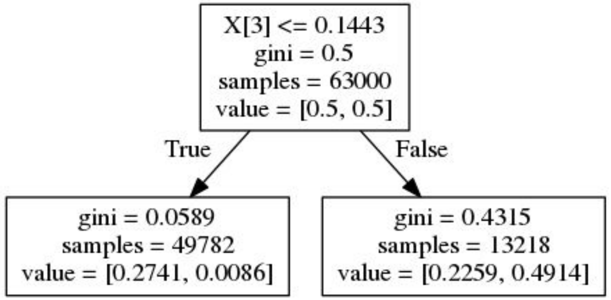
Work done by **Xinran Wang**

Showing only first two of fifty for the two test intervals.

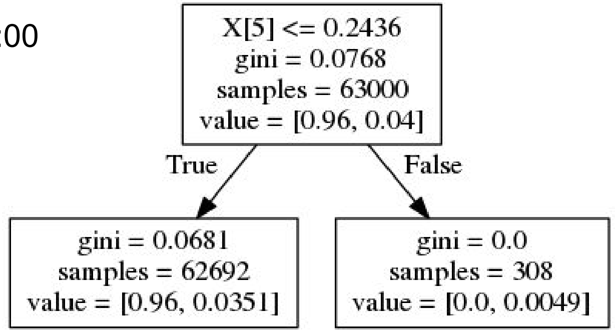
Estimators

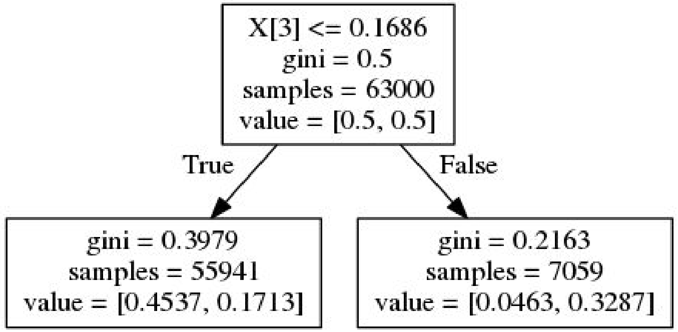
2017-08-02 07:00:00





2017-08-02 09:00:00


X[5] <= 0.2436
gini = 0.0768
samples = 63000
value = [0.96, 0.04]
True
False
gini = 0.0681
samples = 62692
value = [0.96, 0.0351]
gini = 0.0
samples = 308
value = [0.0, 0.0049]


X[3] <= 0.1686
gini = 0.5
samples = 63000
value = [0.5, 0.5]
True
False
gini = 0.3979
samples = 55941
value = [0.4537, 0.1713]
gini = 0.2163
samples = 7059
value = [0.0463, 0.3287]