Simulating network throughput by correlating perfSONAR measurements with link utilisation

Hendrik Borras[1], Marian Babik[2]

[1] University of Heidelberg, 69117 Heidelberg, Germany

[2] CERN, CH-1211 Geneve 23, Switzerland

Content

1 Executive Summary

In this traineeship the modeling of network performance through delay and packet loss measurements is explored. It is shown that correlations exists and different models are developed. Furthermore the developed models are then implemented into the existing network measurement platform and made available to general use.

At the end of this report (Part 7.3) an abstract about possible future work in this field is given.

2 Introduction

2.1 Computing at the LHC

The Worldwide LHC Computing Grid (WLCG) relies on the network as a critical part of its infrastructure to interconnect site's resources. Recently, networks have been evolving, both in terms of their speed and corresponding network traffic growth, as well as in the emergence of new technologies and paradigms. This evolution has caused rapid growth of the traffic between sites, both in terms of achievable peak transfer rates as well as in total amount of data transferred. Some of the major Research and Education Networks (NREN), such as ESnet, have seen traffic grow a factor of 10 every four years. LHC experiments have adapted their computing models to benefit from this trend by introducing a more interconnected system, moving away from strict tier-based hierarchies towards full mesh topologies, while at the same time increasing their overall traffic by a factor 2 every year (recently even larger growths have been reported by NRENs). While the scale and complexity of the current LHC network already grows rapidly, the upcoming High luminosity LHC (HL-LHC), planned for 2025 is currently projecting a factor 20 increase in the amount of disk and CPU utilization. At the moment technology development is only promising a factor 10 in performance increase until then. In order to tackle this new network technologies, advanced monitoring and machine learning will need to pave the way towards performance and cost-optimised networking. This has been recognized by the LHC experiments and projects and the "Advanced Network Serrvices for LHC Experiments"-Project (ANSE), funded by the "National Science Foundation" (NSE), is exploring the possibility to create on-demand topologies (via dynamic advance network bandwidth allocations) to increase the efficiency of transfers between LHC data centres. Some of the developed technologies have already been demoed at the last SuperComputing in 2015[1], demonstrating Software Defined Networks (SDN) at the TBits scale, which performed at single port flows of 170 Gbps (120 Gbps over Wide Area Network (WAN)) and 900 Gbps in total sustained traffic.

An essential part of the future TBit networking will be advanced monitoring and analytics capable of determining real-time performance of the different network paths, which can in turn drive the network scheduling to determine the most efficient ways to transfer PBs of data across WAN.

This report focuses on recent advances in the area of network analytics based on perfSONAR and details a concrete approach towards network telemetry, with the ability to estimate the usage of network links in near real-time based on delays and packet loss detected by the perfSONAR infrastructure.

[1] Caltech, Univ. of Michigan, StarLight, PRP, UNESP, Vanderbilt, NERSC/LBL, Stanford, CERN; ESNet, Internet2, CENIC, MiLR, AmLight, RNP, ANSP

3 Network Measurement Platform

3.1 WLCG Network

WLCG is organised in a three tier structure with CERN as the Tier 0 (T0), thirteen Tier 1 (T1) sites and approximately 170 Tier 2 (T2) sites distributed all over the globe. There are two major networks interconnecting all resources, the LHC Optical Private Network (LHCOPN) connecting T0 and T1 sites and the LHC Open Network Environment (LHCONE) connecting T1 and T2 sites. LHCOPN relies on dedicated long distance 10G and 100G Ethernet links, deployed either individually or as bundles, with routing based on Border Gateway Protocol (BGP) peerings directly managed by the connected sites ensuring that only declared IP prefixes can exchange traffic. LHCONE core service is a Layer 3 VPN (Virtual Private Network), which provides any-to any connectivity between T1 and T2 sites. It relies on a routed VPN, implemented using Virtual Routing and Forwarding (VRF) for providing a dedicated worldwide backbone reserved for High Energy Physics (HEP) data transfers and analysis.

3.2 perfSONAR

Network performance monitoring for both networks has been introduced in WLCG in 2012 by a dedicated working group, which has established a pervasive network monitoring infrastructure based on the perfSONAR Toolkit. The main goals that motivated a large-scale deployment at all sites were the ability to find and isolate network problems, characterize network usage, and provide a source for network metrics to higher level services. The choice of the open source perfSONAR toolkit was mainly driven by the consensus and prior experience of the R&E network providers who had formed the perfSONAR consortium to help develop tools in order to establish a network that would allow to identify and better debug Wide Area Network (WAN) issues. Currently, the perfSONAR toolkit is deployed at over 1600 domains around the world with 250 instances deployed and managed by WLCG. This includes instances at the major network hubs at ESNet, Internet2 and GEANT, totaling to a monitoring infrastructure with over 5500 monitored links.

In close collaboration the Open Science Grid (OSG) and WLCG have developed an extensive network measurement platform using perfSONAR. It involves central configuration and monitoring of the perfSONAR infrastructure, collection of the data measured by perfSONAR toolkits as well as central storage and the possibility to access this data publicly either by using an API (based on ESNet ESMOND) or by subscribing to a real-time message stream (based on STOMP ActiveMQ). The platform was deployed and is operated in production since September 2015 as part of the OSG network service.

3.4 Network Analytics

With the measurement platform in place, various groups have started using in order to develop insights into the way our networks operate, thus exploring how we can become more efficient and make the most of our existing capacities. Recently, the biggest efforts were focused on network analytics and covered the following topics:

- Real-time analytics to detect “obvious” issues with the networks as they arise, using both empirical methods as well as machine learning.
- Detecting how well network paths perform. This is based on computing a network cost-matrix for all existing links and using it to optimize job and data placement, job scheduling, etc., when there is a choice. This project is part of the effort.

- Automated debugging of network issues and helping to find root causes in real-time (PuNDIT).

4 Conceptual approach to network performance measurements

4.1 Network cost matrix

WLCGs High Throughput Computing (HTC) model relies on network performance to transfer significant amounts of data between sites in a short period of time. Thus the primary focus is on the achievable network throughput. In order to obtain an overview of the performance for all existing links, a network cost matrix has been introduced to determine the status of all links between all sites. Computing a network cost matrix is grounded on the ability to determine the performance of a given network path, which is in turn determined by many different factors such as link capacity, latency, message transport unit (MTU), available buffers on the sending/receiving host, operating system, efficiency and available memory of the equipment along the path, etc. It therefore poses a substantial challenge.
From the network operations point of view, data on the link utilization can be gathered directly from the routers via certain protocols such as SNMP, NetFlow, SFlow. While this approach provides the best estimate of the actual traffic seen by the routers, it rarely gives an estimate of the actual end-to-end performance which involves also passive network equipment (switches) and well as both end hosts. In addition, particular the readout is dependent on the actual implementation and configuration at the site, which usually includes significant layer of virtualization and is therefore very challenging to obtain in a federated WAN environment, such as the one used by WLCG. The other option is to obtain an estimate by running a synthetic end-to-end network transfer between two hosts, with tools such as iperf, iperf3, nuttcp, etc. While this provides an accurate view on the available bandwidth at hand, the actual measurements, if done on a regular basis, will not exceed few per day given the number of links in LHCONE. Another popular approach among the experiments is to use the production data management systems directly and obtain an estimate of the achievable bandwidth from the production transfers[2], however since this usually involves disk-to-disk transfers and data management system itself (with protocol overheads, etc.), it's unclear what fraction of the measurement can be attributed to the actual network performance. In addition, there are usually several data management systems operated on the same link, competing for the same resources without any scheduling, which in practice leads to significant fluctuations in the measurements.

4.2 The adopted approach

Our approach leverages from the perfSONAR measurements that can be done at high frequencies, such as one-way delay (OWAMP) measurement, which is measured by default at 10 Hz (600 packets/s) and was specifically designed to detect abnormal network behaviors such as packet loss (up to 0.0016%), packet reordering, jitter or significant path changes by detecting number of hops the packet has travelled. Since those are also the main attributes that determine the achievable throughput on the link, we have explored if we could combine them in a way that would allow us to see events when link is approaching its saturation. Sufficiently precise one-way-delay measurements would provide enough information on the devices along the path that are close to saturation and would either start filling their buffers and thus holding the packets for a little big longer or if out of buffers would start to drop the packets. Our primary focus was thus on detecting higher than usual latencies as well as any possible packet loss, which even at very low numbers (0.001%) is very detrimental to the achievable network throughput.

[2] As production transfers usually don't have regular pattern, coverage is improved by running additional "synthetic" transfers.

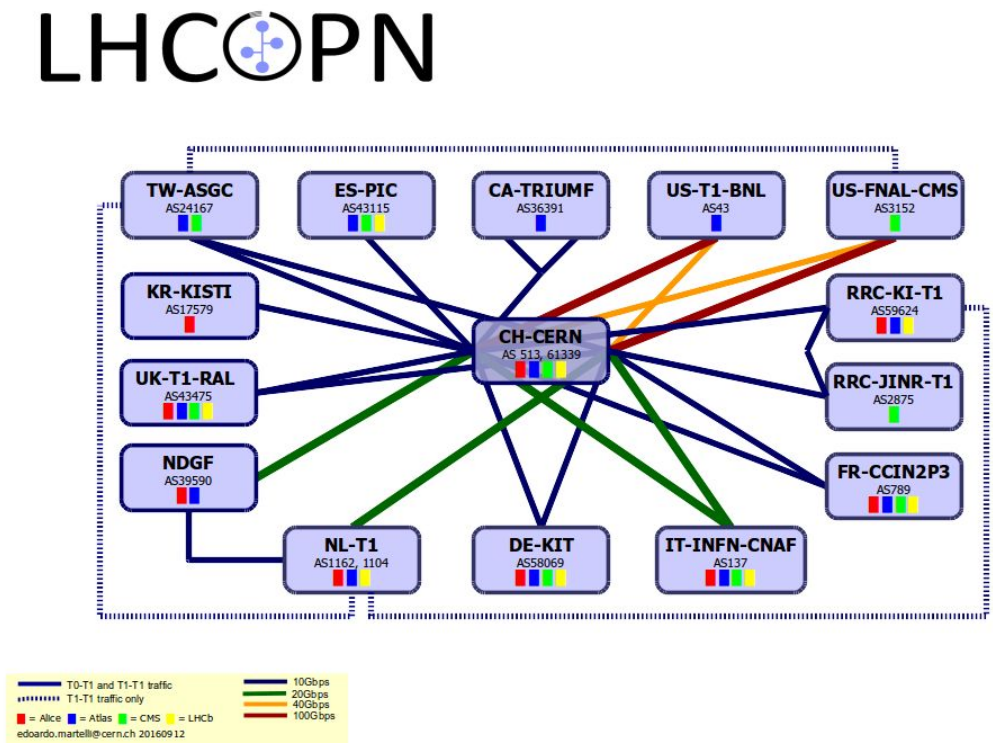
5. Building a model

5.1 LHCOPN

In order to have a solid basis with reliable data for building our model we had to choose a network for which we have reliable measurements and a good understanding of the underlying topology. As discussed, throughput data directly from the routers is likely the most precise data we can get. In terms of accessibility and simplicity the LHC optical private network (LHCOPN) fits these requirements nicely. LHCOPN is a network with direct dedicated connections from CERN to the TIER-1 data centers. The topology is starlike and has for each connection only one hop, which is the service provider in between. For each of those connections there is an accurate assumption about the total available bandwidth.

All the routing for this network is done by two routers at CERN, which are indicated by the two connection ends at CERN in the diagram below. The throughput data for both of these routers has been provided by the IT “Communication Systems”-Group (IT-CS) at CERN.

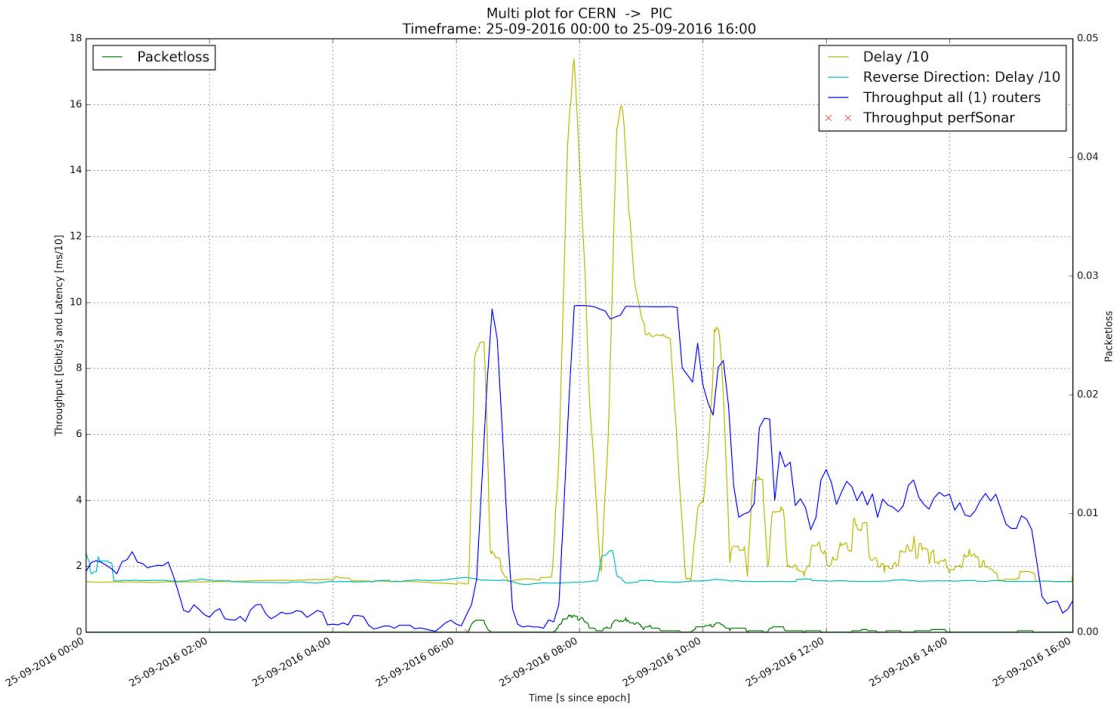
With this precise data form LHCOPN and it’s simple topology we are confident to validate whether the measurements from perfSonar are precise enough to to see congestions in the delay and/or packet loss.



5.2 Observations in our measurements

When looking at the measurements from perfSonar and the routers for the first time many things came to our attention.

The plot below shows part of the recorded data for the connection from CERN to the Tier 1 center “Port d’Informació Científica” (PIC). CERN and PIC are connected via a 10 Gbit/s dedicated optical link. In the plot the router data is shown in five minute steps (blue) and data from perfSonar (yellow, cyan, green) in one minute steps, both data sets have been smoothed via a moving average over 15 minutes in order to show tendencies clearer and filter high frequency noise.
Most interestingly delay (yellow/green) and packet loss (green) seem to react strongly once the link utilization (blue) comes close to it’s design maximum. Even though this reaction is strong, there is close to no warning in advance, that a congestion is about to happen. Similar patterns but with different strengths can be seen on all connections within LHCOPN.

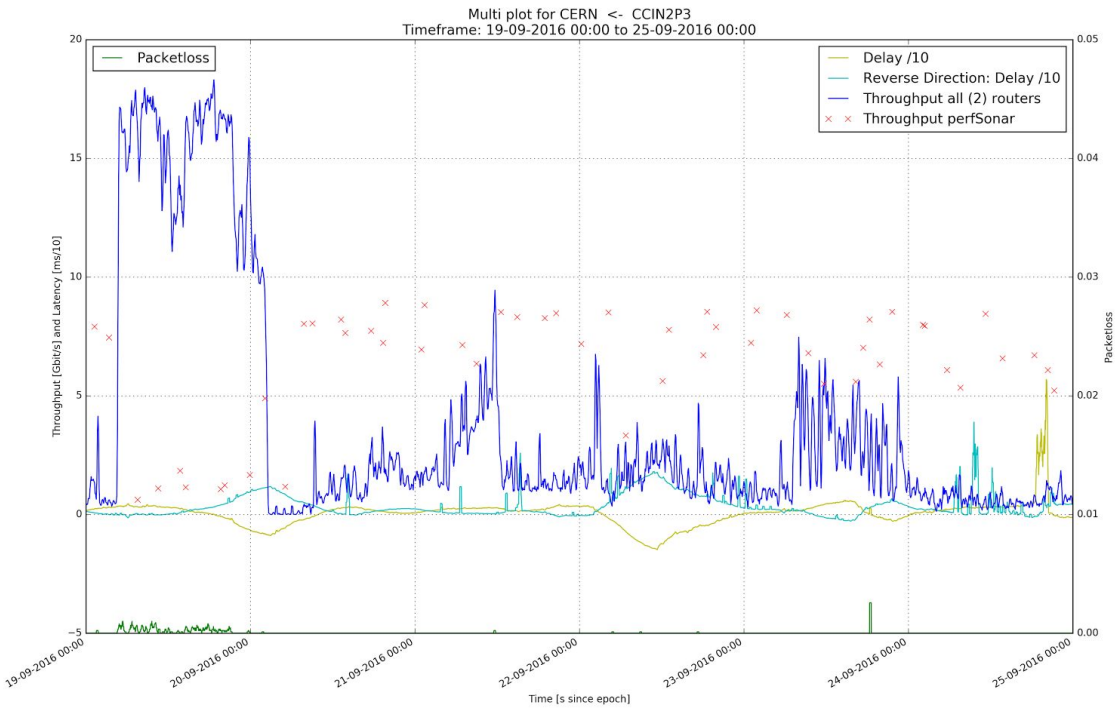


In the plot below the connection from the computing center of the “Institut national de physique nucléaire et de physique des particules” (CCIN2P3) to CERN is shown. The connection with CCIN2P3 has a maximum throughput of 20 Gbit/s. Additionally to the plot before, throughput measurements done by perfSonar (red crosses) are being shown. Those measurements are clearly reacting to high link utilization. Still, they show the limitation of the throughput measurements done by perfSonar. On one side they are done relatively sparsely every few hours on the other side they don’t use the link to it’s fullest when it is unused. Especially the first limitation makes those measurements unfit for the cost matrix we want to construct, as we want to provide services with data that is as close to real time as possible.

CCIN2P3 is stationed at Lyon relatively close to CERN. Normally we would expect delays of about 6 ms on such a connection. What we see indeed is mostly oscillating delay times in both directions, going even into negative ranges. Most notably the seen drift in delays is symmetrically for both directions. This phenoma has been described by Xinran Wang and Ilija Vukotic (
https://docs.google.com/presentation/d/1cvmR7nfCE8RzlHQ2LsVOERwgEPMQrg8EgWBu3t

m45cc/edit?usp=sharing Slide 30 to 40). It is proposed that this is most likely a clock synchronization problem and can be fixed by adding more ntp servers to the site. It could be, that CCIN2P3 has only one ntp server, which we suspect could be causing these oscillations. This limits or ability to model connections to sites, which are close by.

Another interesting phenomena can be seen in the plot on the 24.09.2016 at about 17:00. At this time there is a rather big spike in delay, although the connections delay didn't notably react to the near congestion on the 19.09.2016. This spike with its characteristic shape can be seen across all incoming connections to CERN at the same time. Additionally repeating every week. We therefore concluded that this is likely the result of network maintenance on CERNs side.



Summarizing, this shows that perfSonar measurements are precise enough to make high link utilizations visible in packet loss and delay. We now needed to find a way to couple delay and packet loss in such a way that we can produce an estimation of the current link utilization.

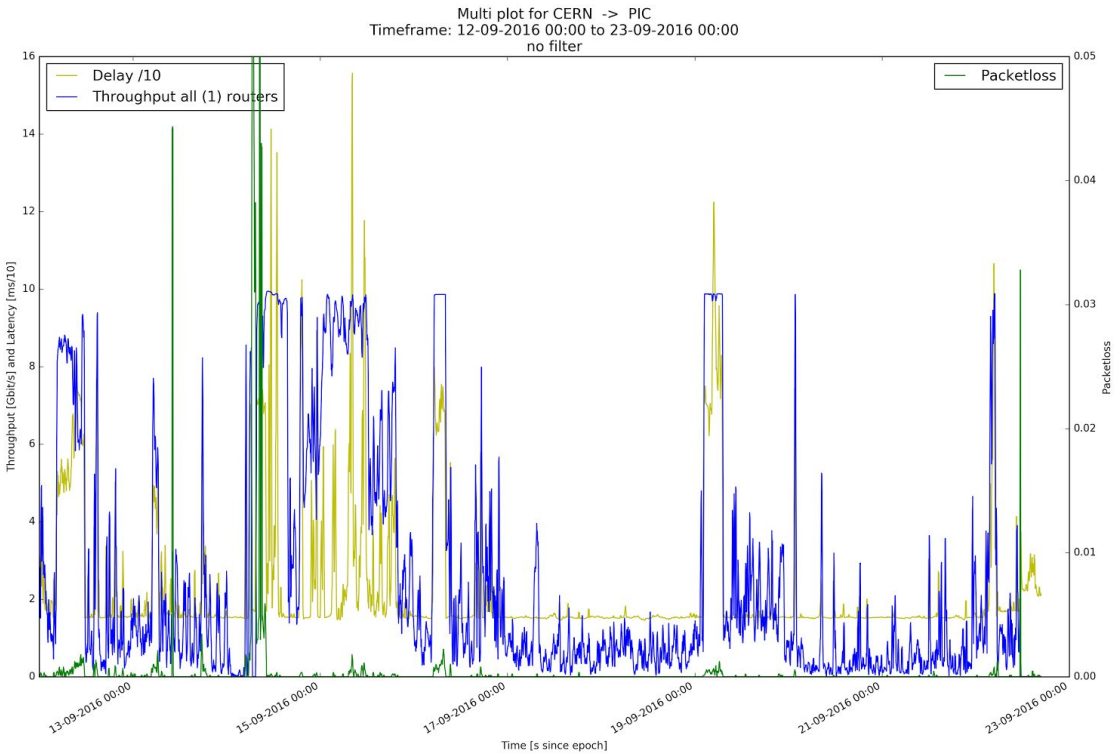
5.3 Empirical model

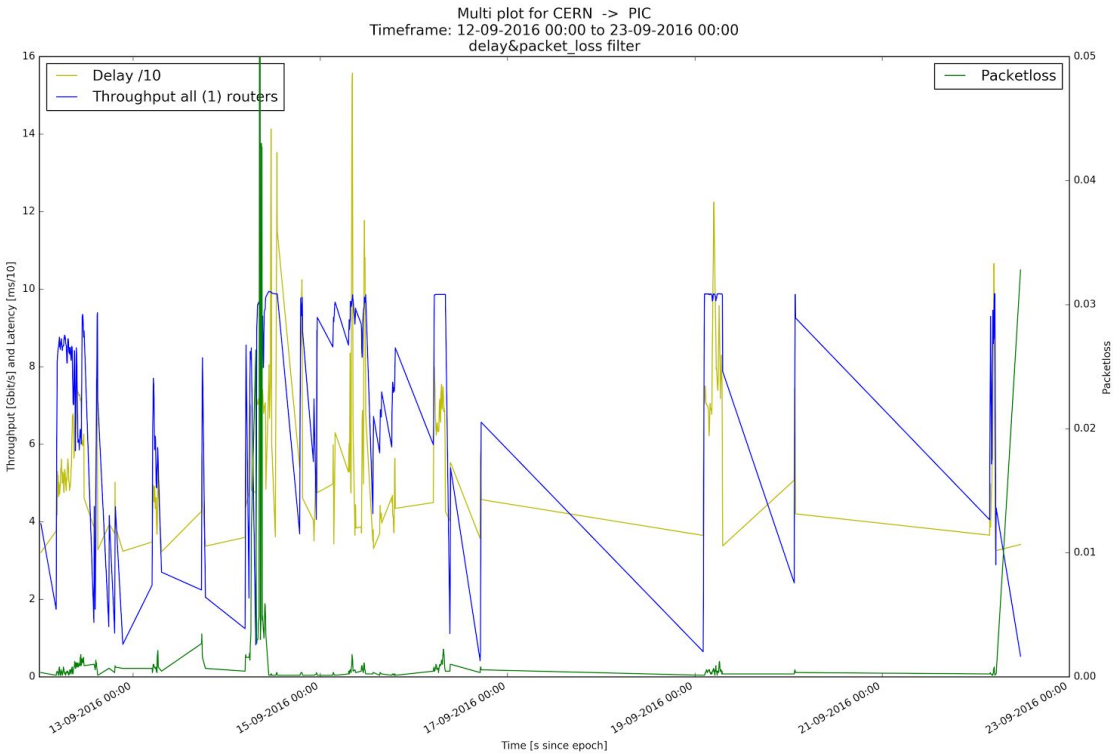
At first we wanted to see whether it would be possible to build a simple empirical model based on filters and a formula. This is particularly interesting as it not only explains how packet loss and delay are coupled while we see congestions, but it is as well potentially fast to compute, when we later use it in practice.

5.3.1 Selecting interesting data

Starting with the filters we wanted to select data, which is interesting to us, e.g. data which shows congestions. Those filters are required to only be based on delay and packet loss measurements from perfSonar, as we won't have router data for every connection in the production system.

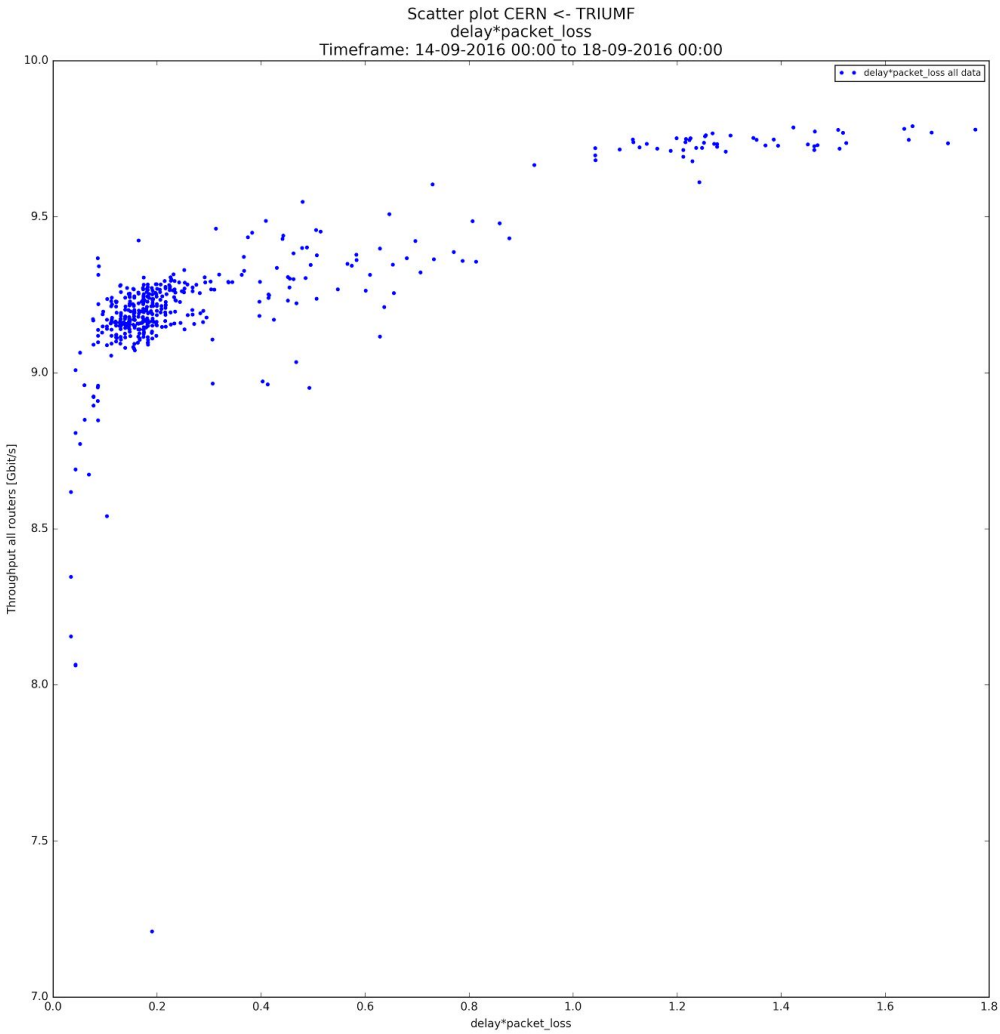
For this I developed two filters.
Filter 1 is based on the assumption, that any data points which have packet loss are interesting. So this filter will select data points which have a packet loss above 0.01%. This filter will not only see congestions but, as well grab data, which has packet loss due to randomly occurring packet loss.
Therefore the second filter looks at the delay. It uses the minimum and the standard deviation of the data over the whole measured time frame and checks if the current data point is outside of the standard deviation around the minimum. When this is the case the data is considered interesting, since there is definitely something going on with the delay. Both filters are coupled with an logical and, so that only data which looks interesting to both filters gets selected. In the plot titled "no filter", the raw data for a long time frame for CERN to PIC is shown in a similar manner as before.
In the Plot titled "delay&packet_loss filter" only the selected data is kept. The long straight lines are connecting points, where there is no data in between. One can see that both filters together seem to work good, as nearly all points with actually high link utilization are being kept. I will later further discuss the efficiency of those filters.



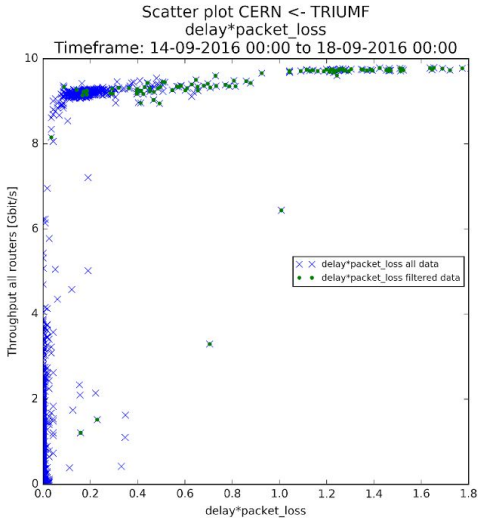


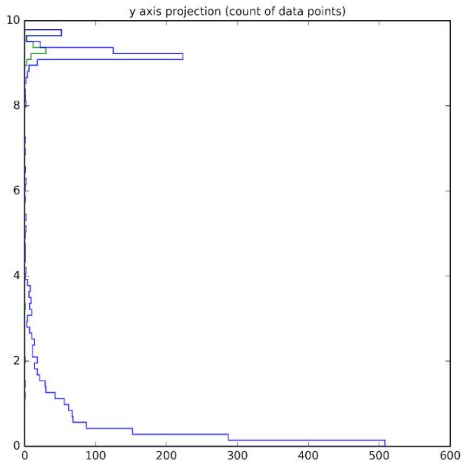
5.3.2 Interplay between throughput, delay and packet loss

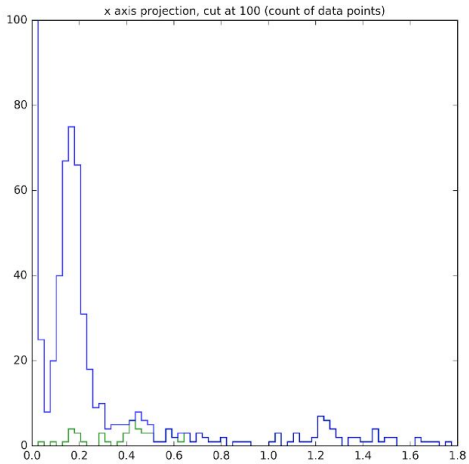
In order to look for correlations in our data we used scatter plots. At first these plots for throughput versus packet loss or delay rarely showed any correlation. Later the approach of multiplying delay and packet loss and plotting it versus the actual throughput was showing better results.
In the graph below such a scatter plot for the connection from “Canada’s National Laboratory for Particle and Nuclear Physics” (TRIUMF) to CERN is shown. The x axis represents delay*packet_loss, while the y axis shows the actual throughput measured by the routers. The connection has a maximum bandwidth of 10 GBit/s. It is visible that there seems to be a correlation from about 85% to 100% of bandwidth usage. Such correlation is visible on all connections incoming to CERN.
Unfortunately this correlation is not visible on any of the connections that are going from CERN and are outgoing rather than incoming. We suspect that this is a systematic error, which results because of the way we measure our throughput data.
Throughput in both directions is measured at the routers at CERN, thus for the outgoing connections we see the outgoing buffer of the router, which is in front of the congesting connection. While the incoming data is measured with the incoming buffer of the routers at CERN, which sees the data after the congestion had already happened and not before as for the outgoing direction, which seems to affect the data, that we see. Whether this is the real cause for this systematic error could be further studied by acquiring router data from the connected TIER-1 computing centers. Apparently we didn’t have the time to further investigate the cause.
Due to this error being very likely a systematic problem and not statistical we concluded, that delay multiplied with packet loss is indeed a simple estimation for the usage of the connection, when it is close to congesting.



In order to further investigate how our filters for the selection of data performed we compared the filtered data to unfiltered data points. In the plot below a scatter plot similar to the one above is shown. It utilizes the same connection and timeframe as above. For each axis of the plot there is a projection in the form of a stacked histogram. Unfiltered data is marked in blue in all plots and filtered data in red.
Looking at the projection of the y axis one can clearly see that data with low throughputs is efficiently filtered out. Compared to this data with high throughput is caught rather often. The efficiency how much high throughput data is seen of course varies with the characteristics of each connection, but is seemed to work reliable for all connections in LHCOPN.


Scatter plot CERN <- TRIUMF
delay*packet_loss
Timeframe: 14-09-2016 00:00 to 18-09-2016 00:00
Throughput all routers [Gbit/s]
delay*packet_loss
delay*packet_loss all data
delay*packet_loss filtered data


y axis projection (count of data points)


x axis projection, cut at 100 (count of data points)

5.4 Machine learning model

5.4.1 Introduction

After the empirical model now produced reliable data we wanted to try out another approach to getting assumptions about the current usage. For this we chose to train a machine learning model. As the model we chose a simple neural network, as it is able to learn linear and nonlinear correlations alike and as well can do classification, what is useful for our case.

5.4.3 Structure and inputs to the neural network

The used neural network is a feedforward network with three hidden layers, with a decreasing number of neurons and one neuron in the output layer. The chosen activation function is the rectified linear unit. It was chosen over the often used tanh as it produced better overall convergence and proved to be robust against overshooting into negative output values.

As input the neural network (NN) received the current delay and packet loss and 15 measurements from the past in one minute steps. As in the graphs before this input data has as well been smoothed via a moving average. Additionally in order give the NN an understanding of how a link normally behaves it received the average, standard deviation and minimum for delay and packet loss over the whole inspected area.

5.4.4 Training the model

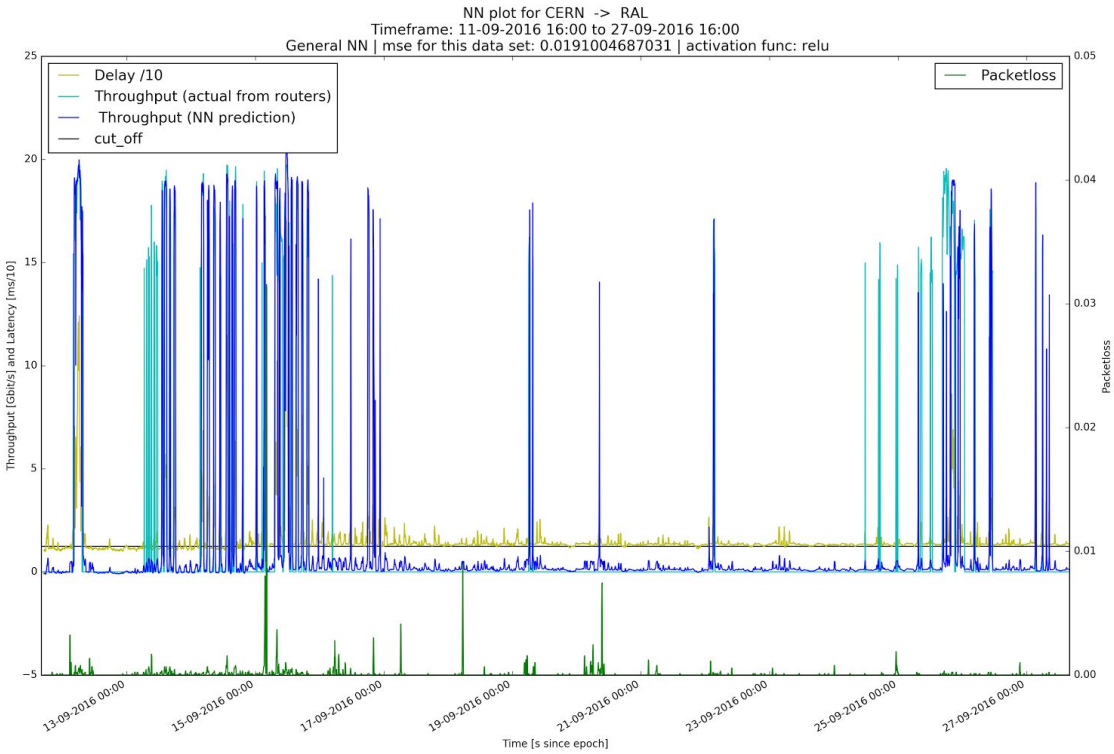
The NN was then trained to predict whether a connection is used below or above 70% of it's total link capacity and if it's above 70% usage the NN should do a regression between 70% and 100% link usage. The training then took place on all connections available except RAL.

Training one neural network for each connection would have been possible is possible as well. We did in fact test this method with good results, but later discarded it as it would have been only applicable for LHCOPN, where we have router data. For for the rest of our connections in LHCONE this would have not worked as we have no router data for most most of the connections. Without reliable router data it would have not been possible to train one network for each link. In contrast a general network for all links in LHCOPN is likely to as well produce satisfactory results on LHCONE.

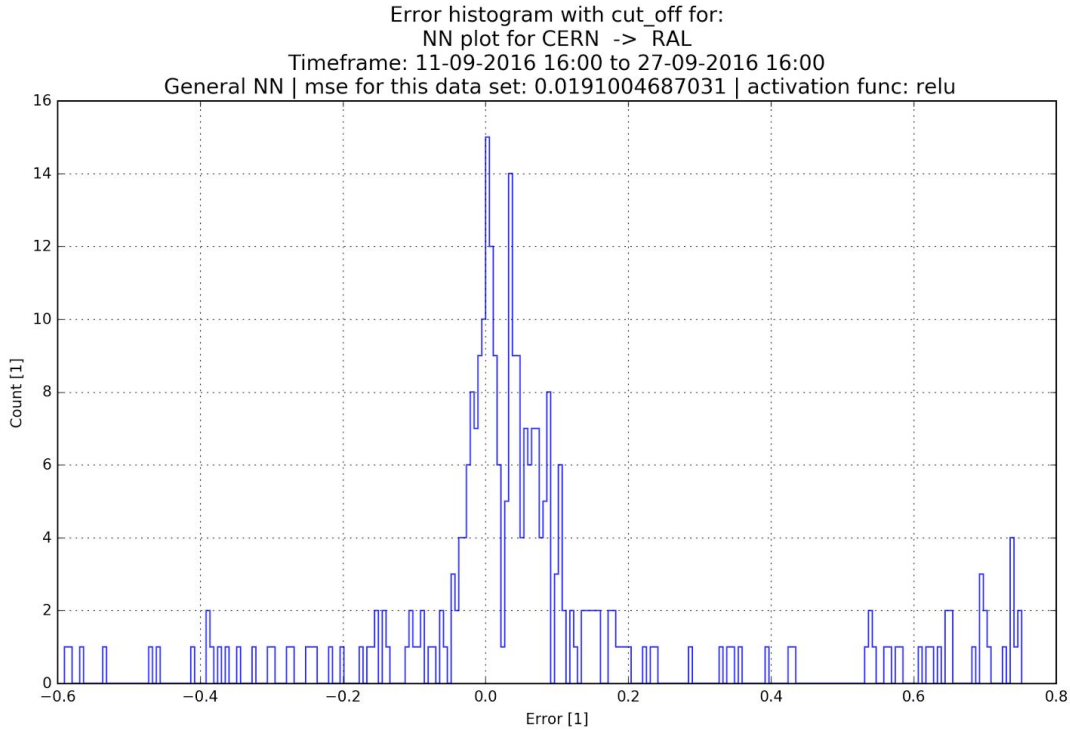
5.4.5 Result verification

In order to verify how good the NN performed two methods were used. One was to look how well the network then performed on the connection to RAL and the other was to look how well it would perform on a later timeframe for all connections except RAL. So to say the verification was done once on completely new data and once on new data for the same connections as the network was trained on.

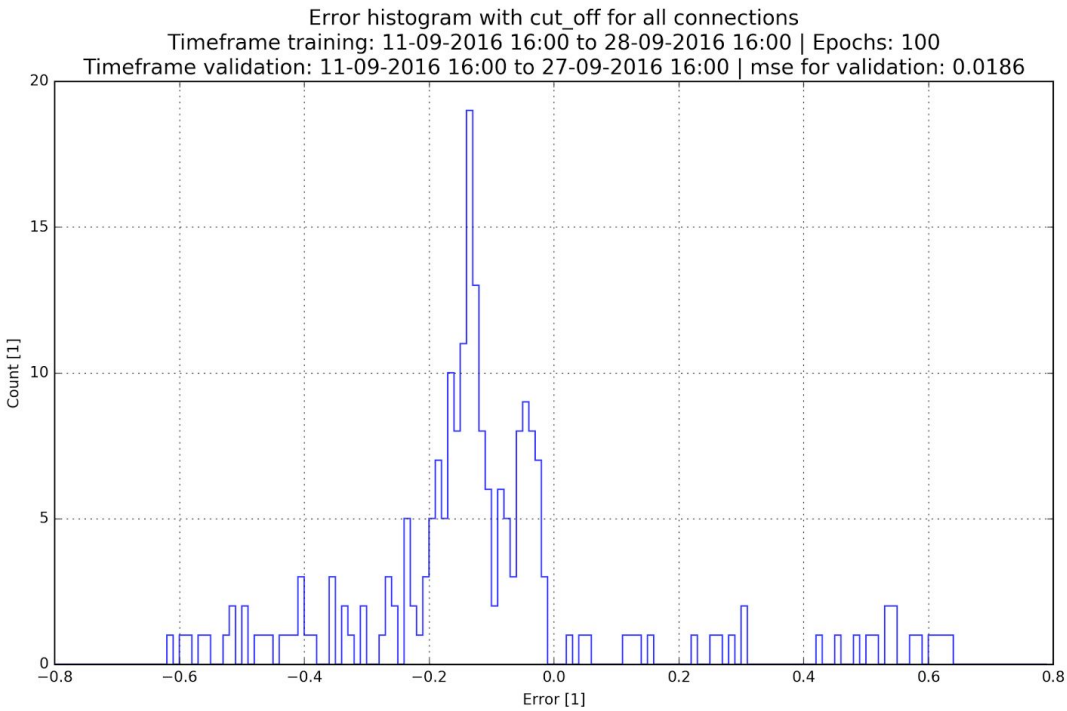
Below the plot for the first verification method is shown. The prediction (blue) seems to be often close to what the actual throughput (cayan) is showing. On this connection, which is completely new to the neural network it seems to correctly indicate most of the congestions. It still doesn't find all of them. To quantify this the mean squared error of 0.019 is indeed quite low.

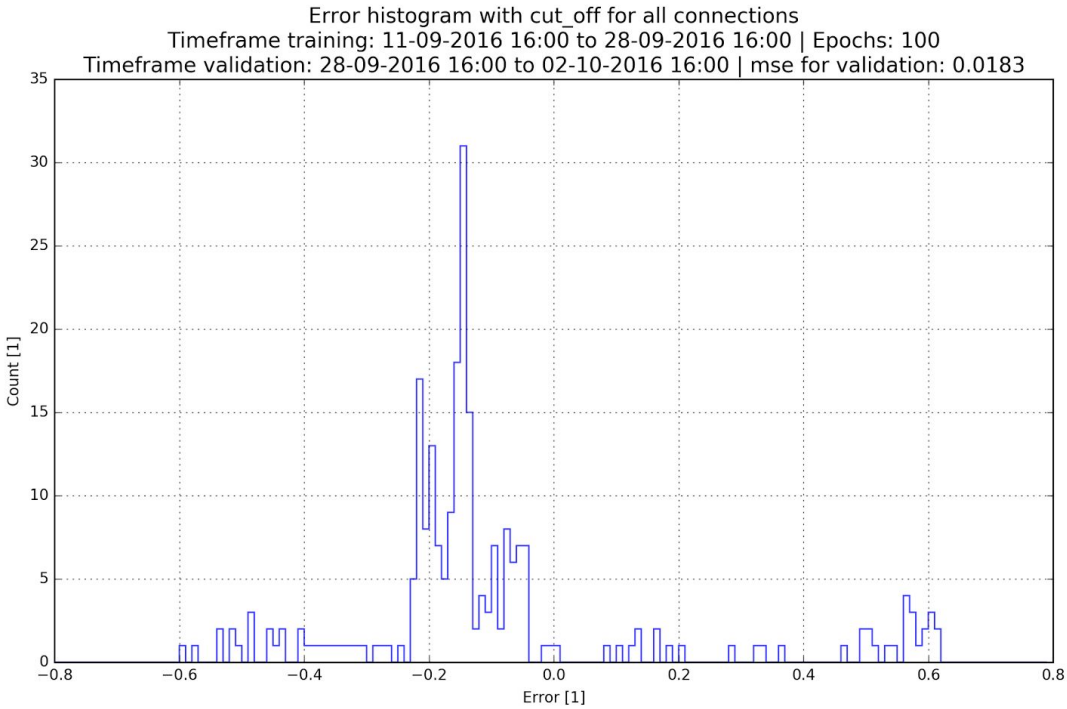


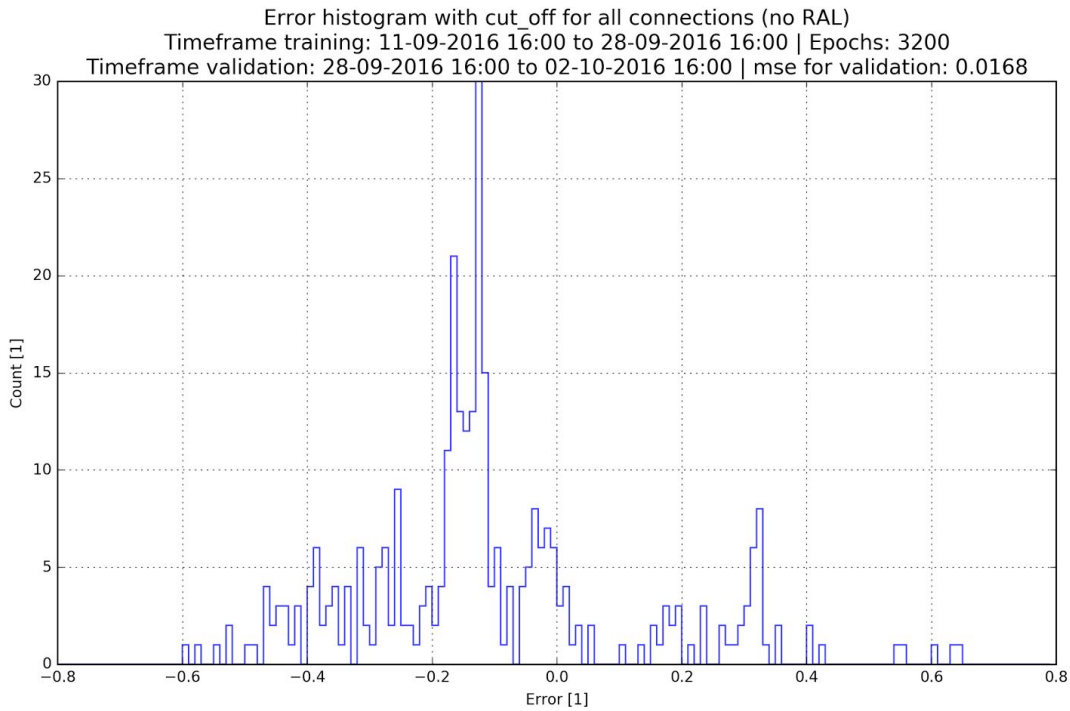
Below the distribution of errors for the verification on RAL is shown. Here the error is given in a measure of percent, where 0.8 means the NN is 100% above the real value. The distribution is nicely centered around 0 and has an acceptable width. From this we concluded, that the neural network is likely not overtrained and predicts most of the spikes sufficiently correct.



Further below the same type of plot is shown for the second verification method. Here the training timeframe spanned from the 11th September 2016 to the 28th September 2016, while the verification utilized data from the 29th September to the 2nd October. The distribution is centered around -0.15, which corresponds to the neural network normally underestimating the current throughput by ~18%. The width of this peak at -0.15 is relatively wide and indicates, that we have not overtrained the network. At the moment it is not quite clear why the network does this characteristic underestimation of about 18%. The classification to divide data between the link utilization being above or below 70% still seems to work though.
One explanation for this behaviour could be, that the regression in the 70% region is not working properly and that the network is only targeting one specific point above 70% and not doing an actual regression. Further investigation to find the cause is needed.


Error histogram with cut_off for all connections
Timeframe training: 11-09-2016 16:00 to 28-09-2016 16:00 | Epochs: 100
Timeframe validation: 11-09-2016 16:00 to 27-09-2016 16:00 | mse for validation: 0.0186
Count [1]
Error [1]
0
5
10
15
20
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8


Error histogram with cut_off for all connections
Timeframe training: 11-09-2016 16:00 to 28-09-2016 16:00 | Epochs: 100
Timeframe validation: 28-09-2016 16:00 to 02-10-2016 16:00 | mse for validation: 0.0183
Count [1]
Error [1]
0
5
10
15
20
25
30
35
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8



5.5 Assessments on model performance and use cases

All in all both models produced interesting and usable results.

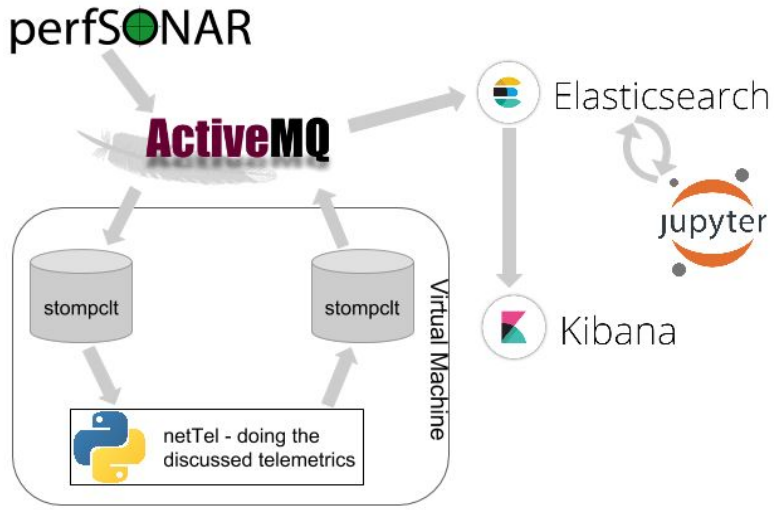
The empirical model is able to perform quite good predictions for the bandwidth usage above 85%. And is able to filter out interesting data points. While this works good. The model has one big limitation, which is that the produced values are only an indication for the performance of one link for itself and are without further processing not comparable to other links. This is because each link is characteristic in how strongly it reacts to congestions. As example the delay of PIC is heavily influenced, while the delay for TRIUMF and CCIN2P3 the is only impacted on a low level.
In order to now compare certain links to others one can as example use the output as a binary indicator. Telling, that if the output is non zero there is something wrong with the link or that the link is close to congesting. Already this would certainly help a lot of services in deciding which paths to choose within LHCONE. Furthermore the output for each link could get normalized and thus be made comparable to other links at a higher granularity.

Compared to the empirical model the model based on machine learning is filtering unimportant data good as well. In contrast to the empirical model it additionally produces a value on how sure it is that the link usage is at least at 70% and then does a regression on how high the utilization is above 70%. Though this regression seems to not work perfectly well at the moment, this approach makes links comparable and classifiable within the produced network cost matrix.

6 Implementation of the network cost matrix

6.1 Architecture

The implementation of the discussed models into a cost matrix was done in the form of a stream and message based approach.
The image below shows a picture of the architecture. The program doing the telemetrics has been named netTel (for network telemetrics) and is written in Python. Following the flow of one measurement, which is first produced by perfSonar (indicated by the perfSonar logo). The measurement then get collected and is posted as a message the an ActiveMQ message broker at CERN, called netmon, indicated by the ActiveMQ logo. This message broker receives all measurements provided by perfSonar. To this message broker clients can connect and recieve those messages containing the raw or summarized data. One example for such a client is the Elasticsearch instance maintained by Ilija Vukotic in Chicago. There the data gets stored and can be visualized by Kibana (example for CERN and RAL) and is further accessible for analytics on a jupyter instance. In fact I have done a lot of my work this jupyter instance, which has a lot of computing power available, including two GPUs currently configured for accelerated training of neural networks. Of course not only Elasticsearch can get data from the message broker, but any other client can register as well. For our use case we first set up a virtual machine (VM) based on Cern CentOS 7 which hosts the needed programs, libraries and frameworks. Then an instance of stompclt was used to receive messages from the ActiveMQ message broker as a client. The received messages are then read by netTel. As soon as netTel starts processing those messages it starts buffering the raw data. And oncehtese buffers which fit 32 minutes of raw data are full, netTel smoothes the data and calculates the empirical and machine learning model, as discussed in Part 5. The results of this calculations are pushed back to another stompclt instance, which sends them back to the ActiveMQ message broker. From there on they can be processed results can be processed by anybody in the normal way, as example with Elasitcsearch.



6.2 Used technologies

netTel is written in pure Python. It uses DataFrames from the library pandas for storing raw and smoothed data. As pandas is based on numpy, numpy arrays are used as well occasionally to do calculations, especially for the empirical model. As a framework for machine learning I originally started with TMVA, which is a machine learning framework built into the widely used analysis framework ROOT. But after some time it got clear to me that the single threaded approach made by TMVA wouldn't scale and that my code could profit from a framework which is built for python

from the ground up. So as we decided to test neural networks I started to use a deep learning framework called Keras. Keras is built on top of Tensorflow and/or Theano, which are both similar tensor manipulation libraries aimed to be highly scalable and use CPUs and GPUs to their fullest for calculations. In this particular case I used Theano as the backend for Keras as it was already installed on the jupyter instance in Chicago. While the neural network was trained on the jupyter instance and not in netTel itself, netTel of course uses Keras in order to utilize the trained neural network and implement the machine learning model. On multiple occasions the machine learning library scikit-learn was used as well. It was not suited for implementing the neural network, but proved useful for pre-processing the data and normalizing the inputs used by the neural network.

6.3 Implementation characteristics

netTel ended up to have a very small footprint on our production system. With full buffers for all ~5600 connections in LHCONE the memory usage is normally around 270 MB, while probably a lot of it still falls to the overhead caused by Python itself. netTel is able to process around 30 messages per second, which is three times as much as the nominal input from stompclt. While running normally it uses one CPU with a load of about 30%. As netTel first needs to fill its buffers before it can start outputting results in near real time it has about a 40 minutes of ramp up time until the number of output messages converges to the number of input messages.

7 Conclusion and Outlook

7.1 Main results on network performance measurements

Within the short timeframe of this traineeship it was possible to show, that network performance measurements based on delay and packet loss measurements done on a regular basis are possible. Indeed it was shown, that measurements currently performed by perfSonar are sufficiently precise to do this. Furthermore we saw, that simple correlations and machine learning methods are good enough to generate assumptions about the current performance of each link.
Indeed the developed models scale to be utilized on all LHCONE links with a near real time precision and further improvement on those models, especially the machine learning model should be easily possible with little to no impact on the performance of the implementation.

7.2 Next steps

As telemetry data is now being published to our production service at CERN it would be interesting to interface with other services in order to see how they can profit from this and especially how big a performance improvement would be. On another side the data from the empirical model can be used to detect anomalies on all links, so that it would be possible to see when sites have broken or badly configured equipment in place.

As the models have now been validated for LHCOPN further validation on LHCONE is needed to be done. We can then see how good the models perform in more complex network structures.

At the same time further investigation on the models at the previously mentioned points would definitely improve the overall understanding and performance of both models. Especially more work on better training the neural network is needed in order to improve it’s performance and rule out the uncertainties we have at the moment.

While netTel will perform nicely and as foreseeable without issues for quite some time it won't easily scale beyond its current processing limits. Here a new approach in going to a highly scalable computing platform such as Hadoop, SPARK or DataTorrent is needed in the future.

Currently the follow up on these steps is planned to be done by the WLCG Network Throughput working group. As at the moment the resources for this are sparse a collaboration with another institute on this topic is highly welcome.

7.3 Potential future projects

8 Acknowledgements

I would like to thank my supervisors Marian Babik and Markus Schulz for their extensive help and support with this work. Every discussion we had was valuable and pushed me and the project further. As well in the daily problems of work their knowledge and openness has always been valuable to me.

I would like to thank Ilija Vukotic and University of Chicago for providing extensive computing resources for this work. Not only this but I was able to learn a lot from the ongoing discussions with Ilija on machine learning and on this project.

As well I would like to thank Shawn McKee from University of Michigan for the weekly discussions and helpful comments on my work.

9 Glossary

- [Whenever something not self-explaining is mentioned]
- Instances
- Sites
- VOs
- Tiers
- FTS
- xRootD
- OSG
- ActiveMQ
- ntp