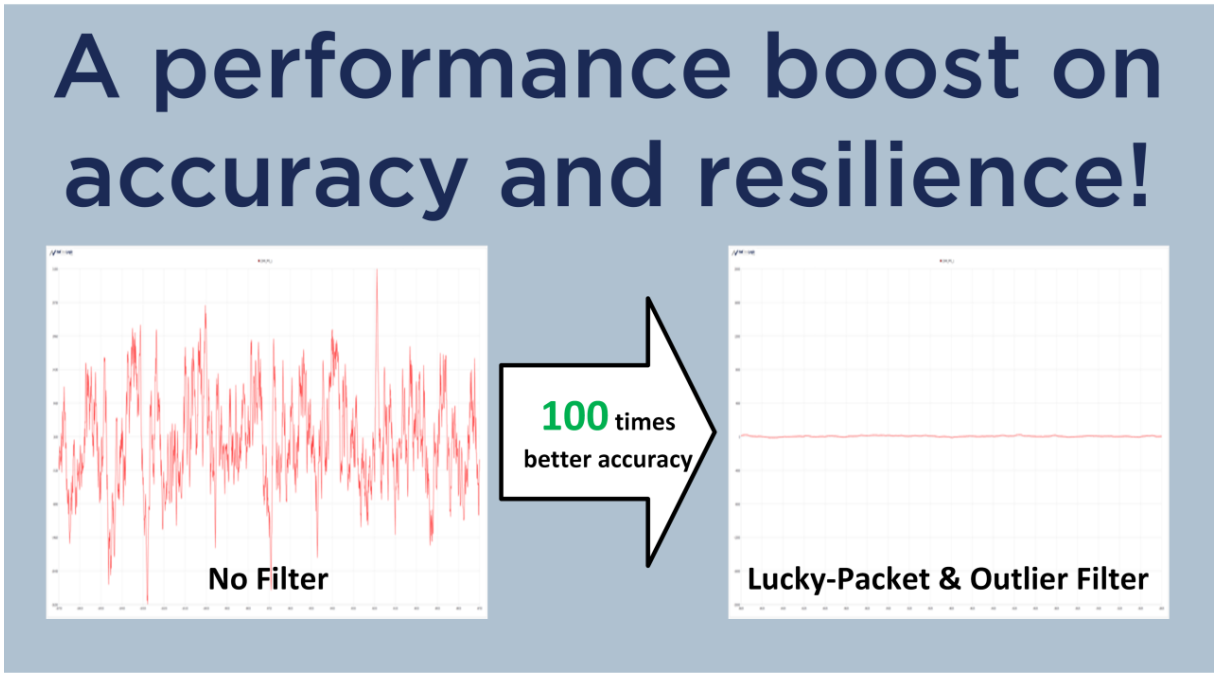


Lucky-Packet and Outlier Filters: a way for increased synchronization accuracy and resilience?

2. April 2025

In the last weeks we had again a look at how to make PTP (or synchronization protocols in general) more resilient and accurate.

Basically, we looked at two well-known mechanisms: The Lucky-Packet Filter and the Outlier Filter. Both are well-known techniques to get better results from a noisy measurement by analyzing larger numbers of measurements.

When it comes to PTP, these techniques are widely adapted especially in Telecom networks. Why specifically Telecom Networks you may ask, there are still a lot of brown field Telecom Networks, which are unfortunately not PTP aware. In these non-PTP aware networks, PTP frames experience a high Packet Delay Variation (PDV) due to the non-deterministic behavior of network equipment like switches etc. which leads to bad synchronization accuracy if no special precautions have been taken. This also because PTP was designed for engineered networks, where

all network equipment is PTP aware (PTP Boundary Clocks, PTP Transparent Clocks).

To still get a pretty good synchronization over such jittery networks, additional filtering is required and this brings us back to the Lucky-Packet and Outlier Filters.

But first we should have a look at a simplified explanation of the two filter concepts:

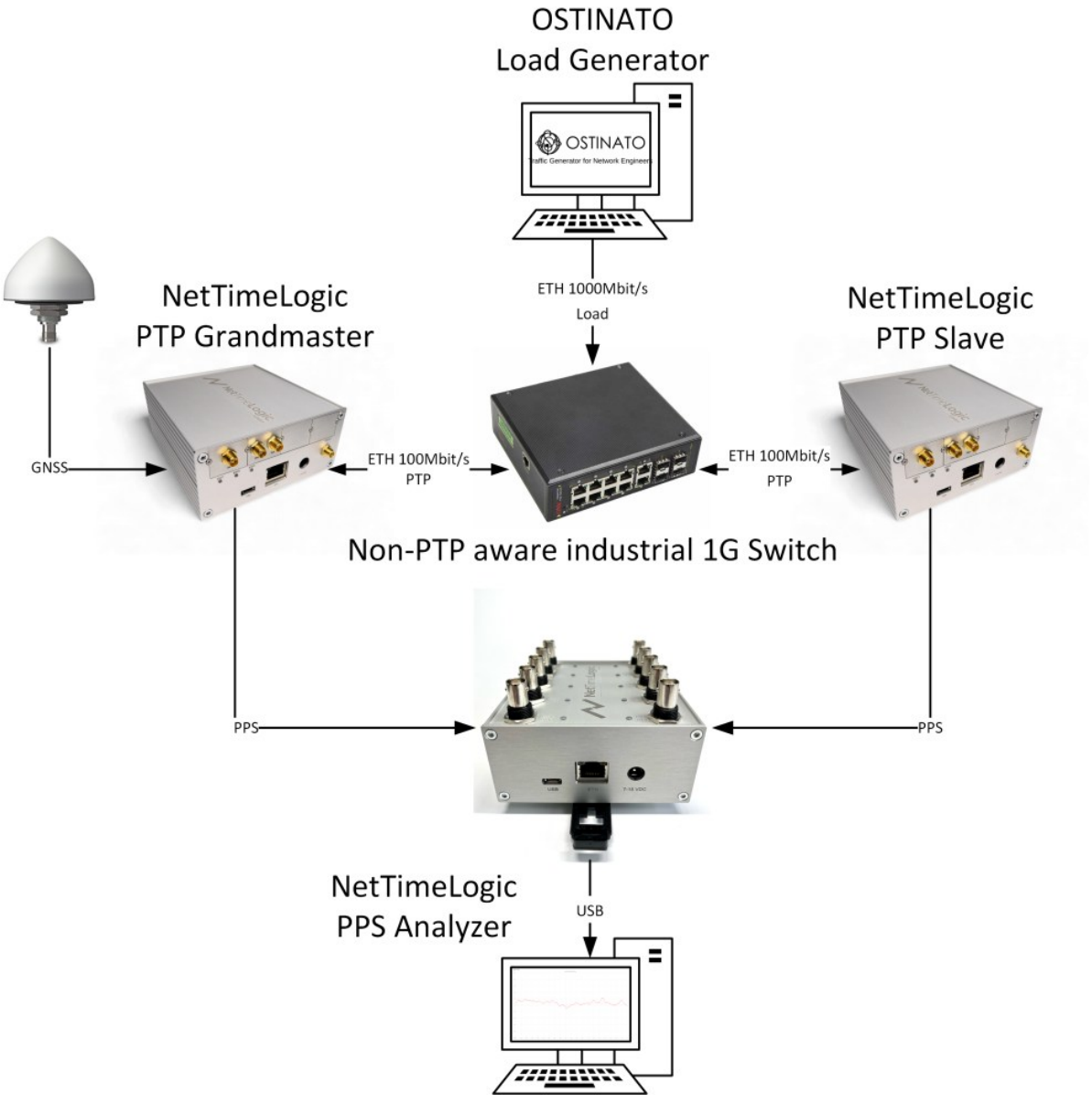
Lucky-Packet Filter:
This filter basically looks for so called Lucky-Packets. Lucky-Packet are frames which, when traveling through the network, experience the least delay (e.g. empty queues on switches). Only Lucky-Packets shall be used for further processing, all others shall be ignored. Since this is a rather rare situation which requires multiple attempts to statistically have at least one Lucky-Packet within a given window (which doesn't mean that we always have a Lucky-Packet). Which brings us to the most important configuration parameter of this filter: the window size. The window size basically tells how many samples shall be searched for the one with the least delay. It also defines the interval of how often corrections can be made. So, it is a trade-off between reaction time and the chance for a Lucky-Packet.

Outlier Filter:
This filter checks for Outliers. If a value differs too much from the last values, it shall be ignored. However, an Outlier Filter must also be able to adapt to sudden changes so it not constantly ignores all new values. There are many different approaches for Outlier Filters but the basic concept is always the same: compare a new vale to a certain threshold and if exceeded ignore the value. The approaches basically differ on how this threshold is calculated and dynamically adapted.

Implementation and Measurements:

These Filters fit nicely into our concept of our FPGA IP cores, so we added them to our IP cores. We added the Lucky Packet filter to our PTP Ordinary Clock IP core (and CSPTP Client IP core) individually for Sync and DelayReq messages and the Outlier Filter to our Adjustable Counter Clock IP core individually for drift and offset corrections.
As with all our IP cores we made pure FPGA implementations of these filters, optimizing them for resource usage and performance (Statistics can use quite a lot of BRAM in an FPGA) and thoroughly tested them with our Simulation framework.

And of course, we wanted to verify its functionality and see the performance gains not only in simulation but also in a real life scenario, so we created the following simple but powerful test setup:



Test-Setup

Devices and Network Setup:

- 10/100/1000Mbit/s non-PTP aware Switch
- NetTimeLogic GNSS synced PTP Grandmaster Clock with 100Mbit/s link
- NetTimeLogic PTP Ordinary Clock with 100Mbit/s link
- NetTimeLogic PPS Analyzer
- Ostinato Traffic Generator

Load Generator via Ostinato from PC:

- 100 packets per burst
- 1400 bursts per second
- random frame sizes between 64 and 1518 bytes
- broadcast traffic (so the congestion happens in both directions)
- ~91Mbit/s injected on a 1000Mbit/s link (To make sure there will be congestion in the switch for the 100Mbit connection between the PTP GM and PTP Slave)

This load pattern will by purpose create quite randomized congestion on the switch which will lead to high Packet Delay Variation (PDV).

PTP Configuration:

- Default Profile, Layer2
- E2E Delay Mechanism
- 128 Sync/DelayReq per Second

Filter & Servo Configuration:

- PI 1/16 Offset, PI 1/48 Drift (to smooth out things at least a bit but without integral part)
- In Sync Threshold 500ns
- Drift Threshold 50ns/s
- Offset Threshold 50ns
- Lucky Packet Window 128

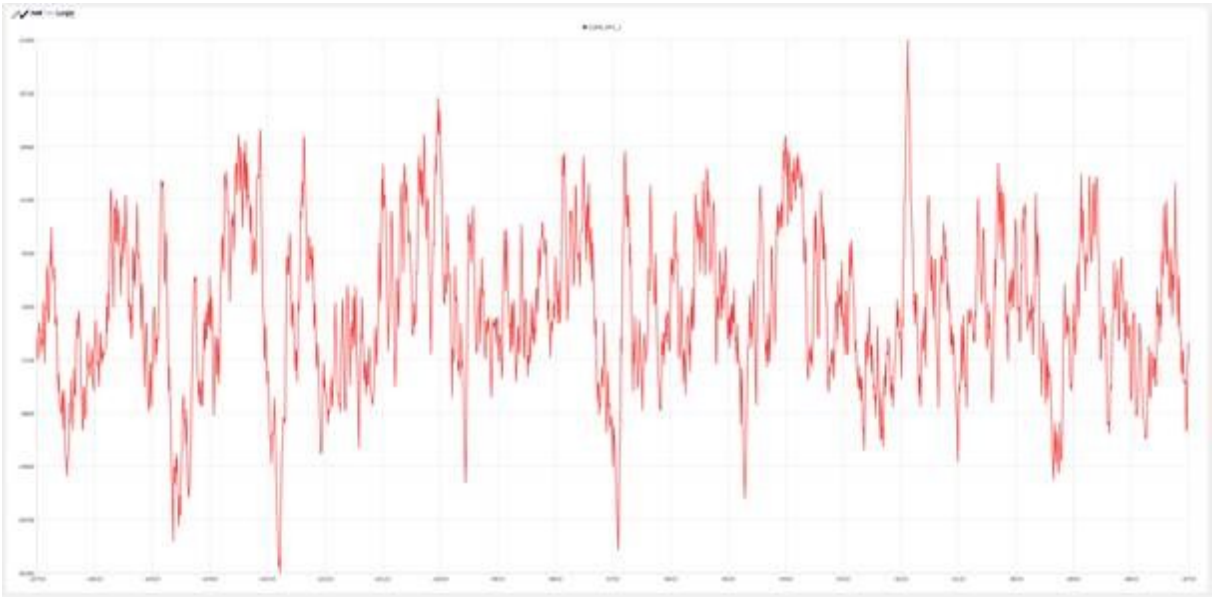
The following 3 test scenarios were created:

1. No Filter
2. Lucky-Packet Filter
3. Lucky-Packet Filter combined with Outlier Filter

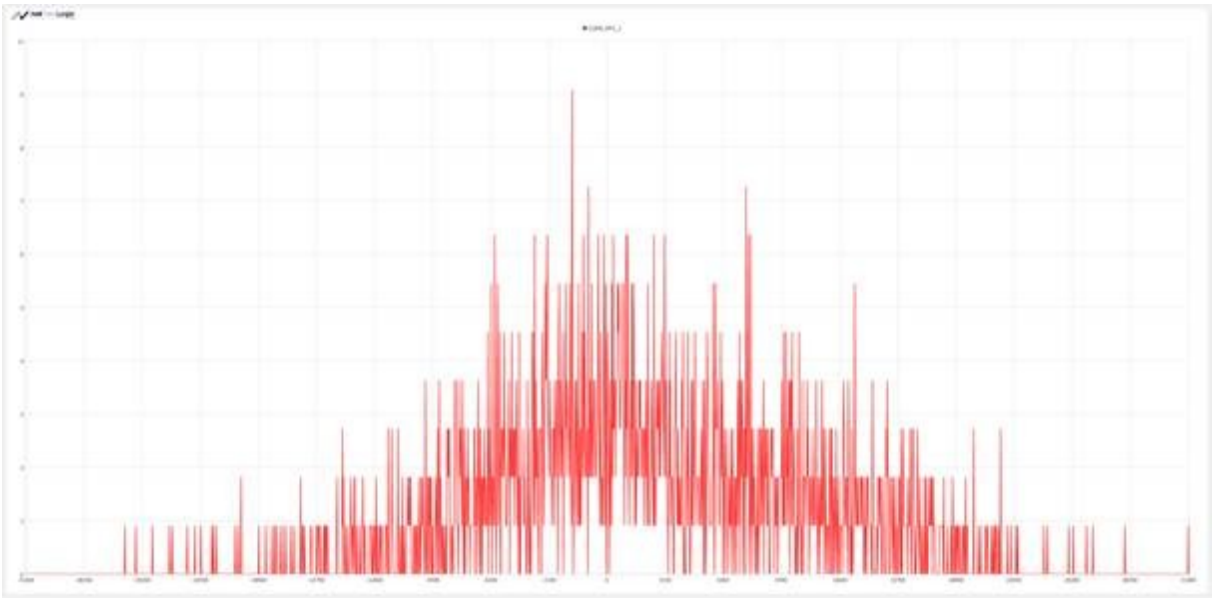
For comparison of the accuracy, the Pulse Per Second (PPS) of the two PTP devices is used, since the calculated error on the Slave side is just a calculated error and does not represent the actual error (as some other companies still try to tell you) and might have a face offset which is not represented in the calculation etc. PPS don't lie 😀 This is also the reason that on every PTP interoperability plugfest PPS is used for accuracy comparison.
NetTimeLogic's PPS Analyzer is used to easily measure the deviation of the Slave's PPS from the Master's PPS. It allows to monitor the offset over a very long time (100k seconds and ~infinite if just logging) and shows its behavior over time (not

like an oscilloscope where you don't know when each deviation was). It also directly shows the distribution of the deviation in a histogram which comes in handy if you want to see if you have an offset and how narrow or wide spread your deviations are.

Scenario 1: No Filter:



No Filter, 2000 samples, +/- 30us scale

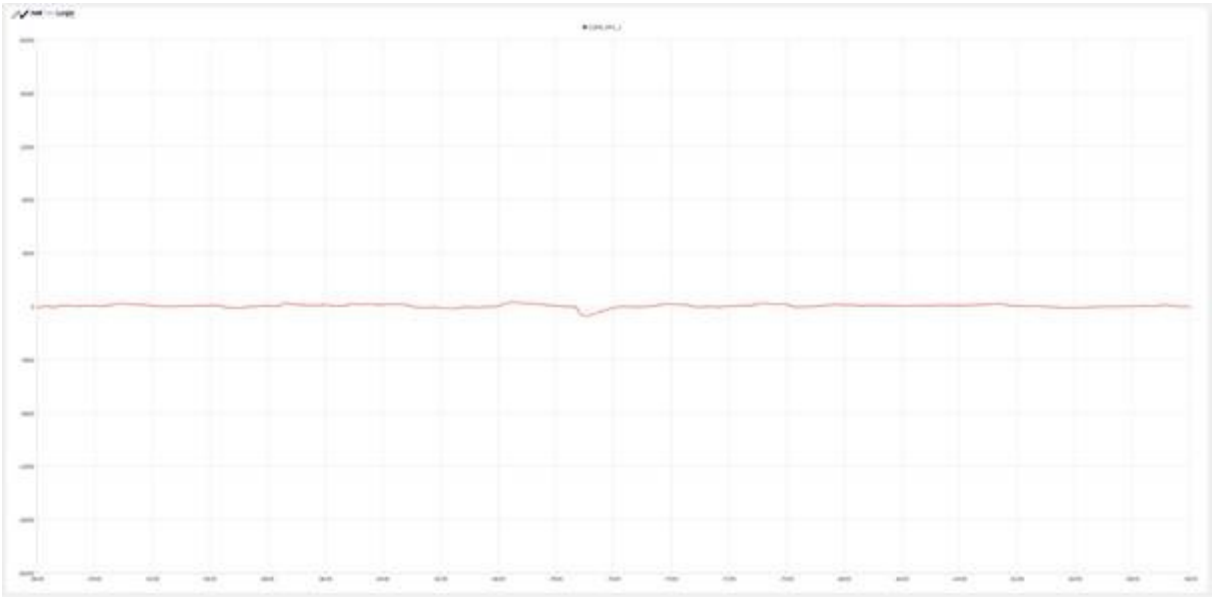


No Filter, 2000 samples histogram +/- 30us scale

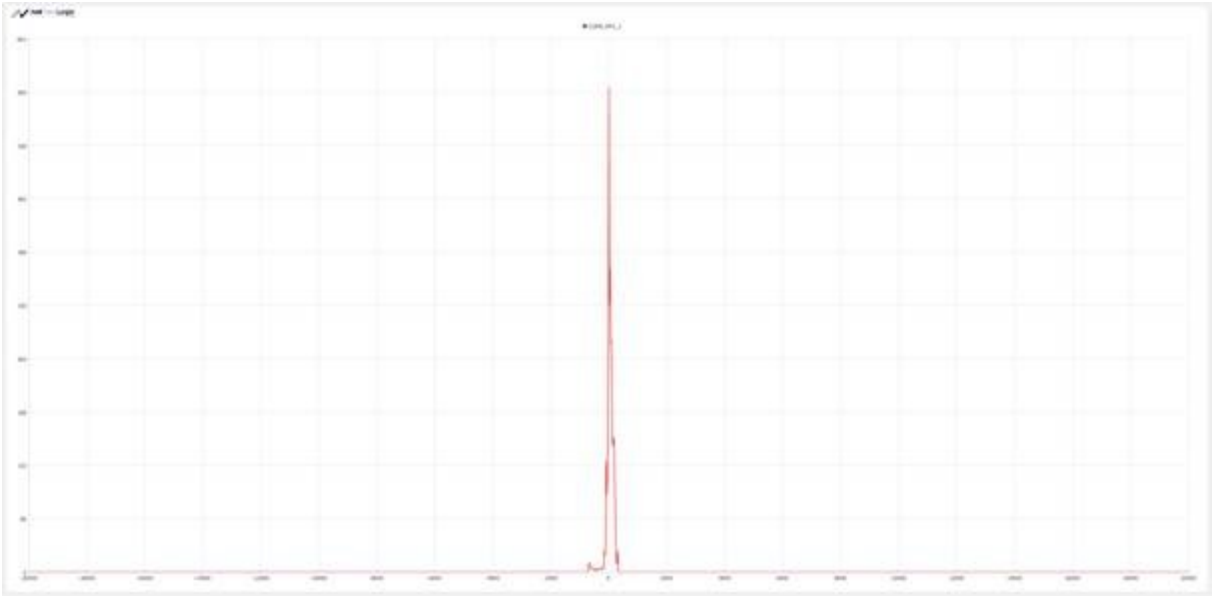
Observed Min/Max deviation on PPS: > +/-20us

Since there is still some filtering by the PI Servo Loop (which is the same for all Scenarios) the Jitter experienced "only" resulted in deviations of ~+/-20us. The packets actually experienced a PDV of up to hundreds of microseconds.

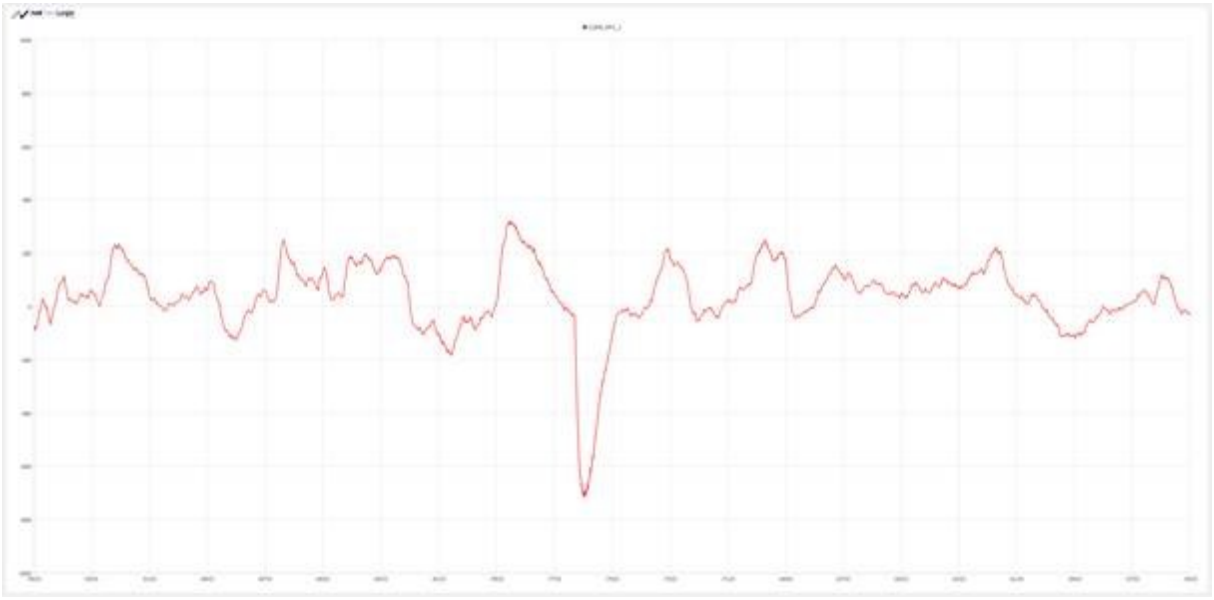
Scenario 2: Lucky Packet Filter:



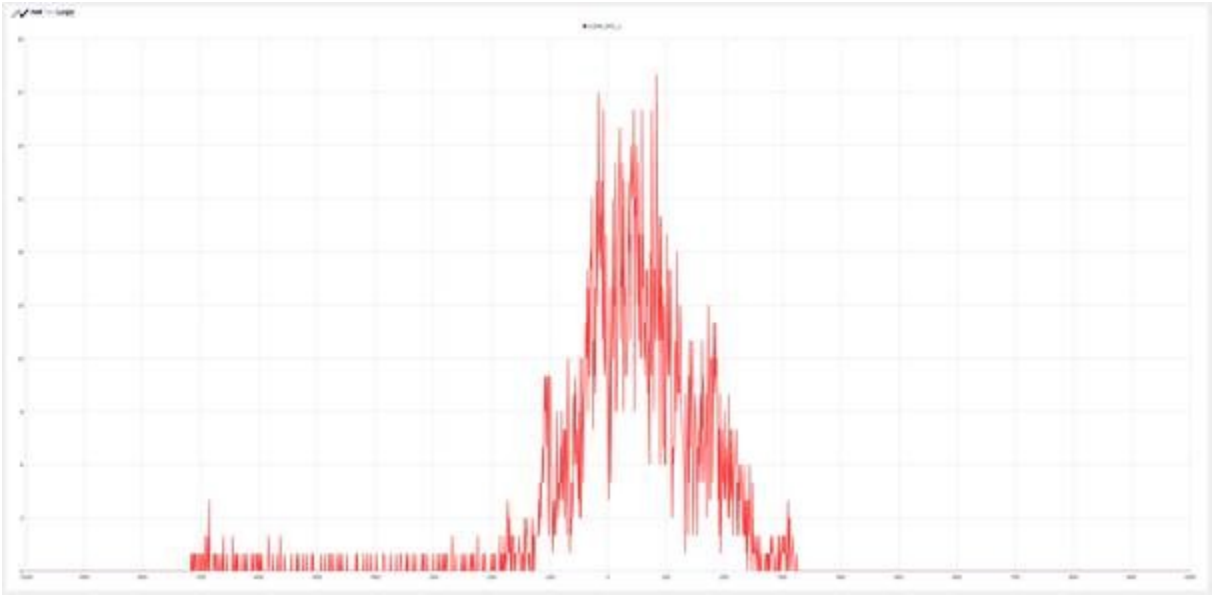
Lucky-Packet Filter, 4000 samples, +/- 20us scale



Lucky-Packet Filter, 4000 samples histogram +/- 20us scale



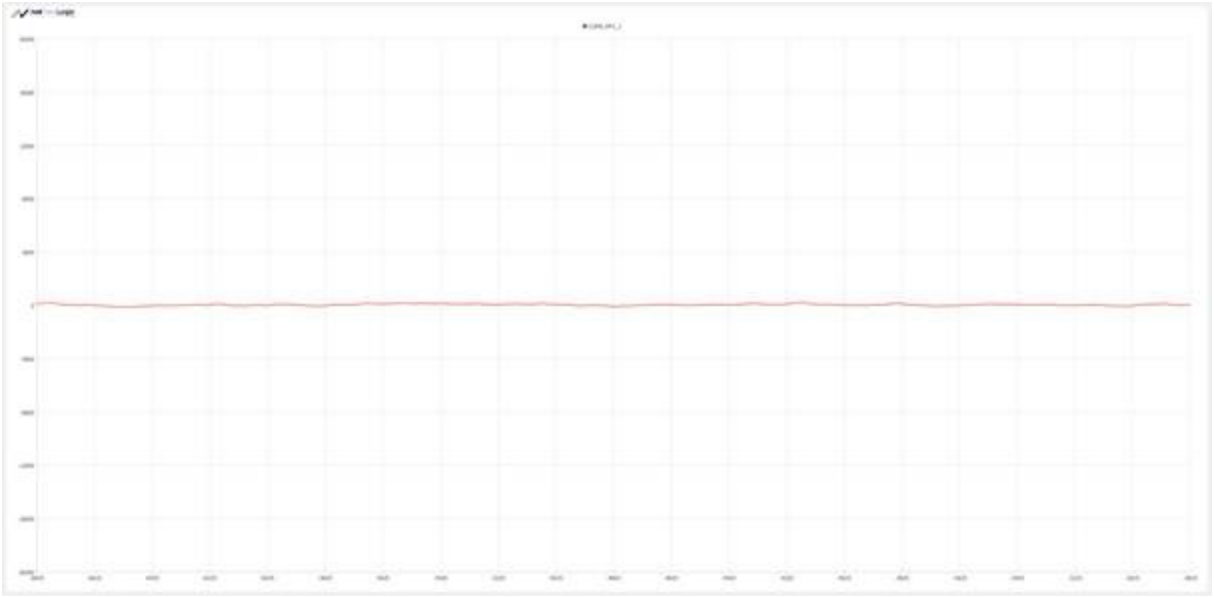
Lucky-Packet Filter, 4000 samples, +/- 1us scale



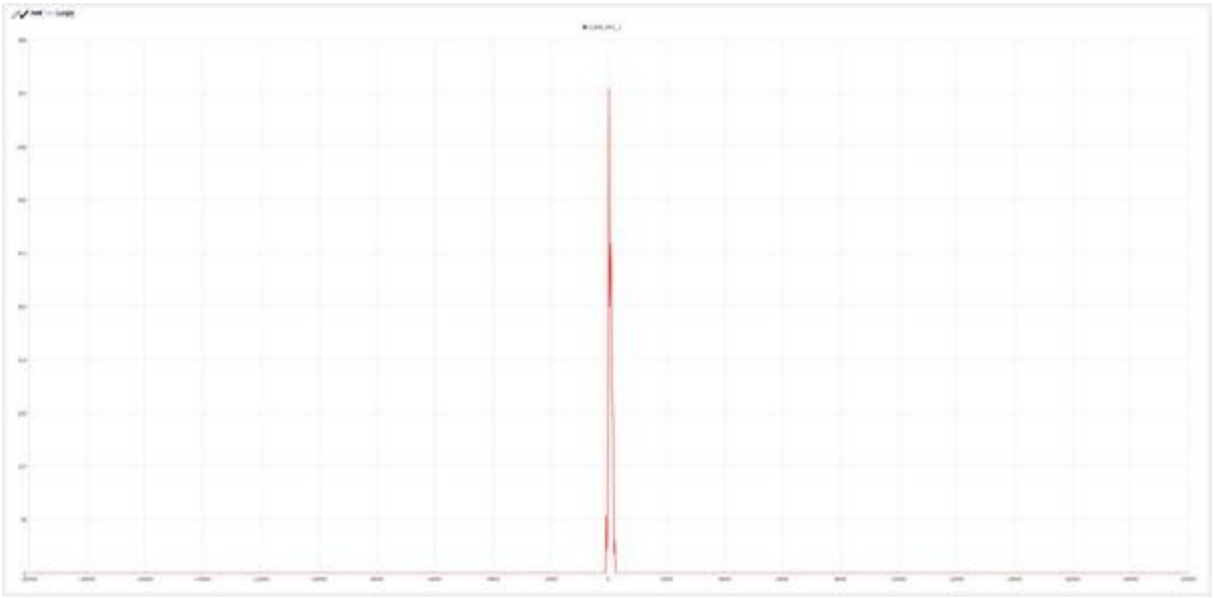
Lucky-Packet Filter, 4000 samples histogram +/- 1us scale

Observed Min/Max deviation on PPS: < +/-1us

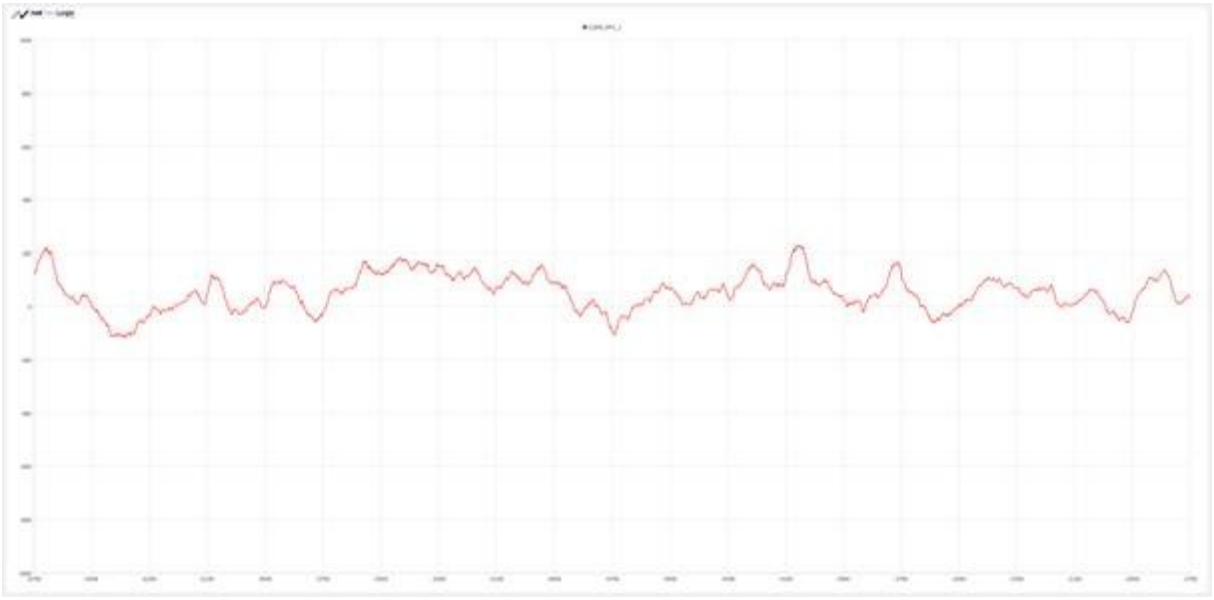
Scenario 3: Lucky Packet Filter & Outlier Filter:



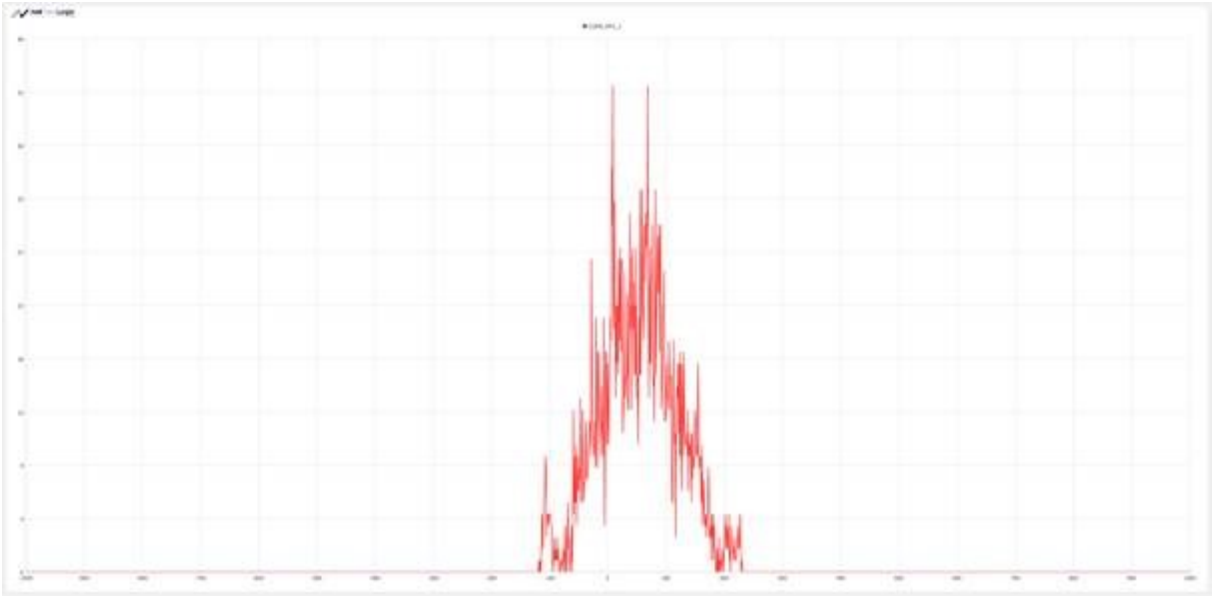
Lucky-Packet & Outlier Filter, 4000 samples, +/- 20us scale



Lucky-Packet & Outlier Filter, 4000 samples histogram +/- 20us scale



Lucky-Packet & Outlier Filter, 4000 samples, +/- 1us scale



Lucky-Packet & Outlier Filter, 4000 samples histogram +/- 1us scale

Observed Min/Max deviation on PPS: +/-200ns

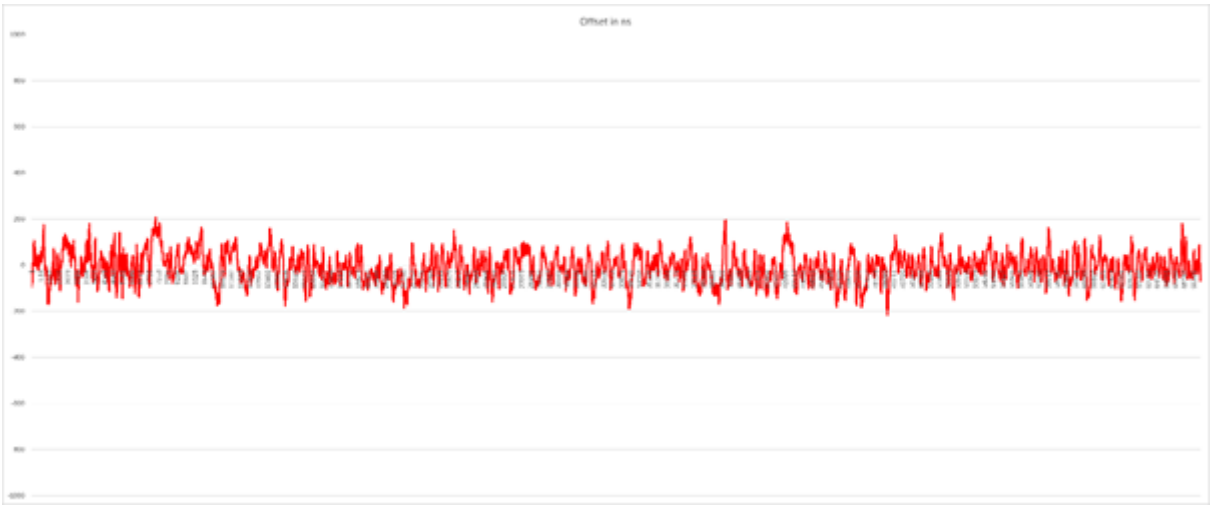
Locking at the results a pure Lucky-Packet filter has improved the accuracy by a factor of more than 20 (from Scenario 1: >+/- 20us to Scenario 2: <+/-1us), which is a great improvement in our point of view. However it also shows the weakness of a pure Lucky-Packet filter which is the fact, that during some windows no Lucky-Packet situation occurred so even the fastest packet experienced probably still some congestion which lead to outlier adjustments. To handle exactly these

situations the Outlier Filter was enabled, which again improved the accuracy by a factor of around 5 (from Scenario 2: >+/- 1us to Scenario 3: +/-200ns) which is again a significant improvement and will satisfy already many applications which require an accuracy better 1us.

So the overall improvement on the accuracy from the No-Filter (Scenario 1) to the combined Lucky-Packet & Outlier Filter (Scenario 3) Scenario resulted in an astonishing factor of 100!

And there is still quite some room for algorithmic improvement on the two filters which could result in an even higher accuracy.

Since in the graphs above we only measured for 4000 samples (4000s = a bit more than 1h) we decided to run a measurement for roughly a day to make sure we don't have any outliers that were taken into account.



Lucky-Packet & Outlier Filter, ~1 day measurement, +/- 1us scale

We are very happy with the result. The observed Min/Max deviation on the PPS is still in the +/-200ns range!

Now back to the title of the article! What do these filters have to do with resilience? Combined they will find the best measurements from a number of measurements and will ignore outliers. Even if you have a PTP aware network there could be some sporadic wrong measurements which these filters basically get rid off, so the Slave would not just follow such a potentially wrong measurement, which increases the resilience of a PTP Slave by quite a lot.

The Lucky-Packet filter in this case can be seen as one type of Outlier filter (mostly similar "small" delay, sporadic outlier "large" delay). And even if an actual outlier passed the Lucky-Packet filter (e.g. calculations of the outlier resulted in the smallest delay) the Outlier filter will detect and ignore it.

Conclusion:

For non-PTP aware Networks the combination of Lucky-Packet and Outlier Filter will significantly improve the accuracy of synchronization (e.g. factor 100).

For PTP aware Networks the combination of Lucky-Packet and Outlier Filter will significantly improve the resilience against erroneous measurements and outliers.

The Outlier Filter is not only useful for PTP but for basically any synchronization mechanism to increase the resilience against any outliers.

There is always room for improvements on the algorithmic of these filters and also combining them with an additional Max-Rate-Change limiter can improve the accuracy even more, but this is something for another article.