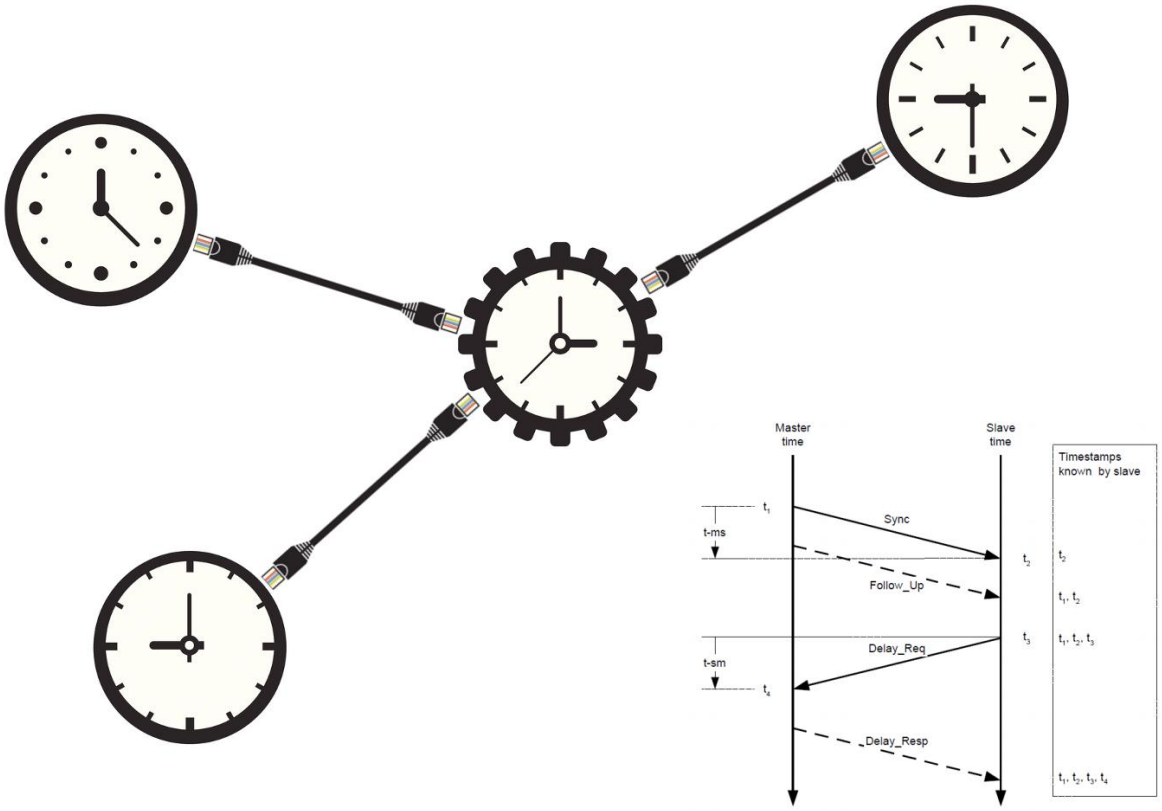
TSEP

Technical
Software
Engineering
Plazotta

IEEE 1588 PTP



What time is it?

Problems and solutions for the implementation of IEEE 1588

The IEEE 1588 Standard

The time synchronization over 1588 has been codified since 2008 as the IEEE standard and is already used in various areas. Previously, the use of this standard was always associated with exotic hardware, that is, implementations of network adapters in various FPGAs or embedded controllers. With the introduction of the Intel network chip families Intel I21x and Intel I35x, this standard is now available for the consumer market. Thus, laying the foundations for new projects based on consumer hardware

There are a number of commercially available IEEE 1588 implementations available for the software implementation. The American LXI consortium even provides a virtually free implementation of the standard (only membership fees are charged). The German company TSEP has developed this IEEE 1588 implementation in cooperation with the LXI consortium. TSEP also sells a paid version specifically for companies that are not interested in becoming a LXI Member.

The fundamental question that arises with every IEEE 1588 project is the accuracy with which the time synchronization must take place. The achievable accuracy usually depends on the hardware used, the topology and the control algorithm used. Modern IEEE 1588 implementations have the ability to define various control algorithms and easily exchange them. The company TSEP has also defined the control algorithm as an independent module with defined interfaces. Thus, the user can simply define their own algorithm and bring this into the system and test it.

For example, when using IEEE 1588 to synchronize wireless LAN speakers, human hearing is the measure of accuracy. The human ear can recognize runtime differences from 10 µs. Thus, the achieved accuracy for the synchronization of the WLAN loudspeakers must be below 10 µs.

But if metrological tasks are considered, other accuracies are required. Within metrology, measurements are usually triggered by triggers. As a rule, these triggers are signal changes (rising or falling edges, exceeding level values, etc.). These signals are transmitted via cable from the source to the meter. Thus, the transit times within the trigger cable is the authoritative target for accuracy. Assuming cable lengths of about 5 meters, which is rather generous, one can expect a running time of 25 ns (running time of 5 ns per meter). Thus, the accuracy of metrological problems would be on this scale. However, in the field of measurement technology, with the introduction of 5G technologies in mobile communications, this magnitude has shifted significantly downwards. For these technologies, accuracies in the subnano range would be desirable.

The IEEE 1588 control algorithm

IEEE 1588 tries to synchronize several freewheeling clocks. Each of these clocks is typically implemented as a counter that increments its counter at a given frequency. Due to the frequency and the meter reading, the current time can now be derived at any time. Since it is not technically possible to have identical frequencies generated by several oscillators, the frequency must be readjusted. Since it is technically much easier to manipulate the counter cycle, this is changed. This adjustment must be made via a control algorithm, as the adjustments are subject to various disturbances. In addition, however, faults that may occur in the transport route must also be taken into account. Since every IEEE 1588 implementation is based on its own hardware and hardware topology, there can not be "the general control algorithm".

In principle, the algorithms can be divided into two groups. The first group is based more on simple algorithms, which usually focus only on the determined time difference between master and slave (also called MeanPathDelay) to determine the error in the frequency of their own clock. This type of algorithm is independent of the hardware topology used and provides quite useful results. The free LinuxPTP implementation and the TSEP implementation each contain such an algorithm by default.

The second group is algorithms that try to identify the errors that are in the system and include them in the calculation of the error of their own frequency. These algorithms only make sense if the hardware used and the expected topology is known. Due to the hardware used, the error models can then be created and used. For this type of control algorithms Kalman filters are particularly suitable, which can be modeled specifically for the corresponding problem.

Each control algorithm contains at least two states. In the first state, the offset between master and slave (MeanPathDelay) is so large that the algorithm can not close this gap within an acceptable control time. In this state, the time received from the master is taken over directly without correction as a separate slave time in the hope that the MeanPathDelay value determined in the next synchronization interval is significantly smaller. This state is maintained until an acceptable MeanPathDelay is reached. This state is the default state after starting the clock or if synchronization is lost due to problems. In the second state, the actual control algorithm is used, which tries to determine the correction values of the own clock and to approximate its own time as exactly as possible to the master time.

Simple IEEE 1588 control algorithms

As already stated, these types of algorithms are designed to determine only the correction value of the own clock on the basis of the determined MeanPathDelay. An own error model is not included here or only sparsely included. This type of algorithm is relatively simple and can be used independently of the hardware.

Based on the standard control algorithm of the TSEP IEEE 1588 implementation, the operation of such an algorithm should be illustrated. In the first step, this simple control algorithm does not attempt to incorporate invalid or incorrect MeanPathDelays into the control. The IEEE 1588 implementation discussed here is based on Intel network chips of the I21x family. These use the Gigabit Ethernet according to IEEE standard 802.3. This type of network system is not deterministic; any participant can access the network at any time. The accesses are organized via packet collisions. This can happen that packets are transmitted much later than is actually assumed. This delay does not appear in the transmitted data packets. In order to protect the control algorithms against these false and thus disturbing data, the control algorithm tries to detect these false data packets and they are excluded in the control algorithm.

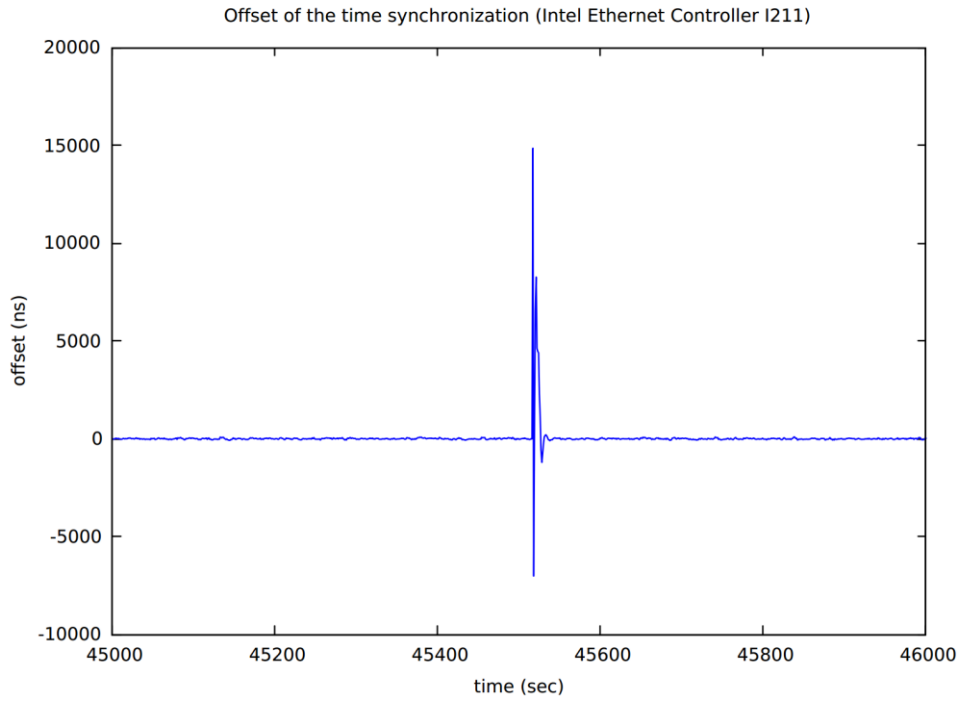


Figure 1: Wrong MeanPathDelay through network delay

The diagram above shows such a wrong package that was subsequently included in the control algorithm and then had to be balanced. These false data can be recognized by the significantly increased MeanPathDelay. To detect these false MeanPathDelays, the standard deviation of MeanPathDelay is calculated:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(xi - \mu)^2}{n}}$$

σ *Standard deviation*
μ *Expected value*
n *Number of measured values*

Figure 2: Calculation of the standard deviation MeanPathDelay

If a new MeanPathDelay clearly exceeds the calculated standard deviation, it will not be used for further processing.

In the next step an attempt is made to calculate the correction of the own clock from the calculated MeanPathDelay. For this purpose, the deviation of the slave from the master, which was determined by the MeanpathDelay algorithm, is transferred to the frequency of the own counter (clock). For this purpose, the error of the own clock per counter step is calculated in the first step:

$$\Delta F\ [sec] = \frac{\Delta tm\ [sec]}{\Delta ts\ [sec]\ *\ (1\ /\ tps)}$$

ΔF: *Error of the clock per tick*
Δtm: *Offset Master/slave = MeanPathDelay*
Δts: *Time since last calculation*
tps: *Number of ticks in one second*

Figure 3: Calculation of internal clock error

The Intel I21x can vary within certain limits the time after which its own counter is incremented (every 8 ns). The error per counter increment is programmed into the hardware by the algorithm (according to the above formula).

In the case of such simple control algorithms, the system starts to oscillate again and again because the control algorithm uses the correction value correspondingly, depending on the detected error. To avoid such upsurge, the TSEP IEEE 1588 control algorithm considers the 1st derivative of the MeanPathDelay.

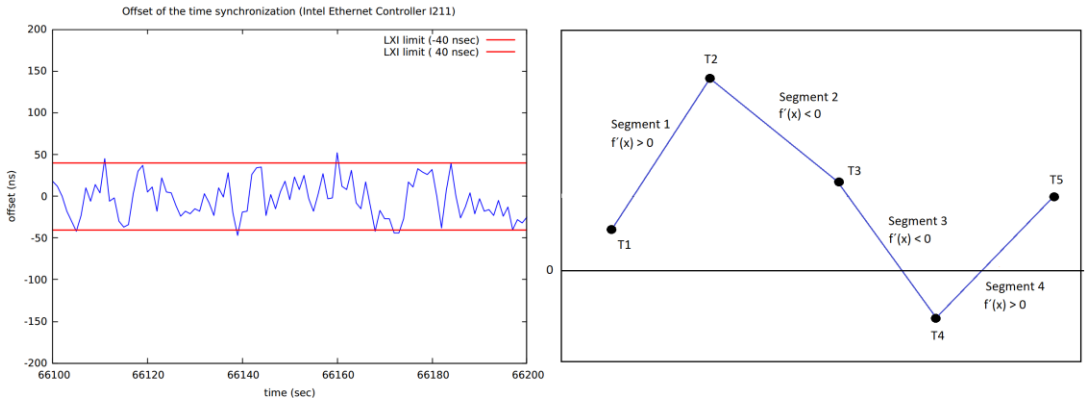


Figure 4: Extended control algorithm

As can be seen in the diagram above, the control algorithm already at the end of segment 2, that is to say at T3, can recognize that the corresponding regulation is taking effect and corrects the adaptation accordingly.

As we have said, these simple control algorithms allow you to build simple and well-functioning systems independent of fault models and hardware topologies. Of course, these are compromises in accuracy. Therefore, these control algorithms are usually included as standard in all IEEE 1588 implementations and allow the user an easy introduction to the IEEE 1588 problem.

Complex IEEE 1588 control algorithms

But if you want to look at error models and hardware topologies, you have to work with different approaches. Kalman filters (or Kalman-Bucy-Stratonovich filters) can be used for such problems. The Kalman filter is named after its discoverers Rudolf E. Kálmán, Richard S. Bucy and Ruslan L. Stratonovich, who independently discovered the process or made significant contributions to it. The Kalman filter is used to reduce errors in real measurements and to provide estimates for non-measurable system quantities. The prerequisite is that the necessary values can be described with a mathematical model. The special feature of the filter presented by Kálmán in 1960 [3] is its special mathematical structure, which enables its use in real-time systems of various technical fields. This includes, for example, the use in electronic control circuits in communication systems. Mathematical estimation theory is also called a Bayesian minimum-variance estimator for linear stochastic systems in state space representation.

The Kalman filter tries to include error models in the estimation for the actual correction value. For this purpose, it is necessary in the first approach to be able to estimate the sources of error within an IEEE 1588 implementation.

Stability of the oscillator within the Intel I21x network chips

One size of error in an IEEE 1588 implementation is the stability of the internal counter used. The current time is derived from an internal counter. The clock at which the counter is incremented is derived from an oscillator or other source. For the Intel network chips I21x and I35x this is derived from the used Ethernet clock, ie 125 MHz.

Figure 5: Measurement setup with Omicron Grandmaster Clock

To get an overview of the stability of this clock, measurements were made in the TSEP laboratory (see picture above). Several computer boards and PC plug-in cards with the corresponding network chips were measured. For this purpose, the internal clock on the network chips was operated with a constant adjustment over several hours. The measured variable was a pps signal (Pulse per Second) programmed on the GPIO of the Intel network chip and measured with a corresponding oscilloscope. The Grandmaster Clock used an Omicron Grandmaster Clock.

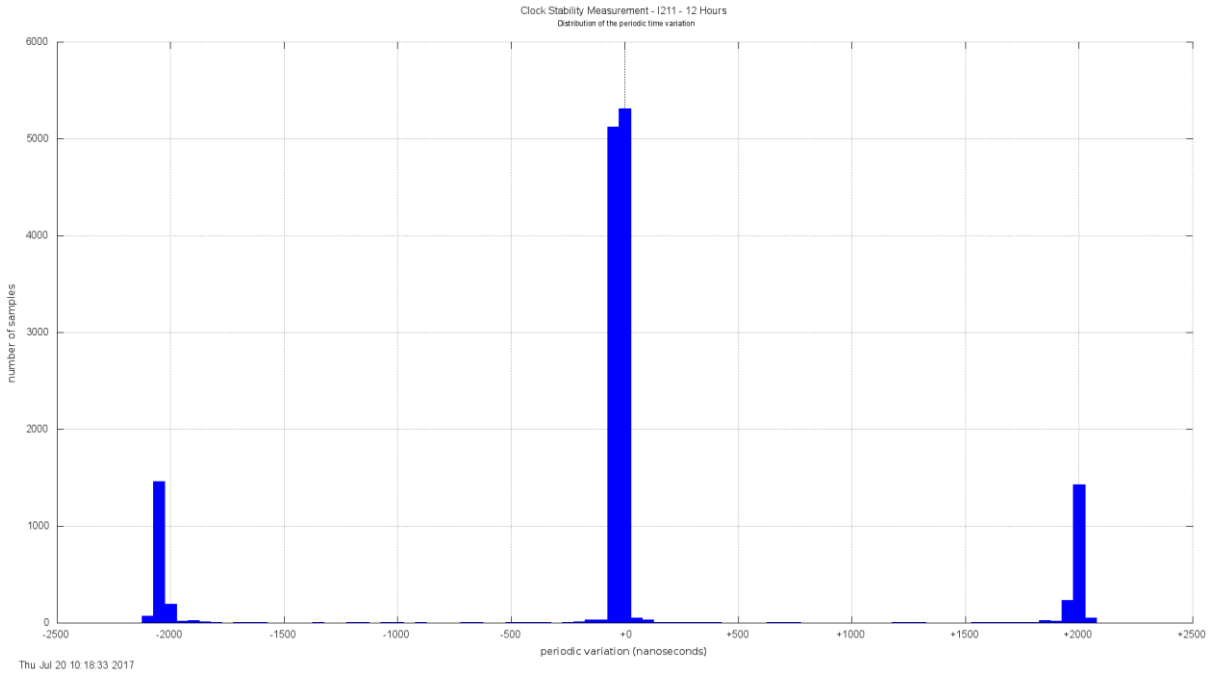


Figure 6: Error distribution room temperature

The diagram above shows the difference in the period of the pps signal. In this diagram it can be seen that the error of the clock is grouped around the middle position. The measurement shows that the error is about +/- 2000 ns. Based on the clock's 125 MHz clock, there is one error per clock / increment of clock from:

2000 ns / 125 MHz = 16 x 10-15 sec

This error size could also be measured with other Intel 21x network cards or embedded computers with Intel I21x.

Another issue is the temperature stability of the internal clock of the Intel I21x chips. For this purpose, the ambient temperature of the network chip was varied. Below are two diagrams showing the error in the corner temperatures for the chip.

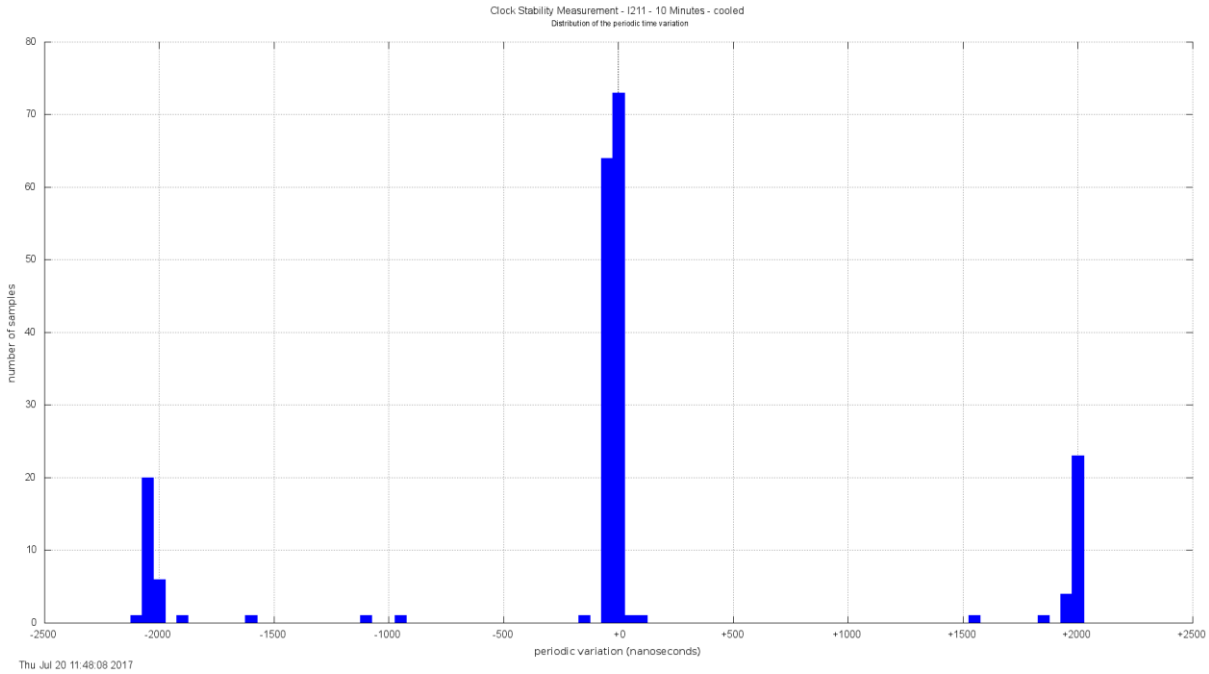


Figure 7: Error distribution cooled network chip

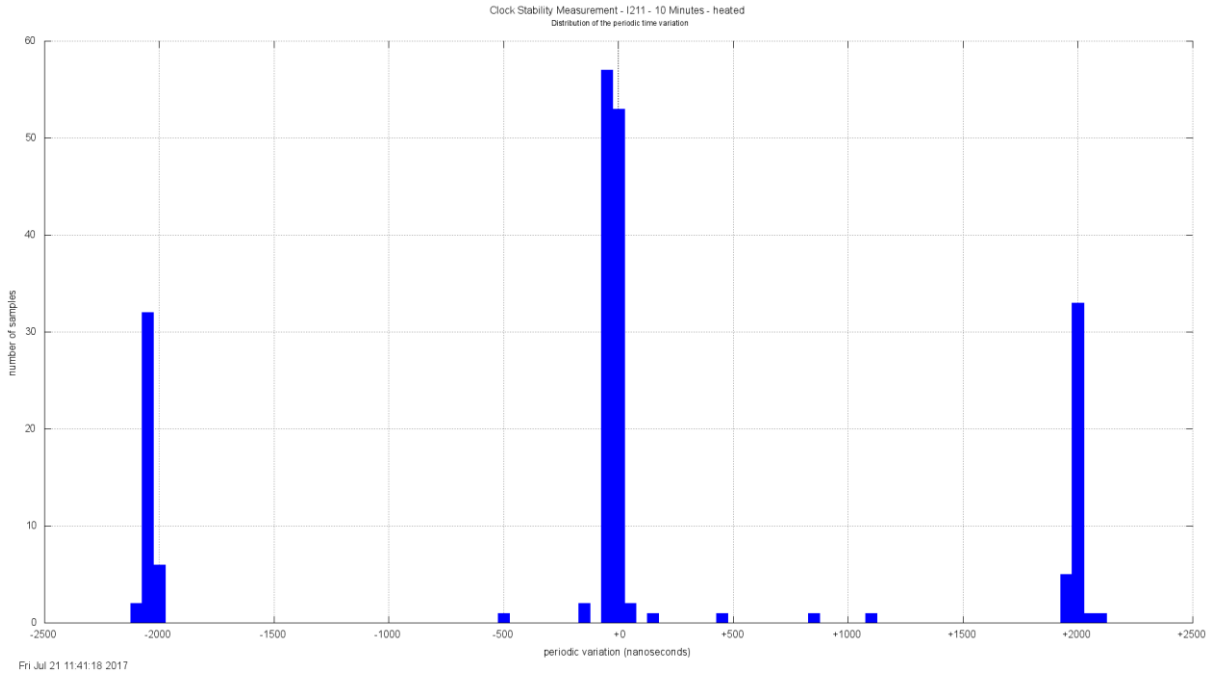


Figure 8: Error distribution of heated network chip

From the diagrams you can see that the error distribution is clearly different. At an elevated temperature, the error noticeably occurs more frequently.

Multiplexing with Gigabit Ethernet data transmission

In order to have a better understanding of these issues, some basics about the IEEE 802.3 need to be mentioned.
The result is the 1 GB Ethernet from the previously established 100 MB Ethernet standard. The Ethernet CAT 5 cables used were intended for the transmission of signals at 125 MHz. These cables had four pairs (two wires each). However, only over two pairs of data were transmitted. Gigabit Ethernet will now transmit two bits over all four pairs, so:

125 MHz x 2 bits x 4 communication channels = 1000 Mbps = 1Gbps

So there are always two bits transmitted on the four twisted pairs. The sender has to divide each byte into four parts and the receiver has to bring them together again. The processing clock here is as said 125 MHz.

The four wire pairs are used simultaneously for both directions. The frequency used within the Gigabit Ethernet is 125 MHz according to GMII (Gigabit Media Independent Interface). The frequencies at the transmitter and receiver are not necessarily coupled. Depending on the data transfer, the frequency for the data is derived from the own clock or from the network. Due to this procedure and the probable assumption that the four lines are not of identical length, a different runtime for the individual partial data (4 x 2 bits) must be expected. Only after all four data parts of a byte are available, they can be put together again, which can lead to delays. Depending on the scenario, up to 20 ns of delay can occur here. The whole problem is very well described in the document "Improving IEEE 1588 synchronization accuracy in 1000BASE-T systems" [1]. Unfortunately, because the reported problems are in the ns range, they significantly affect the accuracy of time synchronization over IEEE 1588. With copper-based Gigabit Ethernet implementations, it can be very expensive or almost impossible to reach the subnano range. This fact was certainly one of the reasons why the White Rapid System [2] uses networks with fiber optic technology.

Consideration of runtime errors in "Long Linear Paths"

Actually, you can only count on small systems or in the development of systems with a Grandmaster Clock, a Transparent Clock (Switch) and a Slave Clock. In reality, however, one must assume connections in which the Grandmaster Clock must run over several Transparent Clocks or even non-PTP compliant switches. All these factors add up over the transport chain from the master to the slave. In order to improve your own control algorithm, you can try to detect the errors that occur in the individual switching nodes and integrate them into your own control algorithm. For this purpose, a model can be set up with the help of the Kalman filter, which takes into account the individual errors. Document [4] "Accurate Time Synchronization in PTP-based Industrial Networks with Long Linear Path" by Daniele Fontanelli and David Macii shows a modeling using a Kalman filter for this problem. The document also shows that this approach can improve the accuracy of IEEE 1588 systems in this scenario.

The Kalman filter

As early as 1960, Rudolf E. Kalman developed a special procedure for time-discrete linear systems to estimate the states of a system (including their parameters) from noisy and partially redundant measurements. This method was known as Kalman filter and first published in [3]. Since then, many different variants of the Kalman filter have been published. The following description can also be read in detail in [5].

In order for the Kalman filter to be used properly, it is necessary that the basic conditions of the measuring system are known. Each classic Kalman filter usually consists of a state space description and the real measurement system with its own system and measurement noise. From this system, the prediction and the correction can be calculated using the Kalman filter. Basically, a Kalman filter estimates the output quantity ŷ (k) and compares this with the measured output variable y (k) of the real measuring system. The difference Δy (k) of the two values is weighted with the Kalman gain K (k) and used to correct the estimated state vector ẍ (k). The structure can be described as follows:

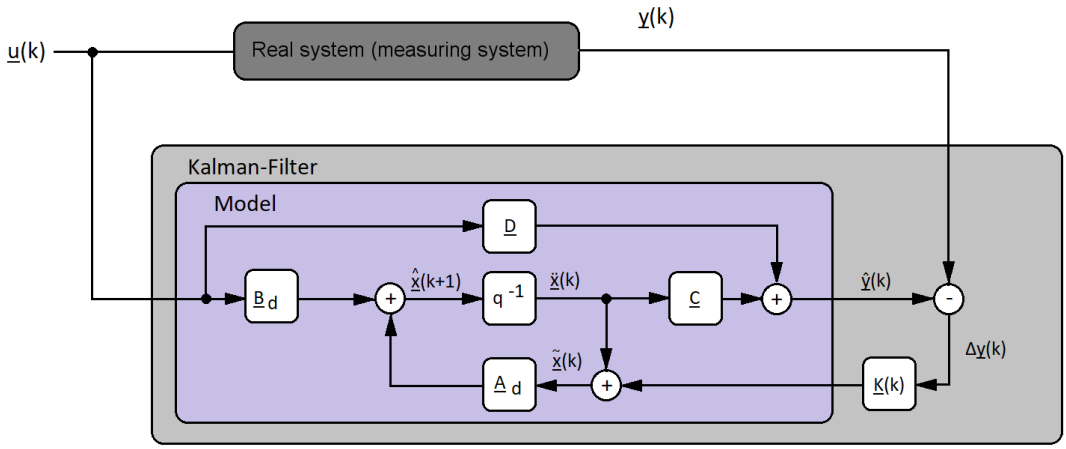


Figure 9: Structure of a Kalman filter

From this structure, the five Kalman filter basic equations can be derived:

Prediction:

$$\ddot{\underline{x}}(k+1) = \underline{A}_d * \dot{\underline{x}}(k) + \underline{B}_d * u(k)$$
$$\underline{\acute{P}}(k+1) = \underline{A}_d * \dot{P}(k) * \underline{A}_d^T + \underline{G}_d * \underline{Q}(k) * \underline{G}_d^T \quad \text{with } Q(k) = \text{variance } (\underline{z}(k))$$

Correction:

$$\underline{K}(k) = \underline{\acute{P}}(k) * \underline{C}^T * (\underline{C} * \underline{\acute{P}}(k) * \underline{C}^T + R(k))^{-1} \text{ with } \underline{R}(k) = \text{variance } (\underline{v}(k))$$
$$\dot{\underline{x}}(k) = \ddot{\underline{x}}(k) + \underline{K}(k) * (\underline{y}(k) - \underline{C} * \ddot{\underline{x}}(k) - \underline{D} * \underline{u}(k))$$
$$\dot{P}(k) = (\underline{I} - \underline{K}(k) * \underline{C}) * \underline{\acute{P}}(k)$$

The derivation is omitted due to the complexity. However, it can be read in [3] or [5].

When designing a Kalman filter, it is necessary to describe the physical system continuously using differential equations. It sets the output vector y (t). It must be ensured that this output vector contains all noisy quantities. The non-noise quantities are described separately in the output vector u (t). When determining the state variable x (t), there may be several approaches. As a rule, these options should be evaluated and the approach that best describes the problem should then be used.

If all equations and parameters are selected, the system is in the following form:

$$\dot{\underline{x}}(t) = \underline{A} * \underline{x}(t) + \underline{B} * \underline{u}(t) + \underline{G} * \underline{z}(t)$$
$$\underline{y}(t) = C * \underline{x}(t) + \underline{D} * \underline{u}(t)$$

This time-continuous description must then be converted into a discrete-time description. The following formulas can be used for this:

$$\underline{A}_d = e^{A*T_S}$$
$$B_d = \int_0^{Ts} e^{A*v}\, Bdv \quad \text{oder } B_d = \underline{A}_d * \underline{B}$$

T_S is the sampling rate of the system. The matrices C and D remain identical in the discrete-time system.

For the determination of the matrix Gd, several approaches can be used, such as the sampling by the Dirac pulse. The method used must be determined individually.

After the description of the system is available in the state space, a check of the observability can take place. As a criterion for this, according to [3] or [5], a linear time-invariant system of order n can be observed if the observability matrix SB, or S * B, has the rank n. However, it is also possible to use criteria from Gilbert or Hautus. If the system is unobservable, it can be split into observable and unobservable.

Finally, the system noise Q (k) and the measurement noise R (k) have to be described.

$$\mathbf{Q(k) = Variance(\underline{z}(k))}$$
$$\mathbf{\underline{R}(k) = Variance(\underline{v}(k))}$$

In order to use Kalman filters optimally, these two quantities must be determined as accurately as possible. In simple systems one can speak of approximately constant quantities, which does not apply however in the environment of IEEE 1588. It can be assumed that these variables change during runtime and thus adaptively adapt.

There is a variant of the Kalman filter that works with an adaptive determination of the two covariance matrices: the ROSE (Rapid Ongoing Stochastic Covariance Estimation Filter) filter. The principle is based on cyclically re-determining the covariance of the measurement noise R (k) by observing the metrologically detectable variable y (k) using two embedded simple Kalman filters (with constant Kalman amplification). Analogously, the covariance of the system noise Q (k) is calculated using the value Δy (k) and the measured quantity y (k), the covariance of the estimation error Ṕ (k + 1), the covariance measurement noise R (k) and a simple Kalman Filter determined.

Considering the individual steps that lead to a complete description of the state space and the covariances of the system and measurement noise, one can say that the creation of a Kalman filter is anything but trivial. However, the boundary conditions of the measurement system can be optimally embedded in the Kalman filter, resulting in the best possible correction of the internal IEEE 1588 clock.

Conclusion

The successful use of an IEEE 1588 implementation is not only based on the use of an existing IEEE 1588 stack or special hardware. The problem-oriented approach is the key to success here. The possibilities of IEEE 1588 are wide-ranging. However, without knowing the requirements for the necessary accuracy for an IEEE 1588 implementation and the available hardware topology, it is difficult to provide a framework that works within the framework of the requirements. The analysis in advance is mandatory and requires appropriate know-how.

The choice of the hardware used is of essential importance, since the individual components affect the accuracy differently. Especially for high accuracies in the subnano range, the use of fiber-based systems is imperative. The White Rabit project did a good groundwork here and created the appropriate hardware and boundary conditions. Also, the Intel network chips can work with fiber optic based PHYs, corresponding hardware is available in the market as consumer goods.

For highly accurate systems, a perfectly matched control algorithm is essential. The choice of the appropriate algorithm is not easy, requiring a lot of know-how and having to be adjusted to the hardware and its topology. The implementation and simulation of the algorithm, especially for highly accurate systems is a not inconsiderable part of an IEEE 1588 implementation. But it is also the key to the success of such an implementation. For the implementation of such efficient control algorithms, Kalman filters are very well suited. However, the description of the state space and the covariances of the system noise and measurement noise require considerable effort and time.

The article does not claim to portray all facets of this extremely complex matter, but is intended to give a brief overview of the problems and solutions that exist. Each available IEEE 1588 stack is just the toolbox for an implementation. The proper use and implementation of the control algorithm is the actual task.

And yes, what time is it right now?

Bibliography:

[1] Improving IEEE 1588 synchronization accuracy in 1000BASE-T systems, Rodney Greenstreet and Alejandro Zepeda

[2] White Rabbit Project. White rabbit wiki, 2016.

[3] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems (Transaction of the ASME, Journal of Basic Engineering). 1960, S. 35–45

[4] Accurate Time Synchronization in PTP-based Industrial Networks with Long Linear Path, Daniele Fontanelli and David Macii

[5] R. Marchthaler, S. Dingler: Kalman-Filter, Springer Vieweg

Author:

Peter Plazotta, Dipl. Ing (FH)
CEO of TSEP and for 30 years with the design and development of system software mainly in the areas of T & M, telecommunications and automotive busy. He has been working on IEEE 1588 for more than 10 years and has been involved in several implementations with his company in this area.

Peter.Plazotta@tsep.com