Input for simulating ClockSync according to IEC/ IEEE 60802

Dragan Obradovic & Günter Steindl, Siemens AG

Purpose of the document:

The intention of this document is to specify all necessary inputs for simulating ClockSync systems. These are:

- signaling and different control and estimation methods for calculating the GrandMaster time at slave elements, including their parameters.
- methods for estimating the underlying error distributions due to the presence of different stochastic uncertainties, such as stamping errors, oscillator drifts, etc. The distribution of the latter will be specified in the document too.

The intended users are the simulator developers / users who have basic knowledge about the Clock Synchronization in IEC/IEEE 60802 standard.

Versions:

V1: This version will focus on 1-step Sync messages (no Follow-up). The 2-step Sync message will not make big differences. Whenever some modeling details are not clear, this will be stated in the document. Completion of the document will, therefore, depend on the inputs from other participants in the standardization process.

Choice of the Simulation Type

The Clock Synchronization is a dynamic process where many actions are triggered by sending and receiving different type of messages (containing different time information). Although sending of some messages is repetitive with known cycles (e.g., the timing of the GrandMaster (GM) element sending Sync-Messages), the sending and receiving times of other messages are not known in advance (e.g., forwarded Sync Messages from one slave to another). Hence, there is no natural sampling-time which can be used in a computer simulation of this system.

Hence, we choose to have a discrete-event simulation of ClockSync. Every event (e.g., receiving a Sync Message at a slave element) triggers other events, like updating this time with the estimated p-delay, presenting this updated time to a controller, and defining the residence time (and therefore the time when this message is forwarded to the next element). Which events are triggered is described with simple finite-state machines.

In the simulation we have models of the latter events (such as the residence time duration, including its distribution) as well as the functions describing frequency changes over time for all involved oscillators (this is a necessary input for any simulation). Hence, we are able to generate an event list which is updated every time a new event is generated. All the events in the list have an associated true time (the simulator generic time) but also the times in different clocks such as the free running clock (e.g., LocalClock) and in the controlled time (e.g., the ClockSlave time).

The event list is typically initiated with the events which are already planned at the system start, such as the GM's sending of SyncMessages and the sending of the p-delay request messages for slave elements. Hence the initial list also reflects the initialization strategy of the system.

Summary: The ClockSync process will be simulated with a discrete-event simulator.

Topology of Interest

The topology of interest is a line topology with "N" elements, where the first element is a GrandMaster (GM). The rest of the elements are slave elements, where some of them might just forward the Sync Messages and not have ClockSlaves, and other slaves which have ClockSlaves and are potentially attached to exterior applications (i.e., are attached to ClockTargets).

Herein we refer to the document (Steindl, 60802-steindl-Sync-Error-Sources-v02_July2022.pptx, 2022), which gives an overview of Clock Synchronization systems and analyzes possible error sources.

Figure 1: Time transmission Path from (Steindl, 60802-steindl-Sync-Error-Sources-v02_July2022.pptx, 2022)

The GM element is the first in the line (the leftmost element). It consists of a ClockMaster and a LocalClock, and it is connected to a ClockSource. The mentioned clocks are counters which count tics of associated oscillators. The ClockSource and ClockMaster clocks are controlled clocks, i.e., they are driven by control loops which can modify their counter values by adding or subtracting single ticks after specified time intervals. In other words, the control loops make “SW-modifications” of the oscillator frequencies. The LocalClocks in all elements (the GM and in Slaves) are not controlled, hence they are pure integrators of the oscillators’ frequencies.

The slave elements have a similar structure, consisting of the free running LocalClocks and controlled ClockSlaves. The ClockSlaves provide time information to ClockTargets at applications. The ClockTargets control loops track the time (the counter value) information they received from attached ClockSlaves.

The simulation will focus on the clocks within the GM and Slave elements, which means that the ClockSource and the ClockTargets will not be explicitly simulated. The errors between the ClockSource and ClockMaster, as well as the errors between ClockSlaves and associated ClockTargets will be specified with appropriate “error budgets”.

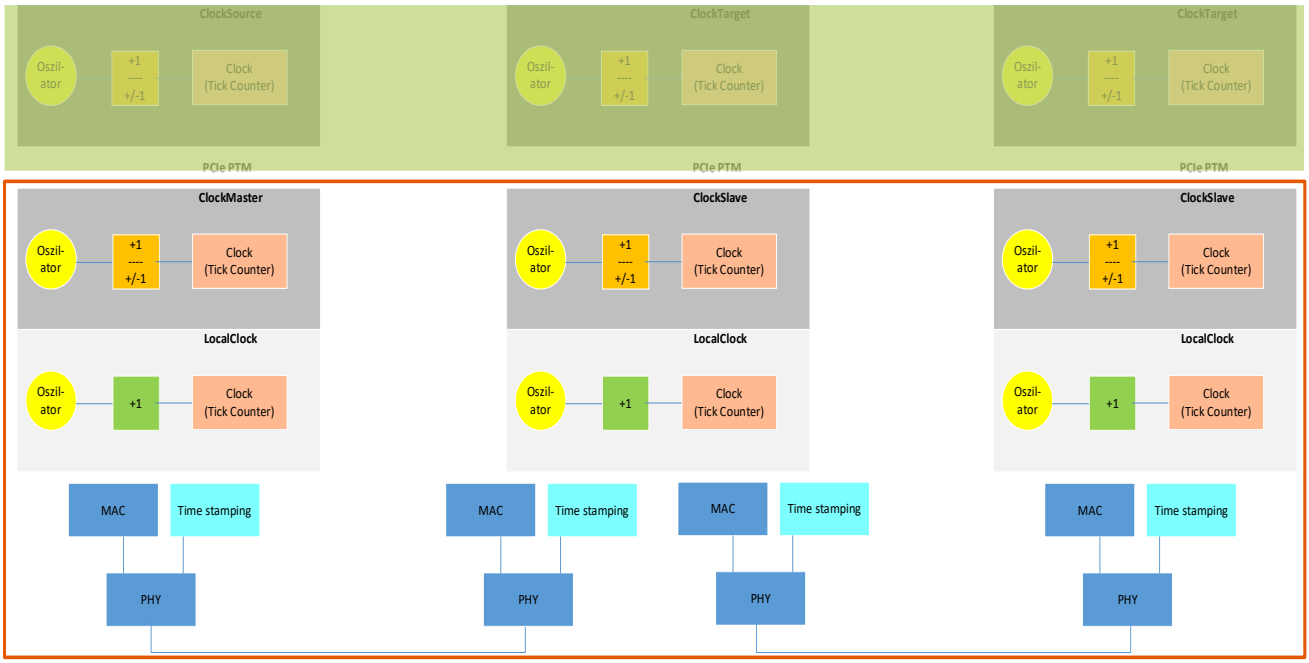


Figure 2: The simulation encompasses elements in the red frame. The errors between the ClockMaster and the ClockSource are assumed to be bounded with a given error budget. The same holds for differences between the ClockSlaves and the associated ClockTargets.

Modeling Oscillators

Both the free running and the controlled clocks are counters using oscillators. The latter are described by the following parameters:

Parameter	units	limit	Time profile
Nominal Frequency f_{nom}	Hz		constant
Fixed offset from f_{nom} (Δf)	ppm		constant
Frequency drift (df)	ppm/s	+/-1ppm/s for Master and Slaves	Known continuous function of time (or temperature)
Impulse frequency change	ppm		Instantaneous (very short time), due to mechanical shocks. Timing known.
Total frequency change from f_{nom} at any time	ppm	+/-50ppm for Slaves and +/-25ppm for Master	Constant limit over time

Table 1: Deviations form nominal frequency might be static or dynamic. The total deviation from the nominal frequency as well as the rate of continuous frequency changes are also limited. The oscillator model used in the simulator has to specify

the complete frequency evolution over time (e.g., it can be a periodic signal which satisfies the overall bounds of the last row in the above table). In Siemens simulations we also had vibrations, they can be specified if needed.

The values in the table above correspond to the TSN-IA profile but they can change. They must be specified before a simulation can start.

In some cases, the frequency change over time will not be specified directly but it must be calculated from the known temperature time profile. In this case a mapping between the temperature changes and the resulting frequency changes has to be known.

Modeling Control Loops

All controllers in Figure 2 have the same structure. The control loop is periodically given a signal with the desired value of the time (counter value), which is compared with the value of the current time. The difference is the error (offset) signal which is input to a controller (typically a PI controller). The output of the controller is the control signal, which is typically referred to as OCF (Offset Compensation Factor). This control signal is the input to the oscillator with a frequency "f", i.e. it modifies the integration of this frequency in order to minimize the offset. Hence, the controlled system is the counter obtained by integrating the oscillator's frequency "f" and adding or subtracting a tick after predefined intervals when necessary.

Since the desired signal is not present all the time, but comes periodically with a period "Tp" (where in the case of the ClockSlave controller Tp=Tsync, which is the Sync-Message interval), it is natural to implement the controller as a discrete time controller with the sampling frequency of 1/Tp. The controller is typically implemented as a ZOH (Zero-Order Hold) system, which means that the control signal remains constant within Tp.

The controller implementation in the case of ClockSlaves is described in the document (Obradovic, 2022).

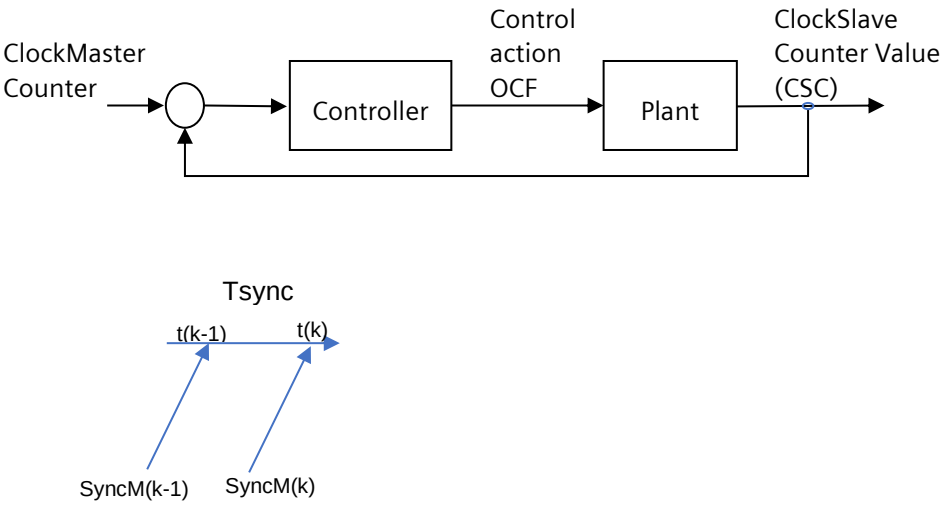


Figure 3: Closed-loop for controlling the ClockSlave counter by SW-adjustment of the oscillator frequency. The desired clock value is received by Sync Messages.

The dynamics of the controlled system clock (CSC) (the integral of the oscillator's frequency which is the free-running clock (FRC) in this case) is:

$$CSC(k) = CSC(k-1) + OCF(k-1) \cdot NumTicsFRC(within\ Tsync)$$

In the case of constant frequency "f" of the oscillator, the above equation becomes:

$$CSC(k) = CSC(k-1) + OCF(k-1) \cdot f \cdot Tsync$$

The corresponding transfer function in the "z" domain is then:

$$CSC(z) = \frac{f \cdot Tsync}{z-1} \cdot OCF(z)$$

The dynamics of the PI-controller is:

$$OCF(k) = OCF(k-1) + K_p \cdot (err(k) - err(k-1)) + K_I \cdot err(k-1) \cdot Tsync$$

The transfer function of the controller is then:

$$OCF(z) = \frac{K_p \cdot (z-1) + K_I \cdot Tsync}{z-1} \cdot err(z)$$

The parameters of the controller can be chosen as multiples of the oscillator's nominal frequency "fnom", such as $K_p = \frac{20}{fnom \cdot 1sec}$ and $K_I = \frac{80}{fnom \cdot 1sec}$. If the true frequency is equal to the nominal frequency, the frequency terms will cancel in the open loop transfer function P*C (where P stands for the controlled system (plant) and C for the controller). If they are not equal, their ratio will affect the gain of the open-loop.

The parameters of the controller have to be chosen to guarantee stability and performance of the clock for acceptable deviations from the nominal frequency of the oscillator, as well as the variations in Tsync and possible sporadic losses of Sync Messages (i.e. not receiving the desired time information after Tsync).

Therefore, in case of the PI controller implementation in controlled clocks, the following parameters are needed:

Parameter	units	limit	Time profile
Proportional Parameter K_p	no		constant
Integral Parameter K_I	1/s		constant
Nominal Frequency of the free oscillator	Hz		constant
Sampling Interval Tsync	s		Changes due to the delays in Sync message propagation
Different performance indicators such as bandwidth of the closed loop	Hz	Limited by the Nyquist frequency	constant

Maximal OCF at ClockMaster		+/- 3ppm/s	
Maximal OCF at ClockSlave		+/-250ppm/s	

Table 2: Parameters necessary for the PI controller design

Since the main goal of the controller is to track the MasterTime signal, the controller performance is related to its ability to track this signal. The bandwidth of the closed loop provides the information over which frequencies the tracking is possible (meaning, where the gains are not small, in this case > -3db). Hence, good tracking implies large bandwidth (here it is limited by the Nyquist frequency which is 1/(2*Tsync)), but at the same time the system should not track the noise (such as the plant or measurement noise) in the system. If the noise is high frequency noise, then the system bandwidth is typically chosen so that the gain becomes small over frequencies where the noise is large.

In the case of the here presented control application of Clock Synchronization, the noise is part of the MasterTime signal to be followed. Hence, a good tracking is possible only if the input signal has a high signal2noise ratio over the frequencies of interest. In other words, the controller will track the noise if the signal2noise ratio is not high.

If some other type of controller is used, the closed loop still has to satisfy the stability and good tracking performance not only for the nominal system but for all allowed variations around it. How to design controllers (e.g. how to choose the parameters of the PI controller) is out of scope of this document.

Simplifying the Master Element model in the simulation

We assume that the free running clock the ClockMaster uses is the LocalClock. The MasterClock controller follows an external time signal provided by a separate clock (ClockSource) attached to an oscillator

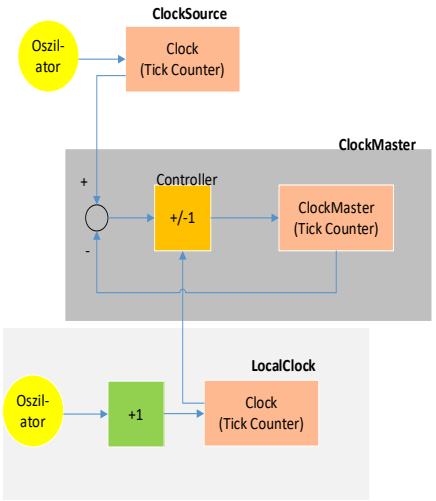
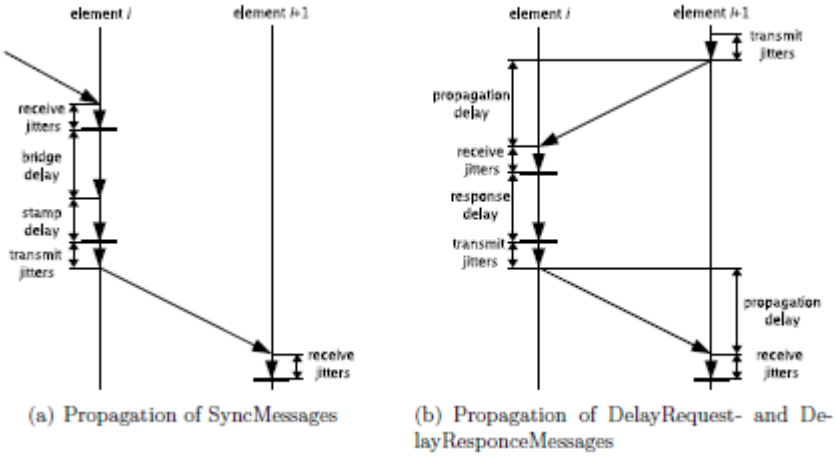


Figure 4: The ClockMaster is disciplining the counter of the LocalClock based on the time signal it receives from an external clock. Both external and internal oscillators have known frequency time evolution profiles.

The sending of the Sync messages is scheduled based on the LocalClock times, i.e. it happens after the fixed number of ticks of the LocalClock counter (obtained by multiplying the desired Tsync interval in seconds by the nominal frequency of the oscillator). This number remains constant over time, but the actual time between sending two Sync messages varies due to the frequency changes of the free running oscillator in the LocalClock.

Modeling the propagation time of Sync Messages and p-delay messages

The propagation of messages is shown in the figure below. The jitter at the receiver means that a message is not detected immediately but with some delays. Similarly, a message is not sent immediately but after some random delay which in this case is the transmit jitter.



(a) Propagation of SyncMessages

(b) Propagation of DelayRequest- and DelayResponceMessages

Figure 5: Propagation of a Sync Message between two slave elements as well as the propagation of the p-delay request and response messages

The necessary parameters for the simulation then are:

Parameter	units	limit	Time profile
Cable delay	s (or in clock tics)		constant
Bridge + stamp delay	s (or in clock tics)		Random, distribution has to be specified (e.g. uniform [5 125]µs)
Response Delay	s (or in clock tics)		Random, its distribution has to be specified
Receive jitter	s (or in clock tics)		Random, its distribution has to be specified
Transmit jitter	s (or in clock tics)		Random, its distribution has to be specified

Table 3: Parameters necessary for the PI controller design

Besides the jitter listed above, there is also a quantization error due to the frequency "f" of the clock used for stamping. Since the transmission is triggered by (certain number of) ticks, the transmit quantization error is equal to zero. On the receiver side, the message can arrive at any time, but it will be "noticed" only at the next tick. Hence, the receive quantization error is given as [0 1/f].

The forwarding or Sync messages can be conditioned with different criteria such as:

1) If there is no p-delay estimation available at receiving a Sync message, this message will not be forwarded
2) If two Sync messages are simultaneously at the same network element and the newer one is scheduled to be sent earlier that the older one (this is possible due to the variation in the residence time), then the older Sync message will be discarded (i.e. not forwarded)
3) Etc.

Calculation of the Rate-Ratios

Since the time is measured in different time frames in different elements (e.g. in the LocalClocks at different elements), and even in the same element (e.g. the LocalClock and the ClockSlave), the ratios between frequency of the mentioned clocks have to be estimated. RR denotes the rate ratio between the ClockMaster and the LocalClock in a given network element. The neighbor rate ratio nRR is on the other hand the ratio of LocalClock frequencies of the element “n” and “n-1”.

The calculation of nRR is based on the time information obtained via two p-delay request and response messages.

The calculation of RR can be done at the given network element directly by using two Sync messages or by cumulative multiplication of inverses of nRRs. Hence, there are two different methods of calculating RR.

ToDo: mention known pro/contras of these two methods.

A possible way to make the rate ratio variables unnecessary, is to start using the ClockSlave time, rather than the free-running LocalClock time, for stamping the Sync and also the p-delay request and response messages. This is possible only when the network elements are in state SYNCED. Hence, it cannot be applied at startup, but only after the system has stabilized and is synchronized.

Filtering for noise suppression

There are two main sources of errors in the system. One is the noise (jitter) which is present at any stamping operation. Hence, if we would like to measure the interval between two different stamping instances, the measurement will be the noisier the shorter this interval is. Hence, for minimizing the jitter influence, it is good to have large time intervals, meaning, among others, that the Tsync should be large.

On the other hand, large Tsync means that we observe the changes in frequencies rarely. One way of finding a compromise between wanting a high signal to noise ratio in time measurements and fast tracking of drifts, is to have Tsync not too big (e.g. 32µs) but to not use two consecutive Sync messages for the RR calculation, but 2 messages which are n*32µs apart (e.g. a typical value is n=7). The same principle is also applied for calculation of nRR and p-delays.

Some information about the averaging (using mean value for the p-delay estimation and median for the RR calculation based on Sync messages) is explained in (Steindl, Synchronization, 2022).

ToDo: include figures illustrating the above methods.

Fast Startup

This is a requirement in industrial use of Clock Synchronization. For some initial details see (Steindl, Synchronization, 2022).

ToDo: expand with different startup scenarios and make an estimate how soon the system will be in a SYNCED state.

Reference List

Obradovic, D. (2022). *60802-Obradovic-Controller-for-ClockSlaves-0622.pdf.*

Steindl, G. (2022). *60802-steindl-Sync-Error-Sources-v02_July2022.pptx.*

Steindl, G. (2022). *Synchronization.* Frankfurt.