Synchronization Performance of the Precision Time Protocol: Effect of Clock Frequency Drift on the Line Delay Computation

Ruxandra Lupas Scheiterer*, Dragan Obradovic*, Chongning Na*, Günter Steindl**, Franz-Josef Goetz***

* Siemens AG, Corporate Technology, Information and Communications, Munich, Germany
** Siemens AG, Automation and Drives, Industrial Automation Systems, Amberg, Germany

*** Siemens AG, Automation and Drives, Advanced Technologies and Standards, Nürnberg, Germany

(e-mails: {ruxandra.scheiterer, dragan.obradovic, na.chongning.ext, guenter.steindl, franz-josef.goetz} @siemens.com)

Abstract

The Precision Time Protocol (PTP) of the IEEE 1588 standard relies on two processes: the timing propagation process and the line delay estimation process. It is important to study the factors that affect the quality of these synchronization sub-processes, in order to expand the limit on the number of slaves synchronizable within a given synchronization precision. This short work-in-progress paper analytically studies the sensitivity of the line delay computation to linear clock frequency drift.

1. Introduction

Ethernet-based applications usually require the networked clocks to be synchronized. The Standard Network Time Protocol (NTP) [1], [2], executed over Ethernet provides synchronization accuracy at the millisecond level, which is appropriate for processes that are not time critical. However, in many applications, for example base station synchronization or motion control, where only sub-microsecond level synchronization errors are allowed, a more accurate synchronization solution is needed. The Precision Time Protocol (PTP), delivered by the IEEE 1588 standard [3] published in 2002 constitutes a promising Ethernet synchronization protocol, in which messages carrying precise timing information, obtained by the hardware time stamping in the physical layer, are propagated in the network to synchronize the slave clocks to a master clock. Boundary clocks adjust their own clock to the master clock and then serve as masters for the next network segment. Authors of [4], [5] introduced the transparent clock (TC) concept, in which intermediate bridges are treated as network components with known delay. By doing this, no control loop in the intermediate element is needed for providing timing information to the next local clock and hence the synchronization at the time client is not dependent on the control loop design in the intermediate bridges. The transparent clock concept has been adopted in the new version of IEEE 1588 published in 2007 (http://ieee1588.nist.gov/: Balloting on IEEE 1588 version 2 began on July 5, 2007).

The current state of the art is to guarantee a synchronization precision of 1µs for topologies with no more than 30 consecutive slaves. To expand this limit it is important to study the factors that influence the quality of the synchronization process and find out methods to minimize the effect of these factors.

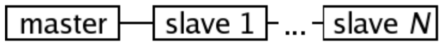
An important factor that affects the synchronization quality achievable by PTP is the stability of oscillators. Besides the long-term frequency drifts caused by aging, industrial environments are such that unpredictable and independent temperature changes at each node are commonly encountered, causing short-term frequency drifts, unless precluded by expensive temperature compensated (TCXO) or oven controlled (OCXO) crystal oscillators. Even those are affected by frequency drifts due to vibrations and shocks.

The peer-to-peer transparent clock implementation of PTP relies on two processes: timing propagation and line delay estimation. In this paper we analytically derive the expression for the error in line delay computation introduced by clock frequency drift.

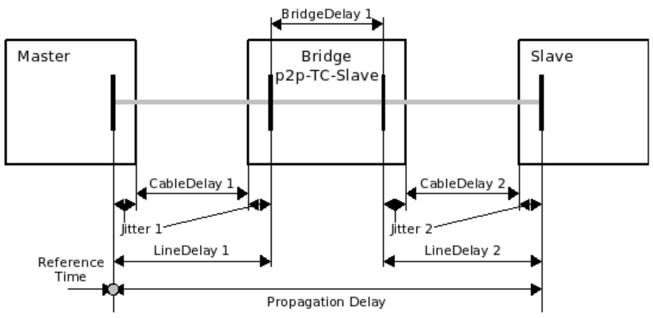
The paper outline is as follows: Section 2 introduces the system model and PTP protocol. Section 3 details the line delay estimation process, whose sensitivity to frequency drift is analyzed in Section 4. The embedding within our work in progress is spelled out in Section 5.

2. System model of PTP with transparent clocks and the timing propagation process

Since the standard leaves open the details, this section introduces our system model and notation. Fig. 1 shows a system with $N+1$ cascaded elements connected in a line topology. The PTP has a master/slave structure. The (grand)master provides the reference time to the other N elements, called slave elements, via time-aware bridges (TCs).



(a) Network topology



(b) System parameters

Fig. 1: System Model

Fig. 2 illustrates the two pillars of the time synchronization process, the process of propagation of Sync messages and the line delay estimation process.

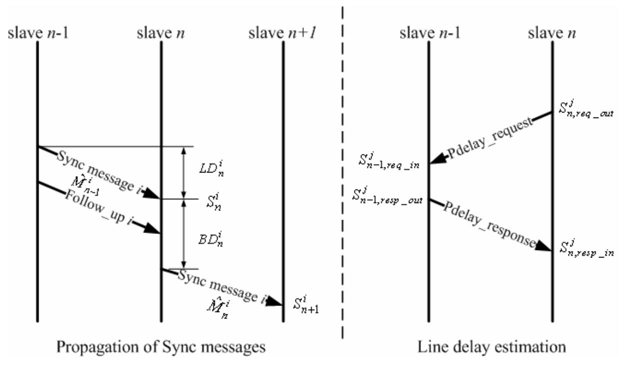


Fig. 2: PTP with transparent clocks

The master periodically sends Sync messages which carry the counter state of the master clock M^i, stamped at the time of transmission, and are propagated along the network. Quantities, certain or uncertain, linked with the Sync message transmitted by the master at time t_i are labeled by the superscript i. S^i_n is the time stamped at slave n upon arrival of Sync message i. A hat on a symbol means "estimate". The propagation time between the nth slave and its uplink element, LD^i_n, is called line delay. The message is forwarded to slave $n+1$ after a bridge delay BD^i_n. Slave n estimates its incurred line delay, $\hat{S}_n(LD^i_n)$, and own bridge delay, $\hat{S}_n(BD^i_n)$. Then it updates the received (estimated) master counter value $\hat{M}^i_{n-1}$ packaged in the Sync message by augmenting it with its own local delay, in order to pass on the best available estimate of the master time at the time of Sync message forwarding. The local delay is calculated by translating the sum of the own estimated line and bridge delays into master time. Bridge delays are recorded at each slave via the Sync message arrival and departure time stamps, while the line delays are estimated by using the line delay estimation process outlined in the next section. To express line delay and bridge delay in master time, each slave element needs to know its frequency offset to the master. The *rate compensation factor* (RCF, also called rate ratio, [6], [7]) is defined as the ratio between the frequencies of two different clocks. We use $RCF_{X/Y}$ to denote the frequency ratio between *X* and *Y*, i.e., ideally $RCF_{X/Y} = f_X / f_Y$. The rate offset to the master, RCF_{M/S_n}, is calculated by using the estimated master counter values in two Sync messages and the local counter values at the time when these messages arrive at slave n:

$$RCF_{M/S_n} = \frac{\hat{M}^i_{n-1} - \hat{M}^{i-1}_{n-1}}{S^i_n - S^{i-1}_n} \qquad (1)$$

Slave n then translates the delay measured in local counter value to master counter value by multiplying it with RCF_{M/S_n}. Hence, the estimated current master counter value is computed according to:

$$\begin{cases} \hat{M}^i_n = \hat{M}^i_{n-1} + \left(\hat{S}_n(LD^i_n) + \hat{S}_n(BD^i_n)\right) \cdot RCF_{M/S_n} \\ \hat{M}^i_0 = M^i \end{cases} \qquad (2)$$

3. The line delay estimation process

The last ingredient necessary for eq. (2) is the local estimate of the line delay to the predecessor. The *line delay estimation process* is shown on the right in Fig. 2, where j is the index of the line delay computation. This process uses 4 time-stamps: node n (the requestor) sends a delay request message to node $n-1$ and records its time of departure, S^j_{n,req_out} (1st). Node $n-1$ (the responder) replies with a delay response message which reports the time-stamps of receiving the delay request message and sending the reply, called "delay response message": S^j_{n-1,req_in} and $S^j_{n-1,resp\_out}$ (2nd and 3rd). The *responder delay* of node $n-1$ is $respD^j_{n-1}$ in absolute time (see Fig. 3), and is in local time:

$$S^j_{n-1,respD} = S^j_{n-1,resp\_out} - S^j_{n-1,req\_in}\,. \qquad (3)$$

Node n records the time when it receives the response message, $S^j_{n,resp\_in}$ (4th), which returned after a *requestor delay* of $reqD^j_n$ in absolute time, and in node n local time of:

$$S^j_{n,reqD} = S^j_{n,resp\_in} - S^j_{n,req\_out}\,. \qquad (4)$$

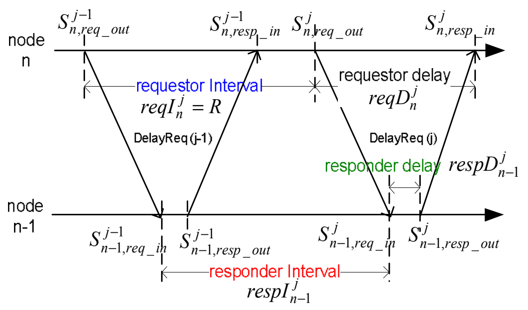


Fig. 3: Line delay and "RCF peer" computation

The responder delay is in counters of node $n-1$ and the requestor delay in counters of node n. To be able to subtract these time intervals, each element maintains an "RCF peer" estimate, i.e. frequency ratio estimate to its predecessor, which is obtained from two consecutive delay request and response cycles, as shown in Fig. 3:

$$RCF^j_{S_n/S_{n-1}} = \frac{S^j_{n,req\_out} - S^{j-1}_{n,req\_out}}{S^j_{n-1,req\_in} - S^{j-1}_{n-1,req\_in}}, \tag{5}$$

where the timeline is flipped to horizontal to gain labeling space. Then the line delay is estimated as:

$$\hat{S}_n(LD^j_n) = \frac{S^j_{n,reqD} - S^j_{n-1,respD} \cdot RCF_{S_n/S_{n-1}}}{2} \tag{6}$$

In the following section we will study the accuracy of this formula in the face of frequency drift, in the absence of other uncertainties. In this work we adopt the usual isolation approach when it is desired to identify the effect due entirely to one specific cause, and therefore neglect jitters (random transmission and reception time noise). Also, we assume zero delay skew, i.e. that the uplink and downlink line delays are equal. The latter is only a mild idealization, since the IEC61784-5-3 mandates stringent requirements for the Delay Skew. E.g. for PROFINET it may not exceed 20ns/100m.

4. Effect of frequency drift on the accuracy of the line delay estimate

We will repeatedly use the fact that in the absence of jitter the requestor and the responder intervals are equal in absolute time: $reqI^j_n = respI^j_n$, see Fig. 3. Denoting by $\bar{f}_n(I)$ the average frequency of node n in interval I, (5) is equivalent to:

$$RCF^j_{S_n/S_{n-1}} = \frac{\bar{f}_n(reqI^j_n)}{\bar{f}_{n-1}(respI^j_n)} \tag{7}$$

The Line Delay estimate is computed by each slave as a counter value increase (number of quartz oscillations). The responder delay from (3) is, in slave $n-1$ counters:

$$S^j_{n-1,respD} = \bar{f}_{n-1}(respD^j_{n-1}) \cdot respD^j_{n-1} \tag{8}$$

while without delay skew the requestor delay from (4) is, in slave n counters:

$$S^j_{n,reqD} = \bar{f}_n(reqD^j_n) \cdot \left(2 \cdot LD^j_n + respD^j_{n-1}\right). \tag{9}$$

Using (7)-(9), Slave$_n$'s Line Delay estimate from (6) becomes:

$$\hat{S}_n(LD^j_n) = \bar{f}_n(reqD^j_n) \cdot LD^j_n + \\ + \bar{f}_n(reqI^j_n) \cdot \frac{respD^j_{n-1}}{2} \cdot \left(\frac{\bar{f}_n(reqD^j_n)}{\bar{f}_n(reqI^j_n)} - \frac{\bar{f}_{n-1}(respD^j_{n-1})}{\bar{f}_{n-1}(respI^j_{n-1})} \right) \tag{10}$$

For constant frequencies all the average frequencies equal f_n respectively f_{n-1}, and (10) becomes:

$$\hat{S}_n(LD^j_n) = f_n \cdot LD^j_n \tag{11}$$

This ideal case is distorted in the case of non-constant frequencies during the estimation interval. To assess the size of the average frequencies in each time interval for changing clock frequencies, we assume a linear drift. In the case where nonlinear frequency changes occur, our analysis can be seen as a local first order approximation. Let the slope of the frequency change of clock k be Δ_k:

$$f_k(t_2) = f_k(t_1) + \Delta_k \cdot (t_2 - t_1) \tag{12}$$

or, equivalently:

$$f_k(t_2)/f_k(t_1) = 1 + \Delta_k \cdot (t_2 - t_1)/f_k(t_1) \tag{13}$$

For linear frequency drift the average frequency in an interval equals the frequency in the middle of the interval. Otherwise this is a good approximation for short time intervals. Hence, using (13), the quotients of the last term if (10) can be expressed as:

$$\begin{aligned} \frac{\bar{f}_n(reqD^j_n)}{\bar{f}_n(reqI^j_n)} &\approx \frac{f_n(mid\_reqD^j_n)}{f_n(mid\_reqI^j_n)} = \\ &= 1 + \Delta_n \cdot \frac{mid\_reqD^j_n - mid\_reqI^j_n}{f_n(mid\_reqI^j_n)} = \\ &= 1 + \Delta_n \cdot \frac{\frac{R + respD^j_{n-1}}{2} + LD^j_n}{f_n(mid\_reqI^j_n)} \end{aligned} \tag{14}$$

For the 3rd line we have used Fig. 3 to express the distance between the two interval middles. R is the length of the requestor interval, which is one of the design parameters of the synchronization algorithm. Likewise:

$$\begin{aligned} \frac{\bar{f}_{n-1}(respD^j_{n-1})}{\bar{f}_{n-1}(respI^j_{n-1})} &\approx \frac{f_{n-1}(mid\_respD^j_{n-1})}{f_{n-1}(mid\_respI^j_{n-1})} = \\ &= 1 + \Delta_{n-1} \cdot \frac{mid\_respD^j_{n-1} - mid\_respI^j_{n-1}}{f_{n-1}(mid\_respI^j_{n-1})} = \\ &= 1 + \Delta_{n-1} \cdot \frac{\frac{R + respD^j_{n-1}}{2}}{f_{n-1}(mid\_respI^j_{n-1})} \end{aligned} \tag{15}$$

The i^{th} Sync message, generated by the master at time t_i, arrives at slave n after a propagation time of L^i_n, called latency. We introduce the "age" $A^{i,j(i)}_1$ of the line delay computation j valid for Sync Message i to be the time elapsed between the middle of the last line delay computation interval and the arrival time of the current Sync message:

$$A_1^{i,j(i)} = t_i + L_n^i - \text{middle of Request Delay Interval}. \quad (16)$$

This enables us to write the frequency in the line delay computation expression more transparently as:

$$\bar{f}_n(reqD_n^j) \approx f_n(mid\_reqD_n^j) = f_n(t_i + L_n^i - A_n^{i,j(i)}). \quad (17)$$

Inserting (14), (15) and (17) into (10), we obtain:

$$\begin{aligned}\hat{S}_n(LD_n^j) &\approx f_n(t_i + L_n^i - A_n^{i,j(i)}) \cdot LD_n^j + \\ &+ \frac{respD_{n-1}^j}{2} \cdot [\Delta_n \cdot \left(\frac{R + respD_{n-1}^j}{2} + LD_n^j \right) - \\ &- \Delta_{n-1} \cdot \frac{R + respD_{n-1}^j}{2} \cdot \frac{f_n(mid\_reqI_n^j)}{f_{n-1}(mid\_respI_{n-1}^j)}]\end{aligned} \quad (18)$$

For typical values: a response delay $respD_{n-1}^j \leq 100ms$, a requestor interval of $R \approx 300ms$ and line delays of $LD_n^j \approx 100ns$, in the 2nd line the line delay is smaller than the preceding term by roughly a factor of 10^{-7}; also, looking at the very last term in (18), the frequency ratio of two identical quartzes S_n and S_m with manufacturing tolerance p from the common nominal frequency is contained in $f_n / f_m \in [\frac{1-p}{1+p}, \frac{1+p}{1-p}] \in [1 - 2 \cdot p, 1 + 3 \cdot p]$; the relative error by approximating it by 1 is $O(10^{-4})$ even for p as high as 100ppm. Therefore we can closely approximate $\hat{S}_n(LD_n^j)$, the line delay estimate computed by slave n, by neglecting both these terms, as:

$$\boxed{\begin{aligned}\hat{S}_n(LD_n^j) &\approx f_n(t_i + L_n^i - A_n^{i,j(i)}) \cdot LD_n^j + \\ &+ respD_{n-1}^j \cdot \frac{R + respD_{n-1}^j}{4} \cdot (\Delta_n - \Delta_{n-1})\end{aligned}} \quad (19)$$

The 1st term in this expression is the one that contains the desired line delay component, however measured as a clock counter increase that was driven by the slave clock frequency present at the middle of the line delay computation interval, which can be as far back from the current Sync message as the maximal possible age of a line delay computation. This expression also shows that, if the two slaves engaged in a line delay estimation process have drifting frequencies during the respective time interval, the desired line delay estimate (given by the 1st term) incurs an error, given by the 2nd term. This error grows linearly with the relative drift of the two clock frequencies; linearly with the requestor interval size, i.e. the interval between two line delay estimations, and quadratically with the duration of the responder delay, i.e. the time it takes the responder to transmit the response message.

5. Current and future work

This work is a first completed part of our work in progress, whose goal is to make more precise our results on synchronization accuracy obtained in [8]. There we derived the error expression on the nth Slave's master counter estimate for the scenario of drifting master clock frequency. The error caused by this drift to the 1st slave's line delay estimate was neglected at that time, an error which is however passed down the line from slave to slave. We are also using this result in our current derivation of the error expression for drifting slave clocks, where the above line delay error contributes one additional synchronization error term for each clock pair with a drifting partner; and we are using it in our ongoing comparison of the relative importance of master versus slave clock stability. We plan to report our findings presently.

References

[1] D. L. Mills, "Internet time synchronization: The network time protocol", *Network Working Group Request for Comments*, 1989.

[2] D. L. Mills, "Precision synchronization of computer network clocks", *ACM SIGCOMM Computer Communication Review*, Vol. 24, pp. 28-43, 1994.

[3] IEEE, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. IEEE, New York. ANSI/IEEE Std 1588-2002, 2002.

[4] J. Jasperneite, K. Shehab, K. Weber, "Enhancements to the time synchronization standard IEEE-1588 for a system of cascaded bridges", in: *Proc. of 2004 IEEE International Workshop on Factory Communication Systems*, Vienna, 2004.

[5] J. Jasperneite, P. Neumann, "How to guarantee real-time behaviour using Ethernet", in: *Proc. of 11th IFAC Symposium on Information Control Problems in Manufacturing (IN-COM2004)*, Salvador-Bahia, 2004.

[6] IEEE, *IEEE P1588TM D2.2 Draft Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE, New York, 2007.

[7] D. Obradovic, R.L. Scheiterer, C. Na, G. Steindl and F. J. Goetz, "Clock Synchronization in Industrial Automation Networks: Comparison of different Syntonization Methods", accepted at: *ICINCO 2008*, Portugal.

[8] C. Na, D. Obradovic, R. L. Scheiterer, G. Steindl and F. J. Goetz, "Synchronization Performance of the Precision Time Protocol", in: *2007 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Vienna, 2007.