A Recursive Method for Clock Synchronization in Asymmetric Packet-based Networks

Mohammad Javad Hajikhani, Thomas Kunz, Howard Schwartz
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
Email: mohammadjavad_hajikhani@carleton.ca, tkunz@sce.carleton.ca, schwartz@sce.carleton.ca

***Abstract*—In the context of the IEEE 1588 Precision Time Protocol (PTP), estimating the delay's bias is a problem that appears in both one-way (using transparent devices) or two- way message exchange mechanisms. For estimating the offset via the two-way message exchange mechanism, it is usually assumed that the expected value of delays in forward and reverse directions are equal. However this is not a realistic assumption for packet-based wide area networks, where delays in down-link and up-link directions may have a significant difference. In this work we propose a solution to estimate the random delay's bias and improve the synchronization accuracy of IEEE 1588. Our method is easy to implement and is compatible with the current version of the protocol. We compared our results with no bias correction and the Boot-strap method. In addition to the improvement in synchronization accuracy, our method allows us to update the slave clock recursively. The proposed method works well even in the presence of large frequency offsets and can also be implemented by using different filters.**

***Index Terms*—IEEE 1588, clock synchronization, two-way message exchange mechanism, asymmetric delays, Boot-strap method, Gamma PDV.**

I. Introduction

The IEEE 1588 PTP standard specifies a clock synchronization protocol to synchronize nodes in a distributed network [1]. The synchronization process consists of two tasks. First, to synchronize the nodes to one common time by estimating their offset with respect to a reference time (phase offset). The second task is to correct the nodes' clock frequency drift relative to a reference frequency (frequency offset). Synchronization happens by exchanging timing information between nodes in a network and a master node. Using the two-way message exchange mechanism, the master clock periodically sends its local time to the slave clock as a Sync message. The arrival time of the Sync message is recorded at the slave node. Then, the slave node sends its local time as a Delay-Req signal to the master. The arrival time of the Delay-Req message is recorded by the master node and is sent back as a Delay-Resp message to the slave node [1]. The offset in the slave local clock is adjusted based on these collected values. This method works well only under the assumption of symmetrical delays, which means that the down-link (from master to slave) and up-link (from slave to master) delays are equal. Although this assumption is not true in practice, for some applications it is sufficient to assume that the expected value of delays in the forward and reverse direction are equal with each other. This assumption reduces the synchronization process to an unbiased estimation problem and has been studied widely, for example in [2]-[5]. In [6], Freris et al. proposed a method to estimate the relative speedup of clocks in a network and subsequently estimating the pairwise offset of them. However, for estimating the offset it is still required to be assumed that delays are symmetric. To provide an accurate synchronization in a multihop wireless network a distributed algorithm is presented in [7] and [8]. Although the aforementioned algorithm forces the summation of all the nodes' offsets to zero, each node may still have non-zero offset due to asymmetric delays. Unfortunately, in packet-based wide area networks such as metro Ethernet, it happens frequently that delays in one direction are dominant and even the expected value of delays in down- link and up-link directions are not equal. As a result we deal with a biased estimation problem.

The synchronization process can also be modeled as a filtering problem. Bias-aware filters like those which have been studied in [9] and [10] would appear to be a straightforward solution. Unfortunately, in the synchronization problems, bias-aware filters cannot estimate the value of the bias successfully. This inability is because of existing dependencies among equations. In other words, there are not enough independent equations to estimate the bias. This issue has thoroughly been discussed in [11]-[13]. For instance, the first theorem from [11] provides a ranked-based proof to exhibit the fundamental indeterminacy in estimating the offset, when delays in the down-link and up-link directions are not equal.

One approach to the problem of inequality in the down-link and up-link delays is to use packet filtering techniques. This group of solutions tries to find some "good" packets based on specific conditions. The definition of a good packet is subjective and varies from one paper to another. In a number of papers such as [14] and [15], after collecting a number of received messages (this number is determined by the window size) the good packet is the one with the least delay. Such algorithms are usually referred to as *sample-minimum* filters. In [16], instead of minimum or maximum, the mode is the criterion for selecting good packets. The authors of [17] proposed using a train of probing packets following the Sync and Delay-Req messages. The aim of using probing packets is to estimate the occurrence of delay in timing packets by measuring the relative time difference between packets. The packets with large value of delays then will be filtered

out. Although using packet filtering methods can improve the synchronization accuracy, none of these methods actually estimates the value of bias (or provide an unbiased estimator of the phase offset). One other problem with this group of solutions is that we need to filter a large number of packets. Dropping packets means losing information and also causes more delay before correction (or alternatively requires more network traffic overhead to collect a window's worth of data within a single correction interval).

Another approach is to directly estimate the bias or in other words to provide an unbiased estimator of the offset. In [18] and [19], Jesk et al. used the Boot-strap method to find the order statistic-based best linear unbiased estimator (o-BLUE) of the clock offset under the assumption that Packet Delay Variations (PDV) have exponential or Pareto distributions. But more recent papers such as [20] and [21] show that the Gamma distribution is a better choice for modeling the PDVs in networks. The authors of the Boot-strap method also have not provided any solution for estimation and correction of the frequency offset concurrently with the phase offset.

The results of [20] and [21], modeling the network delay by Gamma distributions, is the main motivation and one key assumption of this work. In this paper we introduce a solution to estimate the bias and we compare our results with the case with no bias correction and also with the Boot-strap method. Our method is easy to implement and is also compatible with the current version of the IEEE 1588 PTP.

II. IEEE 1588 OVERVIEW

The IEEE 1588 PTP enables us to estimate the phase offset and the frequency offset of a so called slave clock with respect to a master clock. The master clock is assumed to be synchronized with the reference time, which may come from GPS signals. The synchronization process using the two-way message exchange mechanism happens by exchanging timing packets between master and slaves. Each timing message arrives at its destination after a delay. This delay can be divided into fixed and random parts. The fixed part of the delay comes from the processing time of the devices in the network and also the propagation time. The random part of the delay usually comes from the random queuing delays due to the network traffic load [14]. At the n_{th} synchronization interval, a master sends the n_{th} Sync message which includes its local time at t_n. A slave receives this message at $C(t_n + d_{ms} + x_n)$. The term $C(t)$ represents the local time of the slave clock when the master time is t and $d_{ms}+x_n$ is the time that a packet takes to travel from master to slave or the down-link delay. Here d_{ms} is the fixed part and x_n is the random part of the delay. Then the slave node sends its local time as a Delay-Req message at $C(t'_n - d_{sm} - y_n)$ and the master receives this message at t'_n, where $d_{sm} + y_n$ is the up-link delay. Similar to the down-link delay d_{sm} is the fixed part and y_n is the random part of the delay. Finally all these numbers are collected at the slave side. Fig. 1 shows the above process.

From Section 7.4.2 of [1], messages from master to slave and slave to master traverse the same path in order to minimize the asymmetry. As a result it can be assumed that the fixed parts of the delays in the down-link and up-link directions are equal. If for any reason the fixed parts of the delays in the down-link and up-link directions differ significantly, then they need to be measured. Because the fixed parts of the delays do not change during the synchronization process, this measurement can be done once at the beginning. Knowing the process time of the devices between the master and slave nodes, or measuring the delay when the traffic load is at the minimum might be some possible ways for estimating the fixed parts of the delays. In this work, we assume that fixed parts of the delays in the down-link and up-link directions are equal. Consequently, d_{ms} and d_{sm} can be replaced with the term d. The phase and frequency offsets can then be estimated by the following equations:

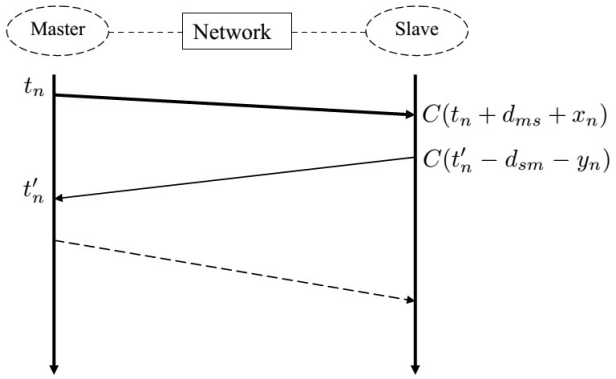

Fig. 1. Two-way message exchange mechanism

$$\theta_m(n) = \frac{[C(t_n + d + x_n) - t_n] - [t'_n - C(t'_n - d - y_n)]}{2} \quad (1)$$

$$f_m(n) = \frac{\theta_m(n) - \theta_m(n-1)}{t_n - t_{n-1}} \quad (2)$$

The denominator of (2) represents the time difference between two successive synchronization interval. Master and slave clocks are perfectly synchronized if $C(t) = t$, but if we assume that the phase offset at the n_{th} synchronization interval is $\theta(n)$, we would have:

$$C(t_n) = t_n + \theta(n)$$

For short enough intervals (the size depends on how large the frequency offset is) it is possible to assume that the phase offset is unchanged, so:

$$C(t_n + \Delta) = t_n + \Delta + \theta(n)$$

By expanding the slave clock's terms in (1) by using the above equation, (1) can be rewritten as:

$$\theta_m(n) = \theta(n) + \frac{x_n - y_n}{2} \quad (3)$$

From (3) it is clear that (1) is a good estimation of $\theta(n)$ only if x_n and y_n are equal. This condition is typically not true, specially in packet-based wide area networks where even the expected value of these delays may not be equal. Inequality in the expected values of these delays causes a bias

in the convergence point of the estimated offset that should be removed.

The synchronization process can also be modeled as a filtering problem. For this purpose we should firstly determine the state and the measurement equations. By using the results of [22], the state equations can be written as:

$$\theta(n) = \theta(n-1) - u_\theta(n-1) + [f(n-1) - u_f(n-1)].\Delta T + \omega_\theta(n-1)$$

and

$$f(n) = f(n-1) - u_f(n-1) + \omega_f(n-1)$$

The terms $\omega_\theta(n)$ and $\omega_f(n)$ are the process noises for the phase offset and frequency offset respectively. The terms $u_\theta(n)$ and $u_f(n)$ are the estimated values for the phase and frequency offsets. ΔT is the time difference between two successive synchronization intervals and is the same as the denominator of (2). The above equation means that the next phase offset would be the previous one, minus the last estimated phase offset, plus the drift that happens between two successive corrections, plus a process noise. For the frequency offset, the next offset is the previous one, minus the last estimated frequency offset plus a process noise. From (1) (or (3)) and (2) the measurement equations can be written as:

$$\theta_m(n) = \theta(n) + \nu_{\theta_m(n)}$$

$$f_m(n) = f(n) + \nu_{f_m(n)}$$

Where $\nu_{\theta_m(n)}$ and $\nu_{f_m(n)}$ are the measurement noises for the phase offset and the frequency offset respectively. The measurement noise for the phase offset from (3) is determined by $\frac{x_n - y_n}{2}$. As we already explained, the mean value of the down-link and up-link delays are not necessarily equal and as a result the mean value of the measurement noise for the phase offset in general is not zero. In contrast, since in the numerator of (2) there is a subtraction of two successive phase offsets, in terms of the bias, they cancel each other out and therefore (2) provides an unbiased estimator for the frequency offset. The state and measurement equations can be written in matrix form as follows:

$$\underline{\chi}(n) = \mathbf{A}\underline{\chi}(n-1) + \mathbf{B}\underline{u}(n-1) + \underline{w}(n-1) \tag{4}$$

$$\underline{z}(n) = \mathbf{H}\underline{\chi}(n) + \underline{\nu}(n) \tag{5}$$

Where $\underline{\chi}(n) = [\theta(n) \quad f(n)]^T$, $\underline{u}(n-1) = [u_\theta(n-1) \; u_f(n-1)]^T$, $\underline{w}(n-1) = [w_\theta(n-1) \; w_f(n-1)]^T$, $\underline{z}(n) = [\theta_m(n) \; f_m(n)]^T$ and $\underline{\nu}(n) = [\nu_{\theta_m(n)} \; \nu_{f_m(n)}]$. Matrices **A**, **B** and **H** are:

$$\mathbf{A}= \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}, \mathbf{B}= \begin{bmatrix} -1 & -\Delta T \\ 0 & -1 \end{bmatrix}, \mathbf{H}= \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}$$

Equations (4) and (5) are a linear pair of state and measurement equations that can be solved using different filters such as a Kalman filter [22]. A summary of [22] is as follow. At the n_{th} synchronization interval, the slave node receives $\underline{z}(n)$. This vector is passed into the filter. The output of the filter is the vector $\underline{u}(n)$. The vector $\underline{u}(n)$ is subtracted from the slave clock in order to correct the phase and frequency offsets. In the simulations section, we call the method of using such a filter to estimate and correct the offset, the *Basic 1588* method. The non zero mean value of the measurement noise for the phase offset creates a bias in the estimated phase offset by the filter. As it has been shown in [11]-[13], for asymmetric delays, equations (4) and (5) are not sufficient to estimate the offset. As a result, in order to remove the bias from the estimated offset, another solution needs to be developed. In the next section we study the random characteristics of the delays.

III. Packet Delay Variation in Networks

In [20] and [21], it was shown that Packet Delay Variation (PDV) in multi-hop Ethernet networks and for strict priority queuing can be modeled by a Gamma distribution. The parameters of the PDF are a function of the traffic load and also the number of hops. The PDF of a Gamma distribution is:

$$f_X(x) = \begin{cases} \frac{x^{\alpha-1}}{\Gamma(\alpha)\beta^\alpha} e^{\frac{-x}{\beta}} & x > 0 \\ 0 & x < 0 \end{cases} \tag{6}$$

where α and β are called the shaping and scale parameters respectively. The exponential distribution is a special case of the Gamma distribution for the case that the shaping parameter is equal to one. In [18], Jesk *et al.* used a Boot-strap solution to provide an unbiased estimator of the phase offset, under the assumption that the PDV can be modeled by an exponential distribution, but the results of [20] and [21] show that the shaping parameter is not always close to one. For example for 80% traffic load, over a 5 hops network, the shaping parameter in [21] is determined as 11. This difference causes an error in the estimated value of phase offset for the Boot-strap method.

In the next section, we introduce a recursive method for estimating the bias and correcting the slave clock. We call our method the *Gamma Based Bias-Correcting 1588* (G-BC 1588) method. We have developed our method for a Gamma distributed PDV. In Section **VI**, we compare our solution with Basic 1588 and the Boot-strap method.

IV. Proposed Solution: Gamma Based Bias-Correcting 1588 Method

In this section we develop our solution for estimating the bias. For the sake of simplicity, we first assume that the frequency offset is zero and a fixed phase offset is the only source of inaccuracy which needs to be estimated. It is worth to note that higher orders of error which can be captured by the process noise have not been considered here. The reason is that phase and frequency offsets are the two main sources of inaccuracy and higher orders of error can be avoided if the synchronization process is repeated fast enough. From the last section, we know that the random delays from master to slave and from slave to master can be modeled by a Gamma distribution. Following Section II, we denote the subtraction of

a master time from the slave time at the n_{th} synchronization interval by $\delta_{DL}(n)$. Similarly, for the up-link direction we denote the subtraction of a slave time from the master time by $\delta_{UP}(n)$. So for the down-link direction we have:

$$\delta_{DL}(n) = C(t_n + d + x_n) - t_n = \theta + d + x_n \tag{7}$$

and for the up-link:

$$\delta_{UP}(n) = t'_n - C(t'_n - d - y_n) = -\theta + d + y_n \tag{8}$$

Where x_n and y_n represent the down-link and up-link delays respectively and θ is the phase offset. Our method estimates the bias for the down-link and up-link separately and similarly, so at this point let us just consider the down-link delay. If the random variable X represent the down-link delay, then its PDF is determined by (6). The CDF of the Gamma distribution is:

$$F_X(x) = \begin{cases} \frac{\gamma(\alpha, \frac{x}{\beta})}{\Gamma(\alpha)} & x > 0 \\ 0 & x < 0 \end{cases} \tag{9}$$

Where $\gamma(\alpha, \frac{x}{\beta})$ is an incomplete Gamma function and is determined by:

$$\gamma(\alpha, \frac{x}{\beta}) = \int_0^{\frac{x}{\beta}} y^{\alpha-1} e^{-y} dy \tag{10}$$

The expected value of the Gamma distribution is determined by the multiplication of α and β, so:

$$E(X) = \alpha\beta \tag{11}$$

It needs to be emphasized that the values of α and β are unknown, so we are not able to calculate the expected value of the delay using (11).

Now assume that instead of one Gamma random variable, we consider the minimum of two successive random variables. In the context of IEEE 1588, this means that when the receiver receives the $(2i-1)_{th}$ Sync message, it waits for the next one, then takes the following minimum:

$$\begin{aligned} \delta'_{DL}(i) = min \quad & [\quad (C(t_{2i-1} + d + x_{2i-1}) - t_{2i-1}) \\ & , \quad (C(t_{2i} + d + x_{2i}) - t_{2i})] \\ = min[(\theta + d + x_{2i-1}), (\theta + d + x_{2i})] \\ = \theta + d + min(x_{2i-1}, x_{2i}) \end{aligned} \tag{12}$$

Where DL stands for down-link.

It is known from stochastic text books (for example [18]) that if $W = min(X, Y)$, where X and Y are independent random variables, then the distribution of W is:

$$f_W(w) = f_X(w) + f_Y(w) - [f_X(w)F_Y(w) + f_Y(w)F_X(w)] \tag{13}$$

Where f and F represent the PDF and CDF of the random variables respectively. If we assume that X and Y are from a Gamma distribution, then by using (6) and (9), the distribution of W is determined by:

$$f_W(w) = \frac{2w^{\alpha-1}e^{\frac{-w}{\beta}}}{\Gamma(\alpha)\beta^\alpha} \left[1 - \frac{\gamma(\alpha, \frac{w}{\beta})}{\Gamma(\alpha)}\right] \tag{14}$$

For finding the mean value of W, we expand $\gamma(\alpha, \frac{w}{\beta})$ by the following series [24]:

$$\gamma(\alpha, \frac{w}{\beta}) = (\frac{w}{\beta})^\alpha \Gamma(\alpha) e^{\frac{-w}{\beta}} \sum_0^\infty \frac{(\frac{w}{\beta})^k}{\Gamma(\alpha + k + 1)} \tag{15}$$

By substituting (15) in (14), we can rewrite (14) as:

$$\begin{aligned} f_W(w) = \frac{2w^{\alpha-1}e^{\frac{-w}{\beta}}}{\Gamma(\alpha)\beta^\alpha} - \sum_0^\infty [& \frac{e^{\frac{-w}{(\beta/2)}} w^{2\alpha+k-1}}{(\beta/2)^{2\alpha+k}\Gamma(2\alpha+k)} \\ & \frac{\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}] \end{aligned} \tag{16}$$

Now by taking the mean value, we finally have:

$$E(W) = 2\alpha\beta - \sum_0^\infty \frac{(2\alpha+k)(\beta/2)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)} \tag{17}$$

So the expected value of W can be determined by (17).

Let us assume that the slave node just uses the sequence of data that comes from (7) to estimate the value of θ. The slave node passes the data into a filter (for example a Kalman filter). Since the expected value of the random delay is not zero (in this case it is the mean value of the Gamma distribution), the convergence point of the filter is:

$$\Delta_{DL} = \theta + d + \alpha\beta \tag{18}$$

Now assume that the slave clock uses (12) to estimate the value of θ. So it passes the result of (12) into a filter. As a result, the new convergence point would be:

$$\Delta'_{DL} = \theta + d + E(W) \tag{19}$$

where $E(W)$ is determined by (17). By subtracting (19) from (18):

$$\Delta_{DL} - \Delta'_{DL} = \alpha\beta - E(W)$$

$$\Delta_{DL} - \Delta'_{DL} = \alpha\beta - \left[2\alpha\beta - \sum_0^\infty \frac{(2\alpha+k)(\beta/2)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}\right]$$

$$\Delta_{DL} - \Delta'_{DL} = -\alpha\beta + \frac{\beta}{2} \underbrace{\sum_0^\infty \frac{(2\alpha+k)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}}_{f(\alpha)}$$

$$\Delta_{DL} - \Delta'_{DL} = -\alpha\beta + \frac{\beta}{2} f(\alpha) \tag{20}$$

where $f(\alpha)$is:

$$f(\alpha) = \sum_0^\infty \frac{(2\alpha+k)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)} \tag{21}$$

From (20) the following expression for estimating the mean value can be concluded:

$$\boxed{\widehat{\alpha\beta} = \frac{\widehat{\Delta_{DL}} - \widehat{\Delta'_{DL}}}{\frac{f(\alpha)}{2\alpha} - 1}} \tag{22}$$

where $\widehat{\Delta_{DL}}$, $\widehat{\Delta'_{DL}}$ and $\widehat{\alpha\beta}$ are the estimated values for Δ_{DL}, Δ'_{DL} and the mean value of the Gamma distributed delay. In other words, equation (22) provides an explicit equation for estimating the bias. The problem with this equation however, is the unknown value of α.

To proceed, we first focus on $f(\alpha)$ to study its characteristics. Fig. 2 shows $f(\alpha)/\alpha$ versus α:

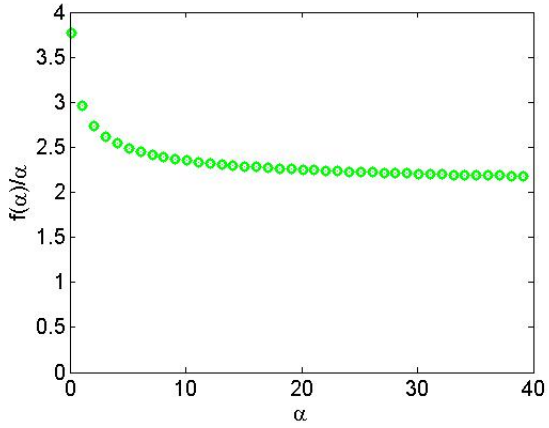

Fig. 2. $f(\alpha)/\alpha$ versus α

As we observe in Fig. 2, as α decreases, $f(\alpha)/\alpha$ changes more rapidly. For large values of α, this ratio gets closer to a constant number. By trial and error, we notice that:

$$\frac{f(\alpha)}{\alpha} \simeq 2(1 + \frac{0.56}{\sqrt{\alpha + 0.3}}) \tag{23}$$

Equation (23) provides an equation much simpler than (21) for calculating the denominator of (22).

In order to estimate the shaping parameter, in addition to (22) one more independent equation is needed This equation can be found from the variance of the delay. So in the following, we first develop a method for estimating the variance of the delay.

The bias estimator which is discussed above, updates the slave clock not at each iteration, but after two iterations. This characteristic provides for us a simple solution for estimating the variance of the noise. Fig. 3 shows two typical signal exchanges between a master and a slave for the down-link direction. We compute the following expressions:

$$\begin{aligned}(\delta(2i) - \delta(2i-1))^2 &= ((\theta + x_{2i} + d) - (\theta + x_{2i-1} + d))^2 \\ &= (x_{2i} - x_{2i-1})^2 \\ &= x_{2i}^2 + x_{2i-1}^2 - 2x_{2i}x_{2i-1}\end{aligned}$$

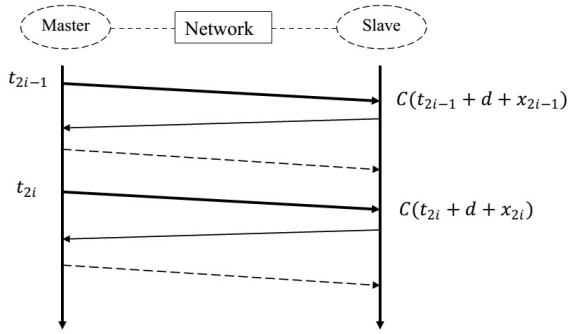

Fig. 3. Two successive message exchanges for the down-link direction

where $\delta(i) = C(t_i + d + x_i) - t_i$. By taking the expected value:

$$\begin{aligned}E[(\delta(2i) - \delta(2i-1))^2] &= E[x_{2i}^2 + x_{2i-1}^2 - 2x_{2i}x_{2i-1}] \\ \xrightarrow{x_{2i} \text{ and } x_{2i-1} \text{ are i.i.d} \sim X} &= 2(E(X^2) - E^2(X)) = 2\sigma^2\end{aligned}$$

As a result, the above expected value can give us an estimate of the down-link delay variance. A similar approach is then taken for the up-link delay. Note that because we correct the slave clock after two iterations, the phase offset (θ) in these two iterations is almost constant, so they cancel each other out.

On the other hand, the variance of the Gamma distribution is equal to:

$$\sigma^2 = \alpha\beta^2 \tag{24}$$

where $\alpha\beta^2$ is the expression for the variance of a Gamma distribution. By combining (22), (23) and (24), we end up with the following closed form equation for estimating α:

$$\hat{\alpha} = \frac{0.3}{\left(\frac{0.56\hat{\sigma}}{\widehat{\Delta_{DL}} - \widehat{\Delta'_{DL}}}\right)^2 - 1} \tag{25}$$

where $\hat{\alpha}$ and $\hat{\sigma}$ are the estimated values for the shaping parameter and variance respectively. All the variables on the right hand side of (25) can be estimated recursively.

The synchronization process can be summarized as follows. After two successive synchronization intervals, the estimated values for σ, Δ_{DL} and Δ'_{DL} are updated. These new estimated values are plugged into (25) to update the estimate of α. Then the new estimated value for α is substituted in (22) in order to give us a new estimation of the bias. This process is done for the down-link and up-link directions separately.

One problem with (25) is that the denominator of this equation may get very close to zero. To see the reason, we rewrite (25) as follow:

$$\left(\frac{0.56\widehat{\sigma}}{\widehat{\Delta_{DL}} - \widehat{\Delta'_{DL}}}\right)^2 - 1 = \frac{0.3}{\widehat{\alpha}}$$

As we observe, specially for larger values of α the above

expression is a small number. Consequently, during the estimation process, it is possible that the estimated values for σ, Δ_{DL} and Δ'_{DL} make the denominator of (25) very close to zero which can cause computational problems.

In order to solve this problem, we apply two additional mechanisms to the algorithm. The first mechanism is to define a reasonable neighborhood for α. By defining a neighborhood for α, we can filter out all the bad estimated values of α before using them for estimating the bias. This neighborhood should not be necessarily very tight and is just for preventing very poor estimates of α. For example let us denote the estimated value of α from (25) for the down-link direction as $\hat{\alpha}_{DL}$. If the value of $\hat{\alpha}_{DL}$ is more than a lower boundary α^l_{DL} and less than an upper boundary α^u_{DL}, then it would be used for estimating the bias. The second mechanism that we use is to take the average of the estimated values for α. Taking the average smooths the results and will reduce the effect of bad estimated values. The first few estimated values may not be accurate enough, so we start to take the average of estimated values after a predetermined number of iterations. A summary of these two additional mechanisms is as follows. First we update our estimates of α using (25). If the estimated value falls between α^l_{DL} and α^u_{DL}, it can be used for the estimation of the bias. For a few initial estimated αs, we use them directly to update the bias (in the algorithm bellow, we consider the ten first iterations). After a predetermined number of iterations we start to average the estimates.

For estimating the frequency offset, since the bias correction happens after receiving two timing packets, equation (2) needs to be changed slightly as follow:

$$f_m(n) = \frac{\theta_m(2n) - \theta_m(2n-1)}{T_{2n} - T_{2n-1}} \tag{26}$$

Which means the estimation of the frequency offset should be updated after two successive synchronization interval.

In the context of the IEEE 1588, bias correction needs to be done for the down-link and up-link separately. One important feature is that this method can be implemented recursively. We define $C_{BC}(t)$ as the bias free version of the slave time. So $C_{BC}(t)$ is equal to the $C(t)$ minus the estimated bias. To see how to implement this method recursively, a summary of this algorithm is given by Algorithm 1.

Algorithm 1 :G-BC 1588

$\frac{f(\alpha)}{\alpha} = 2(1 + \frac{0.56}{\sqrt{\alpha+0.3}})$ *%by knowing an approximation of* α
%eight auxiliary variables to keep the summations
$sum_{DL} \leftarrow 0$, $sum'_{DL} \leftarrow 0$, $sum_{UL} \leftarrow 0$, $sum'_{UL} \leftarrow 0$, $sum^{\sigma}_{UL} \leftarrow 0$, $sum^{\sigma}_{UL} \leftarrow 0$, $sum^{\alpha}_{DL} \leftarrow 0$, $sum^{\alpha}_{UL} \leftarrow 0$
$i_{DL} \leftarrow 1$, $i_{UL} \leftarrow 1$ *%counters*

For $n = 1, 2, ...$

$\delta_{DL}(n) = C(t_n + x_n) - t_n$
$\delta_{UL}(n) = t'_n - C(t'_n - y_n)$
$\theta_m(n) = \frac{\delta_{DL}(n) - \delta_{UL}(n)}{2}$

If n is even

$k = n/2$
$\delta'_{DL}(k) = min(\delta_{DL}(n), \delta_{DL}(n-1))$
$\delta'_{UL}(k) = min(\delta_{UL}(n), \delta_{UL}(n-1))$

$sum_{DL} = sum_{DL} + \delta_{DL}(n) + \delta_{DL}(n-1)$
$sum'_{DL} = sum'_{DL} + \delta'_{DL}(k)$
$sum_{UL} = sum_{UL} + \delta_{UL}(n) + \delta_{UL}(n-1)$
$sum'_{UL} = sum'_{UL} + \delta'_{UL}(k)$
$sum^{\sigma}_{DL} = sum^{\sigma}_{DL} + (\delta_{DL}(n) - \delta_{DL}(n-1))^2$
$sum^{\sigma}_{UL} = sum^{\sigma}_{UL} + (\delta_{UL}(n) - \delta_{UL}(n-1))^2$

$\widehat{\Delta_{DL}} = sum_{DL}/n$
$\widehat{\Delta'_{DL}} = sum'_{DL}/k$
$\widehat{\Delta_{UL}} = sum_{UL}/n$
$\widehat{\Delta'_{UL}} = sum'_{UL}/k$
$\widehat{\sigma^2_{DL}} = sum^{\sigma}_{DL}/2k$
$\widehat{\sigma^2_{UL}} = sum^{\sigma}_{UL}/2k$

%estimate the value of α
$\hat{\alpha}_{DL} = \frac{0.3}{\left(\frac{0.56\widehat{\sigma_{DL}}}{\widehat{\Delta_{DL}} - \widehat{\Delta'_{DL}}}\right)^2 - 1}$
$\hat{\alpha}_{UL} = \frac{0.3}{\left(\frac{0.56\widehat{\sigma_{UP}}}{\widehat{\Delta_{UL}} - \widehat{\Delta'_{UL}}}\right)^2 - 1}$
%the estimated α *should be in the proper neighborhood*
If $\alpha^l_{DL} < \hat{\alpha}_{DL} < \alpha^u_{DL}$
%Start to take the average after a number of iterations
  If $i_{DL} > 10$
    $sum^{\alpha}_{DL} = sum^{\alpha}_{DL} + \hat{\alpha}_{DL}$
    $\bar{\alpha} = sum^{\alpha}_{DL}/(i_{DL} - 10)$
  else
    $\bar{\alpha} = \hat{\alpha}_{DL}$
  end If
  $i_{DL} = i_{DL} + 1$
end If
If $\alpha^l_{UL} < \hat{\alpha}_{UL} < \alpha^u_{UL}$
  If $i_{UL} > 10$
    $sum^{\alpha}_{UL} = sum^{\alpha}_{UL} + \hat{\alpha}_{UL}$
    $\bar{\alpha} = sum^{\alpha}_{UL}/(i_{UL} - 10)$
  else
    $\bar{\alpha} = \hat{\alpha}_{UL}$
  end If
  $i_{UL} = i_{UL} + 1$
end If

%expected value for the down-link delay
$E_{DL} = \frac{\widehat{\Delta_{DL}} - \widehat{\Delta'_{DL}}}{\frac{f(\bar{\alpha})}{2\bar{\alpha}} - 1}$

%expected value for the up-link delay
$E_{UL} = \frac{\widehat{\Delta_{UL}} - \widehat{\Delta'_{UL}}}{\frac{f(\bar{\alpha})}{2\bar{\alpha}} - 1}$

$B = \frac{E_{DL}-E_{UL}}{2}$ *%total bias*

$f_m(k) = \frac{\theta_m(n)-\theta_m(n-1)}{t_n - t_{n-1}}$

%$\theta_m(n)$ *and* $f_m(k)$ *would be passed into the filter*
$[u_\theta(k), u_f(k)] = Kalman(\theta_m(n), f_m(k))$

$C(t) = C(t) - u_\theta(k)$ *%*$C(t)$ *is the slave time*

$f(t) = f(t) - u_f(k)$ *%*$f(t)$ *is the slave frequency*

%The total bias is subtracted from the slave clock
$C_{BC}(t) = C(t) - B$
end If
end For

In many practical problems, we may not be able to know the exact value of the delay parameters, but it is possible to estimate a neighborhood of it. For example in the busy hour, when the traffic is at its maximum, or during the night, when the traffic is at its minimum, we may have different expectations for the traffic load. When any information about the network traffic load is available, the neighborhood that we defined for α in order to prevent those large spikes can also be used as a tool for improving the synchronization accuracy. For example, if we know that the traffic load is more or less than a specific value, then we can adjust a tighter neighborhood for α. In the simulation results section we will see that this knowledge can improve the synchronization accuracy noticeably.

V. Variance Analysis

In order to carry out the variance analysis, in (22) it is assumed that the value of α is known. This simplifying assumption, although unrealistic, allows us to have an insight about the variance of the estimation. To this end, from equation (22) we have:

$$var(\widehat{\alpha\beta}) = E[(\widehat{\alpha\beta} - \alpha\beta)^2] = \frac{var(\widehat{\Delta}) + var(\widehat{\Delta'})}{\left(\frac{f(\alpha)}{2\alpha} - 1\right)^2} \qquad (27)$$

To find the variance of $\widehat{\Delta}$ we have:

$$\begin{aligned}
&var(\widehat{\Delta}) = E[(\widehat{\Delta} - E(\widehat{\Delta}))^2] \\
&= E[\left(\frac{\sum_{i=1}^{n}\delta_i}{n} - (\theta + d + \alpha\beta)\right)^2] \\
&= \frac{1}{n^2}E[\left(\sum_{i=1}^{n}(\delta_i - (\theta + d + \alpha\beta))\right)^2] \\
&= \frac{1}{n^2}E[\sum_{i=1}^{n}\sum_{j=1}^{n}(\delta_i - (\theta + d + \alpha\beta))(\delta_j - (\theta + d + \alpha\beta))] \\
&= \frac{1}{n^2}\sum_{i=1}^{n}E(\delta_i - (\theta + d + \alpha\beta))^2 = \frac{var(\delta)}{n} = \frac{\alpha\beta^2}{n}
\end{aligned}$$

Similarly, it can be shown:

$$var(\widehat{\Delta'}) = \frac{var(\delta')}{k} = \frac{2var(\delta')}{n}$$

where $k = \frac{n}{2}$. In Appendix A, it is shown that:

$$var(\delta') \leq 2\alpha\beta^2(2 + f(\alpha)) - \frac{\beta^2}{2}f(\alpha)^2$$

by substituting the found expressions for $var(\widehat{\Delta})$ and $var(\widehat{\Delta'})$ in (27), the following upper bound is obtained:

$$var(\widehat{\alpha\beta}) \leq \left(\frac{1}{\frac{f(\alpha)}{2\alpha} - 1}\right)^2 \left(\frac{9\alpha + 4\alpha f(\alpha) - f(\alpha)^2}{n}\right)\beta^2 \qquad (28)$$

where $f(\alpha)$ can be approximated using (23).

From [18] it is known that the variance of the Boot-strap method converges to zero with $\frac{1}{n^2}$ whereas (28) indicates that the order of convergence for G-BC 1588 is $\frac{1}{n}$. Consequently it is observed although our method provides an unbiased estimator for the offset, but in terms of the variance of the estimator, Boot-strap method is superior.

VI. Simulation Results

To evaluate our method, PDVs are generated using the results of [20] and [21]. The value of α in a network, differs for different traffic loads. A coarse knowledge about the interval of α is assumed to be available at the slave node. This assumption, as discussed in Section IV, prevents the denominator of (25) to get close to zero. The authors of the Boot-strap method did not provide any algorithm for estimating the frequency offset, so we first assume that the frequency offset is zero and compare the performance of our method with the Boot-strap method. Then we take the frequency offset into the consideration and evaluate the performance of our method in the presence of a frequency offset. Finally, as was described in Section IV, the calculated values for the phase and frequency offsets are passed into a filter. In the following simulations, a Kalman filter is used for this purpose.

It is first assumed that α^l_{DL}=α^l_{UL}=1 and α^u_{DL}=α^u_{UL}=15. From [20] and [21] the value of α for 20 percent and 80 percent traffic loads in a 5 hops network is 2 and 11 respectively. So the interval that is considered for the shaping parameter is loose and is just based on a very approximate knowledge about the range of α.

In Fig. 4 a 20% traffic load from master to slave and 60% traffic load from slave to master is considered. PDVs are generated based on the results of [20] and [21], so the values of α for 20 and 60 percent traffic loads are 2 and 8 respectively. The synchronization interval is 1 sec, the frequency offset is assumed to be zero and the phase offset is set to 1 sec. The time axis starts from t=15 sec.

From the result of the Basic 1588 method it can be seen that the existence of a bias in the estimated phase offset can degrade the synchronization accuracy. It also can be observed that because the PDVs are following the Gamma distribution instead of the exponential distribution, the Boot-strap method is not very accurate. Our method, however, can provide a more

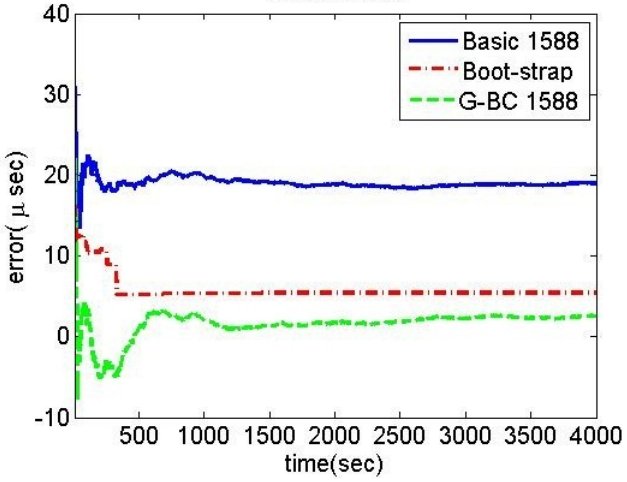

Fig. 4. Error in the estimated phase offset for 20 percent down-link and 60 percent up-link traffic loads.

accurate synchronization. G-BC 1588 estimates the shaping parameters along with the bias. In Fig. 4, the estimated shaping parameters for the down-link and up-link directions are 2.34 and 8.39 respectively. We also observe that although G-BC 1588 has better results compared to the Boot-strap method, it still suffers from a noticeable level of error.

In Fig. 5, the up-link and down-link traffic loads are assumed to be 80 and 20 percents respectively. The value of α for 80 percent traffic load from [21] is 11. By increasing the traffic load, and as a result, increasing the value of α, the PDV becomes less exponential. Consequently, as can be seen in Fig. 5, the Boot-strap method suffers from a larger error. The estimated shaping parameters from G-BC 1588 for the down-link and up-link directions are 2.1 and 9.2 respectively.

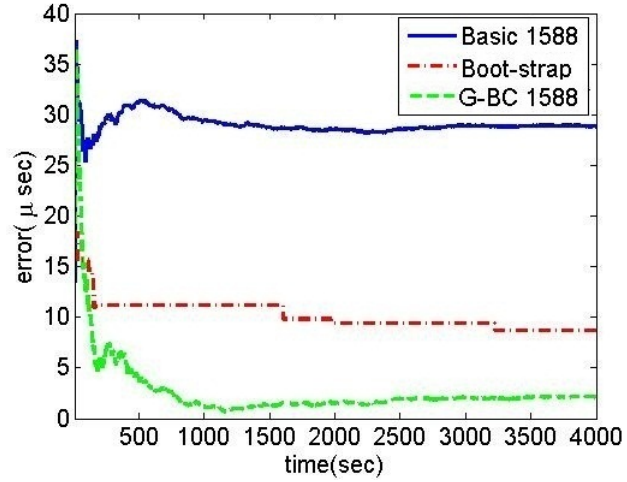

Fig. 5. Error in the estimated phase offset for 20 percent down-link and 80 percent up-link traffic loads.

In both Fig. 4 and Fig. 5 we observe that during the initial iterations G-BC 1588 does not have a good performance. The reason is that this method, in addition to the bias, estimates the variances and the shaping parameters of the down-link and up-link delays. As a result it takes some time/iterations for the method to obtain good estimates for each of these variables.

So far we assumed that the frequency offset is zero. In Fig. 6 we set all the initial settings the same as in Fig. 5, except that the frequency offset of the slave clock is 10^{-6} per second. As it was mentioned before, because the authors of the Boot-strap method have not provided any method for estimating the frequency offset, we just compare our method to Basic 1588.

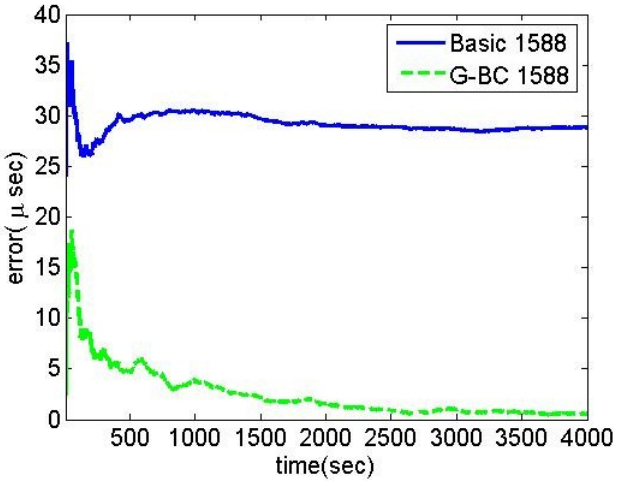

Fig. 6. Error in the estimated phase offset for 20 percent down-link and 80 percent up-link traffic loads, with a frequency offset of 10^{-6}.

Comparing the results in Fig. 6 with the results in Fig. 5, we observe that the existence of a frequency offset does not degrade the results of our method. The estimated shaping parameters for the down-link and up-link delays in Fig. 6, are 3.2 and 11.2 respectively.

As it was mentioned, when information about the network traffic load is available, our method is flexible enough to provide better results. For this purpose we can readjust the values of α^l_{DL}, α^u_{DL}, α^l_{UL} and α^u_{UL} based on the available information. In Fig. 7 we compare the performance of G-BC 1588 with a looser neighborhood for α with G-BC 1588 with a tighter neighborhood. Consider a 20 percent down-link and 60 percent up-link traffic load. For the looser neighborhood, we consider the same neighborhood as in previous figures. So $\alpha^l_{DL} = 1$, $\alpha^u_{DL} = 15$, $\alpha^l_{UL} = 1$ and $\alpha^u_{UL} = 15$. For the tighter neighborhood, assume that the receiver just knows that the down-link traffic load is somewhere between 10 to 40 percent and the up-link traffic load is between 40 and 80 percent. So $\alpha^l_{DL} = 1$, $\alpha^u_{DL} = 6$, $\alpha^l_{UL} = 6$ and $\alpha^u_{UL} = 11$.

The experiment is repeated for 20 times and the average is depicted. The vertical lines show the standard deviation of error around the average values. It can be observed that considering a tighter neighborhood makes the average error closer to zero and also make the standard deviation of error smaller. This flexibility is one of the advantages of our method over the Boot-strap method.

Although the results of [20] and [21] show that the PDVs can be modeled by Gamma distributions, we also examined the robustness of our method against a non-Gamma distributed PDV. A rule of thumb is that even if the PDVs do not follow

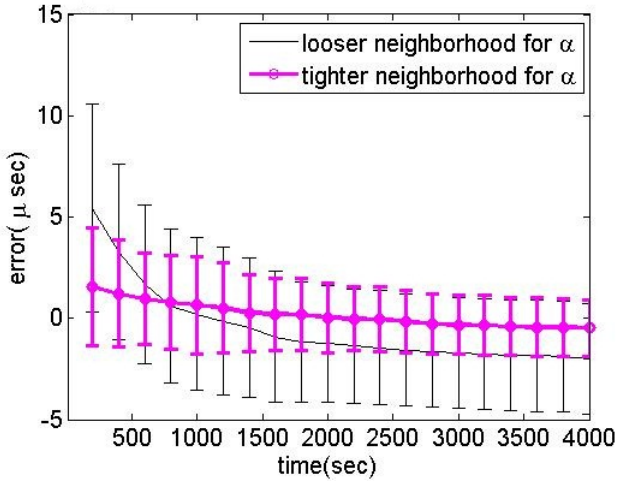

Fig. 7. Error in the estimated phase offset for 20 percent down-link and 60 percent up-link traffic loads, with a frequency offset of 10^{-6}.

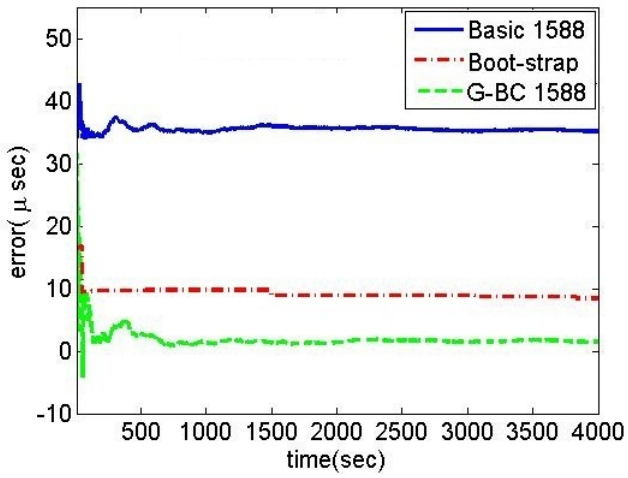

Fig. 8. Error in the estimated phase offset for the Weibull distributed down-link and Gamma distributed up-link delays.

the Gamma distribution, as long as there are some values for the shaping and scale parameters to make the Gamma distribution similar to the non-Gamma distributed PDVs, our method still can provide good results. However, if the PDVs are very different from the Gamma distribution, then our method may have a large amount of error. The reason is that for G-BC 1588 method we also estimate the shaping parameters. In order to estimate the shaping parameters, we first estimate the variance of the delays. When the estimated variances do not follow the Gamma distribution, then the estimated values for the shaping parameters would not be accurate anymore. Estimating the bias using a poor estimation of the shaping parameters will result in a bad estimate of the bias.

Fig. 8 illustrates the result of using the Weibull distribution to generate the PDV for the down-link delay. The shape and scale parameters for Weibull distribution are assumed to be 6.5 and 0.5 respectively. The generated random number is multiplied by 10^{-6} to keep the random delay in the order of micro-seconds. The up-link delay is generated using the results of [21] for 80 percent traffic load. We assumed that $\alpha^l_{DL} = 1$, $\alpha^u_{DL} = 6$, $\alpha^l_{UL} = 6$ and $\alpha^u_{UL} = 11$. As we observe in Fig. 8, although the distribution of the delay for the down-link direction does not follow the Gamma distribution, G-BC 1588 can estimate the bias successfully.

To make the comparison easier, in Fig.9 we compare the performance of G-BC 1588 when the down-link delay is generated using Weibull and uniform distributions. The up-link delay is the same as in Fig. 8. The random numbers from the uniform distribution are generated between 0 and 1, then are multiplied by 10^{-6} to keep the numbers in the order of micro-seconds.

We observe that changing the distribution from Weibull to uniform degrades the result. This was predictable because there is less similarity between the Gamma and uniform distributions compared with the Weibull and Gamma distribution.

So far we have assumed that we have a constant traffic load in the network. Clearly this assumption is not valid and traffic

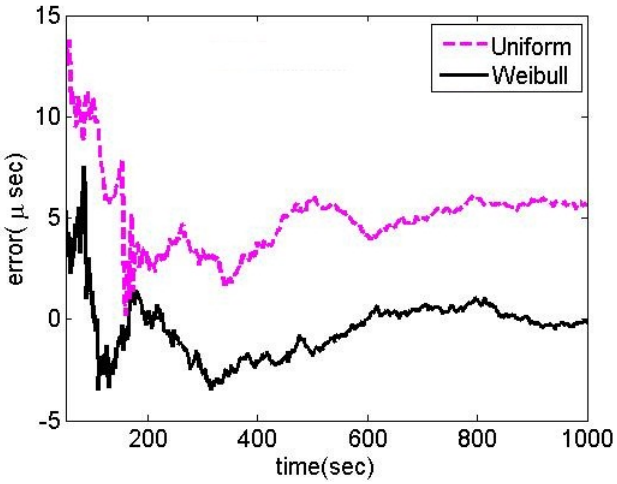

Fig. 9. Comparison between the error in the estimated phase offset for the Weibull and uniform distributed down-link and Gamma distributed up-link delays.

loads and as a result PDVs change with time. When the traffic load changes, the previous estimated values act as a wrong memory. It means that, until after a long time when the effect of the wrong memory fades, we are unable to obtain accurate estimates. One simple solution to this problem is to restart the synchronization process periodically. The process needs to be restarted often enough to catch significant changes in network traffic load. For example, in [25], traffic loads change never faster than every 12 minutes, and sometimes at an even slower frequency. In reality, changing the traffic load does not usually happen very fast, rather it happens gradually. However, in this section, the extreme case has been considered. It is assumed that the traffic load changes instantly and significantly. In Fig. 10 the traffic load from master to slave and from slave to master is 20 percent and 60 percent respectively. At t=500 sec, the up-link traffic load suddenly changes from 60 to 80 percent. Then, at t=3000 the up-link traffic load changes from

80 to 40 percent. The values of α for the 40, 60 and 80 percent traffic loads from [20] and [21] are 6, 8 and 11 respectively. We assumed that $\alpha^l_{DL} = 1$, $\alpha^u_{DL} = 6$, $\alpha^l_{UL} = 6$ and $\alpha^u_{UL} = 11$. The Basic 1588 and G-BC 1588 are restarted with a period of 1000 sec. It means that when the time is a multiple of 1000, we restart all the summations and matrices that are involved in Kalman filtering and bias estimation processes. The red line in Fig. 10 shows the zero line, in order to make the comparison easier.

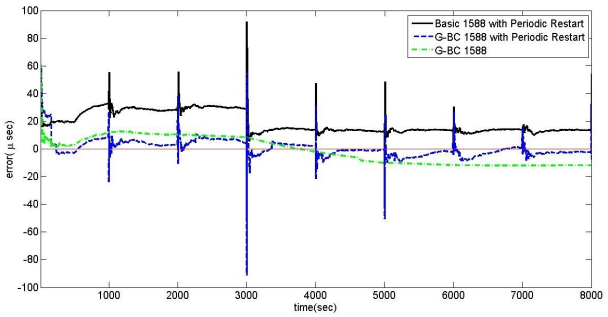

Fig. 10. Error in the estimated phase offset. The down-link traffic load is 20 percent. The up-link traffic load changes from 60 to 80 percent at t=500 sec and changes form 80 to 40 percent at t=3000 sec. The period of restart is 1000

Whenever the process is restarted, previous estimated values are assigned to be zero. As a result when traffic load changes with time, in comparison with the case that the process is not restarted, there is less outdated and potentially misleading memory in the estimation equations. We observe that G-BC 1588 with Periodic Restart, between two successive restart points, has a better performance than Basic 1588. However, a serious problem with this approach are the random spikes that happen, for the first initial iterations, whenever the synchronization process is restarted.

In order to mitigate the effect of random spikes, the following solution can be applied. At the beginning of each periodic restarting of synchronization process, a temporary slave clock sets its time and frequency to the slave clock. Whenever the synchronization process is restarted, the temporary slave clock starts to work and acts as the slave clock for a predetermined interval. Meanwhile, the slave clock updates its estimates. When the interval is finished, the time again is read from the slave clock. In Fig. 11 all the initial settings are the same as in the Fig. 10. The predetermined interval in Fig. 11 is assumed to be 1/10 of the restart period so it is 100 sec.

As we observe in Fig. 11, using this strategy mitigates the effect of random spikes significantly. Consequently, G-BC 1588 with Periodic Restarts, provides good results without severely getting affected by changing traffic loads.

VII. Conclusions and Future Work

In this paper, we introduced G-BC 1588 as a recursive method for estimating and correcting the delay's bias in the synchronization process of the packet-based networks. this method is compatible with the current version of the IEEE 1588 protocol and allows us to correct the slave's clock

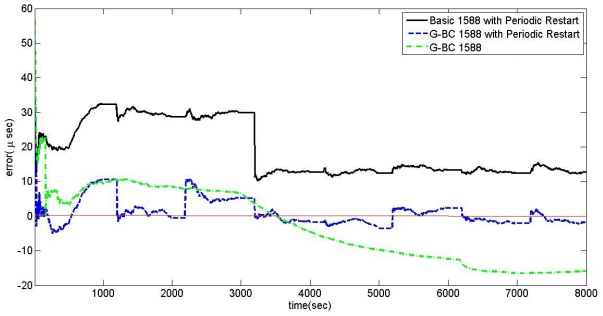

Fig. 11. Error in the estimated phase offset. The down-link traffic load is 20 percent. The up-link traffic load changes from 60 to 80 percent at t=500 sec and changes form 80 to 40 percent at t=3000 sec. The period of restart is 1000. The interval for working the temporary clock is 100 sec

recursively. If any knowledge about the network traffic load is available at the slave clock, G-BC 1588 is flexible enough to use this knowledge in order to improve the synchronization accuracy. This method also allows us to use different filters for processing data. Given the fact that PDV in networks does not follow a Gaussian distribution, trying other filters such as Particle filters may provide more improvement in the synchronization accuracy. Comparison between the performance of different filters can be an interesting subject for the future works.

VIII. Acknowledgments

This work was made possible with supports and funds from Ericsson. Any findings and conclusions in this paper are those of the authors, and not necessarily reflect the views of Ericsson.

Appendix A

From equation (12) it can be observed that the variance for δ' is the same as the variance for the random variable W. From equation (16) we have:

$$E(W^2) = 4\alpha(\alpha+1)\beta^2 - (\frac{\beta^2}{4})\sum_{k=0}^{\infty}\left(\frac{\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}\right)^2 (2\alpha+k)(2\alpha+k+1)$$
$$\approx 4\alpha(\alpha+1)\beta^2 - (\frac{\beta^2}{4})\sum_{k=0}^{\infty}\left(\frac{(2\alpha+k)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}\right)^2$$
$$\leq 4\alpha(\alpha+1)\beta^2 - (\frac{\beta^2}{4})\left(\sum_{k=0}^{\infty}\frac{(2\alpha+k)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}\right)^2$$
$$= 4\alpha(\alpha+1)\beta^2 - (\frac{\beta^2}{4})f(\alpha)^2$$

On the other hand, from (17) we have:

$$E(W) = 2\alpha\beta - \frac{\beta}{2}f(\alpha)$$

as a result, the variance of Δ' is bounded by:

$$Var(\Delta') = \sigma_W^2 = E(W^2) - E^2(W) \leq 2\alpha\beta^2(2 + f(\alpha)) - \frac{\beta^2}{2} f(\alpha)^2 \quad (29)$$

REFERENCES

[1] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std. 1588-2008, 2008.

[2] K. Noh, Q. M. Chaudhari, E. Serpedin, and B. W. Suter, " Novel clock phase offset and skew estimation using two way timing message exchanges for wireless sensor networks," *IEEE. Trans. Commun*., vol. 55, no. 4, pp. 766-777, Apr. 2007.

[3] M. Leng and Y. C. Wu, " On clock synchronization algorithms for wireless sensor networks with unknown delay," *IEEE. Trans. Vehicular technology*., vol. 59, no.1, Jan 2010.

[4] Q. M. Chaudhari, E. Serpedin, and K. Qaraqe, "On maximum likelihood estimation of clock offset and skew in networks with exponential delays," *IEEE Trans. Signal Processing*., Vol. 56, no. 4, pp. 1685-1697, Apr. 2008.

[5] Q. M. Chaudhari, E. Serpedin, and K. Qaraqe, "On Minimum Variance Unbiased Estimation of Clock Offset in a Two-Way Message Exchange Mechanism," *IEEE Trans. Inf. Theory*., Vol. 56, no. 6, pp. 2893-2904, Jun. 2010.

[6] N. M. Freris, V. S. Borkar, and P. R. Kumar, "A model-based approach to clock synchronization," *Proceedings of 48th IEEE Conference on Decision and Control*., pp. 5744?5749, Dec. 2009.

[7] R. Solis, V. S. Borkar, and P. R. Kumar, "A New Distributed Time Synchronization Protocol for Multihop Wireless Networks," *Proceedings of the 45th IEEE Conference on Decision and Control*., pp. 2734?2739, Dec. 2006..

[8] A. Giridhar and P. R. Kumar, "The Spatial Smoothing Method of Clock Synchronization in Wireless Networks," *Theoretical Aspects of Distributed Computing in Sensor Networks*., pp. 227?256, 2011.

[9] B. Friedland, "Treatment of bias in recursive filtering" *IEEE Trans. Automatic Control*., Vol. 14, no. 4, pp. 359-367, Aug. 1969.

[10] M. B. Ignagni, " Separate-bias Kalman estimator with bias state noise," *IEEE Trans. Automatic Control*., Vol. 35, no. 3, pp. 338-341, Mar. 1990.

[11] S. Graham and P. R. Kumar, "Time in general-purpose control systems: The Control Time Protocol and an experimental evaluation," *Proceedings of the 43rd IEEE Conference on Decision and Control*., pp. 4004?4009, Dec. 2004..

[12] D.Veitch ,S.Babu, and A.Pasztor , "Robust synchronization of software clocks across the Internet," *Proceedings of the 4th ACM SIGCOMM Conference on Internet measurement*., pp. 219?232, Oct. 2004.

[13] N. M. Freris, S. R. Graham, and P. R. Kumar, "Fundamental Limits on Synchronizing Clocks over Networks," *IEEE Transactions on Automatic Control*., vol. 56, no. 2, pp. 1352?1364, 2011.

[14] D. T. Bui, A. Dupas and M. L. Pallec, "Packet delay variation management for a better IEEE1588V2 performance" in *Proc. IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS'09)*., Brescia, Italy, Oct. 2009, pp. 75-80.

[15] T. Murakami and Y. Horiuchi, "Improvement of synchronization accuracy in IEEE 1588 using a queuing estimation method" in *Proc. IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS'09)*., Brescia, Italy, Oct. 2009, pp. 12-16.

[16] M. Anyaegbu, Cheng-Xiang Wang, W. Berrie, "A sample-mode packet delay variation filter for IEEE 1588 synchronization" *ITS Telecommunications (ITST), 2012 12th International Conference on*.,pp. 1-6, Nov. 2012.

[17] T. Murakami and Y. Horiuchi, "Improvement of synchronization accuracy in ieee 1588 using a queuing estimation method," *in Proc. International Symposium on Precision Clock Synchronization for Measurement, Control and Communication(ISPCS)*.,2009, pp. 1-5, 2009.

[18] D. R. Jesk, A. Sampath, "Estimation of clock offset using Bootstrap bias-correction techniques" *Technometrics*., Vol. 45, no. 3, pp. 256-261, Aug. 2003.

[19] D. R. Jesk, A. Chakravartty, "Effectiveness of Bootstrap bias correction in the context of clock offset estimator" *Technometrics*., Vol. 48, no. 4, pp. 530-538, Nov. 2006.

[20] I. Hadzic, D. R. Morgan, "On packet selection criteria for clock recovery" *ISPCS 2009*.,pp. 1-6, Oct. 2009.

[21] I. Hadzic, D. R. Morgan, "Adaptive packet selection for clock recovery" *ISPCS 2010*.,pp. 42-47, Oct. 2010.

[22] G. Giorgi, C. Narduzzi, "Performance analysis of Kalman filter-based clock synchronization in IEEE 1588 networks" *IEEE Trans. Instrumentation and Measurement*., Vol. 60, no. 8, pp. 2902 - 2909, Aug. 2011.

[23] A. Papoulis and S. U. Pillai, "Probability, random variables and stochastic processes " *McGraw-Hill*., Forth edition, p.194.

[24] N. M. Temme, "Uniform Asymptotic Expansions of the Incomplete Gamma Functions and the Incomplete Beta Function" *Mathematics of Computation*., Vol. 29, No. 132, pp. 1109-1114, Oct. 1975.

[25] "Timing and synchronization aspects in packet networks," *ITU-T G.8261/Y.1361,*, April 2008.

MohammadJavad Hajikhani received his B.Sc degree from Iran University of Science and Technology, Tehran, Iran, in 2011 and M.Sc degree from Carleton University, Ottawa, Canada, in 2013, both in Electrical Engineering. He is currently a PhD student in the department of Electrical and Computer Engineering at McGill University. His research interests include Energy harvesting and wireless transfer of energy, wireless sensor networks and synchronization.

Thomas Kunz Thomas Kunz is currently a professor in the Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario. He heads the Mobile Computing Group, researching wireless network architectures (manets. wireless mesh networks, and wireless sensor networks), network protocols (routing, mobile IP, and OoS support), and middleware layers for innovative wireless applications. He has served on more than 50 TPCs of international conferences and workshops in the mobile and wireless domain. He is the author or coauthor of more than 180 technical papers and received a number of awards and best paper prizes. He is a senior member of both ACM and IEEE.

Howard Schwartz Professor Howard M. Schwartz received his B.Eng. degree from McGill University, Montreal, Quebec and his M.S. degree and Ph.D. degree from M.I.T., Cambridge, Massachusetts. He is currently a Professor in Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada. His research interests include adaptive and intelligent systems, robotics, system modelling and system identification. His most recent research is in multi agent learning with applications to teams of mobile robots. He is a senior member of IEEE.