Using Physical Layer Clock Recovery to Augment Application Layer Time Synchronization

S. M. Usman Hashmi, Imran Shafi, Jamil Ahmad and Syed Ismail Shah

Abstract—Achieving same notion of time remains an important task for most distributed systems. Time synchronization requires a unique combination of high accuracy (μsec level) and energy efficiency. Several application layer protocols have been developed to meet these requirements. This article propose that the physical layer clock recovery process can provide application layer clock drift estimate and the application layer clock can be corrected with the help of this estimate. This eliminates the need of application layer time synchronization protocol i.e. the cross layer approach reduces the number of message exchanges required by application layer for time synchronization which leads to energy conservation. It argues that such a cross layer approach can provide a more accurate frequency offset estimation, or alternatively can achieve greater energy savings, for a given accuracy, by reducing the message exchanges. Analysis of proposed method provides concrete bounds on achieved improvement. Experimental evaluation showed that physical layer clock drift can be used to correct application layer clock drift as they are identical.

Index Terms—Time synchronization, clock recovery, wireless sensor network, cross layer time synchronization, energy efficient time synchronization, frequency offset estimation, distributed systems

I. Introduction

Distributed systems need a shared notion of causality to enable coordination. In cyber physical and sensor networks, due to interaction with physical processes governed by time dependent laws, the causality notion is strengthened to establish a synchronized notion of time which can have microsecond precision requirements [18], [19], [20]. Thus, systems can now localize intruders, determine projectile trajectories and even establish accurate movements [14].

Establishing this shared notion of time remains a challenge. The notion of time in any computing device is kept by using an oscillator that produces a periodic signal at *known* frequency. The signal is then used to increment a hardware counter, with time derived as a simple multiplicative factor of the known frequency [28]. This approach of keeping time has two fundamental limitations.

First, with systems initialized at different time, the global value of time will be different. This absolute difference between system clocks is called the *phase offset*. The second limitation arises from the variation in the "known" frequency, due to environmental and manufacturing differences, of the oscillator and is known as the *frequency offset*. Thus even when different clocks calculate and compensate for the phase offset, a different notion of the time increment causes the clocks to drift apart over time [14].

Researchers have found several ways in which to compensate for both these limitations [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [18], [19], [20], [21]. Phase offset is calculated with message exchanges, with several novel mechanisms taking care of the uncertainties. Frequency offset can be eliminated by using stable but expensive oscillator. A more traditional, and low cost, approach is to estimate the frequency offset using regression techniques on the results of several message exchanges. A significant amount of research in frequency estimation has gone into making it more accurate while using lesser number of message exchanges, thus conserving energy.

Our key insight is the observation that physical layer symbol recovery already compensates for frequency offset using feedback and feed forward time synchronization system [11], [16], [17], [29], [31], [32], [33]. Feedback systems adjust the synchronization parameters using the error signal in a recursive manner and can operate at symbol rate and higher then symbol rate. Error feedback system uses different types of interpolations, timing error detectors, loop filters and interpolation controllers to achieve physical layer time synchronization. The clock recovery process includes in itself information to estimate the frequency offset.

This article argue here that not only can this approach estimate the frequency offset accurately, it can use this estimate to significantly improve the application layer frequency offset calculation. The contribution of this paper are, thus, threefold: (i) It present the key idea that physical layer clock recovery can be used to estimate frequency offset (Section III) (ii) It experimentally evaluate the estimation using physical layer signal and compare it with application layer mechanism (Section IV) (iii) It provide energy efficiency of the model and theoretical bounds that shows the minimum possible variance of the timing frequency error (Section V).

II. Related Work

Time synchronization is a very important aspect of wireless distributed networks. WSN make use of time synchronization for sensor data fusion or any coordination in the network. Clock precision and accuracy varies with requirements of different WSN. Critical metrics of time synchronization in any

WSN can be precision, efficiency, lifetime, cost, scope and availability. Precision of a clock refers to the maximum error that a clock can have in a network with respect to the master clock. Efficiency corresponds to the time, energy and computations required to perform the synchronization. Energy efficiency is a major requirement and one way to concur is to have a synchronization protocol with very few computations. The simpler the protocol, higher the energy efficiency and longer the battery life. Energy efficiency can also be increased by decreasing the number of data transmissions required for a node to work properly in the network. Time synchronization in less number of transmission also decreases the energy requirements. Lifetime of the synchronization refers that how often time synchronization should be done. Increasing lifetime of synchronization improves energy efficiency but effects the precision and accuracy of the synchronizing clocks. Cost is another major factor, as there is a requirement of very large number of devices/sensors in a network. Technology advancements are improving cost factor. Scope and availability corresponds to the coverage issues.

Every network, whether it is wired or wireless, needs clock synchronization. There are different protocols available for the wired and wireless network synchronization at application layer. Network Time Protocol (NTP) is most widely used in wired networks for time synchronization [18] among different protocols. NTP creates a hierarchical tree of time servers. Primary server synchronizes with the global clock and distributes the time information to the secondary servers and then to the clients. Secondary servers are also called backup servers, as they act as backup for the primary servers. NTP tries to ensure an extremely accurate clock and that's why primary servers usually equipped with atomic clock. Precision Time Protocol (PTP) uses a master-slave architecture for the distribution of clock among the nodes of wired network. The master node provides the time information to the slave nodes. Master node is synchronized with grand master node attached to time reference such as Global Positioning System (GPS). GPS is a satellite navigation system that provides time and location information and can be used as a time reference.

Wired network time synchronization protocols do not offer very good results in wireless sensor networks because of the different requirements i.e., energy efficiency, infrastructure, end to end latency and reliability [19] etc. Using GPS in every wireless node to maintain same time in the network is also not favorable option, as it makes the inexpensive wireless nodes more expensive and energy consuming. GPS communication also requires line of sight with the satellite which may not be available, depending of deployment environment. Time synchronization in wireless networks can be done in three different ways. Relative timing seems to be the simplest one, as it only relies on the order of the events and do not maintain an actual clock, but this can only be used in limited types of wireless networks. Another way of synchronizing is that each node has the information about its frequency and phase offset with other nodes and can synchronize their clock values. This approach is mostly deployed by many of the time synchronization protocols. Global synchronization, another method of synchronization, tries to maintain a global clock throughout the network. This synchronization approach is very rarely used.

Many protocols are available for the time synchronization at the application layer for any wireless network. Reference Broadcast Synchronization (RBS) protocol can be used in wireless distributed network for time synchronization that pursues to decrease non-deterministic latency using receiver-to receiver synchronization [20], [19], [27]. Wireless node sends reference broadcast beacons to its neighbors using physical layer broadcasts and compute the non-determinism of packet send time, access time and propagation time, depending only on the packet receive time. This reference broadcast packet can be used to synchronize a set of receivers with one another [20]. RBS maintains a table that contains the local clock values of each node in the network. RBS relates the local clocks of node with each other using table and let the clocks run without correction.

In [28] the Romer's protocol uses an innovative time transformation algorithm for achieving clock synchronization. It uses message delay, which is estimated by the lower bound and the upper bound round trip time to compare the utmost difference between two communicating nodes and consequently synchronize them [20]. Romer's protocol also uses the same principle as RBS to let the clocks run untethered. Continuous clock synchronization protocols spread the correction of clock over a finite interval. The local clock time is corrected by gradually speeding up or slowing down the clock rate [19]. It provides the minimal message complexity, fault tolerance [14], [15], [19] and avoids unpredictable instantaneous corrections of clock values.

Timing-sync Protocol for Sensor Networks (TPSN) is a sender-receiver based synchronization protocol designed for WSN. TPSN creates tree of the nodes and synchronize them in two phases namely, level discovery phase and synchronization phase. In level discovery phase a tree of node is created by assigning a level to each node. Level discovery phase also defines the master node or root node. In synchronization phase all nodes synchronize with its upper level nodes and eventually synchronize to root node.

Flooding Time Synchronization Protocol (FTSP) [21] seems to be a most promising application layer protocol used for time synchronization. FTSP provides good bandwidth efficiency and robust to wireless node failures. FTSP transmits it time stamp periodically to all of the nodes in the wireless network and these nodes correct their application layer clock by comparing the time stamps. Using FTSP, a master node can synchronize

every node in its range by using a single radio message time-stamped at both the master and the receiving nodes.

Protocols discussed by [20] and [19] are network wide time synchronization protocols that can hold large node density. Delay Measurement Time Synchronization which is an energy efficient protocol but less accurate than the RBS [27] protocol. Probabilistic Clock Synchronization extends RBS by providing probabilistic bounds on the accuracy of clock synchronization. Time-Diffusion Protocol (TDP) based on a diffusion of messages involving all the nodes in the synchronization process. All of the above protocols [18], [27], [28], [12], [13] can be used for application layer clock synchronization in any wireless distributed network (e.g. Wireless Sensor Network) [20], [19] depending on the environment and requirements because each has its own merits and demerits.

Time synchronization at physical layer does offer a very broad literature. It refers to symbol timing recovery [32], [33], [31] and extracting the phase and frequency of transmitter's timing clock at the receiver end [32], [33], [31], [30]. Timing synchronization can be done using feedforward or feedback systems [31]. Feedback systems adjust the synchronization parameters using the error signal in a recursive manner and can operate at higher then symbol rate as well as symbol rate [30]. Error feedback system uses different types of interpolations, timing error detectors, loop filters and interpolation controllers to achieve different synchronization schemes.

Interpolation required to compute interpolant from given number of symbols, which is the output of matched filter, can be done in various ways where the polynomial interpolation and polyphase filterbank interpolation are common. Special cases of polynomial interpolation are linear, quadratic and cubic interpolation which can also be computed using farrow structure [32], [33], [31]. Linear interpolation requires two samples to compute the interpolant whereas quadratic and cubic interpolation requires three and four samples respectively. Increase in number of samples used for interpolation increases accuracy but makes the timing recovery system computationally expensive and complex. Farrow structure used for interpolation offers less computations and hence feasible from implementation point of view.

Reference [32], [33] and [31] describe different types of timing error detectors (TED) that can be used to produce error signal. Maximum likelihood timing error detector (ML-TED) uses the slope of the symbols and apply sign correction to get the error signal. Zero crossing timing error detector (ZC-TED) only operates at two samples per symbol [32], [33]. Gardner timing error detector (G-TED) is non-data-aided version of ZC-TED.

Phase lock loops (PLL) tracks the phase and frequency error and its parameters can be controlled to adjust the acquisition time and tracking performance. PLL is the main component of many synchronization systems. PLLs are differentiated on the order of the filter used in the design. Proportional plus integrator loop filter can be used to track phase and frequency error.

Reference [32], [33], [31] discusses some interpolation controls that can be used with the feedback synchronization systems. Modulo-1 counter interpolation control is used where interpolants are required after every fixed number of samples. Interpolation control can also be recursive.

Cross layer approach presented in this paper saves the computation and communication required for the application layer timing offset. It computes the physical layer timing frequency offset and apply it to application layer which saves energy required to transfer time stamps. Next section discusses the details of the proposed model.

III. SYSTEM MODEL

Application layer clocks are usually timers that counts the oscillations of crystal and maintains two registers to define that how many oscillations of crystal is equal to one application layer clock tick [19]. Clock used for symbol timing synchronization at physical layer also counts the oscillations of the same quartz crystal. Both the clocks at the physical layer and application layer are derived from the oscillation of quartz crystal, as the devices with one hardware oscillator implements every clock within that device as

$$C(t) = k\int_{t_0}^{t}\omega(t)dt + C(t_0) \qquad (1)$$

Where t is real-time, $\omega(t)$ is angular frequency of oscillator and k is the constant of proportionality [28] and this fact is exploited to perform the cross layer synchronization.

Time synchronization at physical layer is discussed in detail by [32], [33], [31] and [30]. In [31] many symbol timing synchronization systems are discussed with different modulation schemes, interpolators, timing error detectors, loop filters and interpolation control. One such symbol timing synchronization system is shown in Fig. 1.

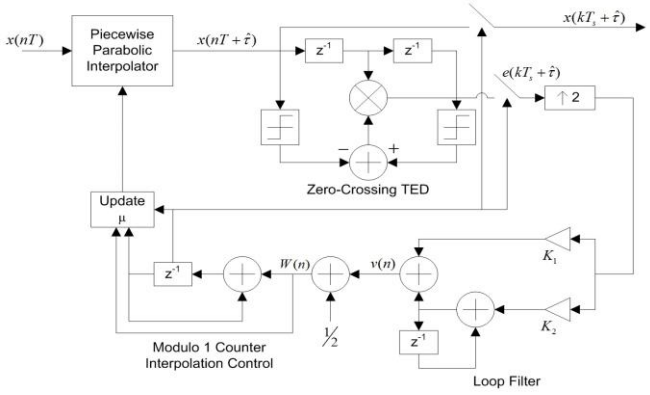

Fig. 1. Symbol timing synchronization system for physical layer

This synchronization system, shown in above figure, is for binary pulse amplitude modulation (PAM) and based on zero-crossing timing error detector (ZCTED) using a piecewise parabolic interpolator, proportional plus integrator loop filter and modulo-1 counter interpolation control. This synchronization system is for physical layer clock recovery process. It is designed to track and compensate for the phase and frequency error of the clock. The output of the matched filter $x(nT)$ is fed to piecewise parabolic interpolator for kth interpolant which is defined as

$$\begin{aligned} x\Big((m(k)+\mu(k))T\Big) &= \left(\frac{\mu(k)^3}{6}-\frac{\mu(k)}{6}\right)x\big((m(k)+2)T\big) \\ &-\left(\frac{\mu(k)^3}{2}-\frac{\mu(k)^2}{2}-\mu(k)\right)x\big((m(k)+1)T\big) \\ &+\left(\frac{\mu(k)^3}{2}-\mu(k)^2-\frac{\mu(k)}{2}+1\right)x(m(k)T) \\ &-\left(\frac{\mu(k)^3}{6}-\frac{\mu(k)^2}{2}+\frac{\mu(k)}{3}\right)x\big((m(k)-1)T\big) \end{aligned} \quad (2)$$

Where $m(k)$ is basepoint index and $\mu(k)$ is the fractional change and these two values are computed by modulo 1 counter interpolation control and fed to interpolator.

The output of the interpolator is processed by ZCTED operating at 2 samples/symbol which tries to find the zero crossing in the eye diagram [32], [33], [31] and give zero error when perfectly aligned. The timing error signal is given by

$$e(k) = x\Big(\big(k-{}^1\!/_2\big)T_s+\hat{\tau}\Big)\,[a(k-1)-a(k)] \quad (3)$$

Where $a(k)$ and $a(k-1)$ are symbol decisions for binary PAM

$$a(k-1) = sgn\big\{x\big((k-1)Ts+\hat{\tau}\big)\big\} \quad (4)$$
$$a(k) = sgn\{x(kTs+\hat{\tau})\} \quad (5)$$

The output of the loop filter, with constants K1 and K2, is fed to modulo 1 interpolation control to compute the estimate of fractional change and basepoint index. K1 and K2 can be computed using Kp and K0 (loop gains), Bn (noise bandwidth), Ts (symbol time), T (sample time),$N=\frac{T_s}{T}$ and loop parameter ξ [32], [33], [31]. Tracking performance and acquisition time of the synchronization system depends on the above parameters of loop filter. The loop parameters K_1, K_2, K_p, K_0 and ξ are related to noise bandwidth B_n as $K_pK_0K_1=\frac{4\xi B_n}{\xi+\frac{1}{4\xi}}$ and $K_pK_0K_2=\frac{4B_n^2}{\left(\xi+\frac{1}{4\xi}\right)^2}$ .The acquisition time T_{LOCK} of the PLL depends on the time required to track phase T_{PL} and frequency offset T_{FL} as $T_{LOCK}\approx T_{FL}+T_{PL}$. Now T_{FL} and T_{PL} are directly related to the noise bandwidth and that's why the loop parameters will have an impact over the acquisition time that is required to track the phase $\left(T_{PL}\approx\frac{1.3}{B_n}\right)$ and frequency offset$\left(T_{FL}\approx 4\frac{(\Delta f)^2}{B_n^3}\right)$. Where Δf is the frequency offset.

Modulo-1 counter interpolation control uses the output of loop filter ν(n) to find the basepoint index m(k) and fractional interval μ(k). The fractional interval is computed using the modulo-1 counter η values and the following equation.

$$\eta(n+1) = (\eta(n) - W(n)))mod1 \quad (6)$$

Where$W(n)=\frac{1}{N}+\nu(n)$. Whenever the value of η underflows then n = $m(k)$ Now to find the fractional interval $\mu(k)$ for the computed basepoint index $m(k)$ we have

$$\mu\big(m(k)\big)=\frac{\eta(m(k))}{W(m(k))} \quad (7)$$

$\mu(k)$ and $m(k)$ is now used to compute the next interpolant. Using the above iterative synchronization system, the variations of fractional change μ can be used to compute the actual frequency offset. Different methods for computation of frequency offset are illustrated by [32], [33], [31], [30]. The frequency offset is computed here using the slope of the fractional change and new sampling rate can be given by the equation below to achieve time synchronization at the physical layer.

$$f_s = (2+m)f_d \quad (8)$$

Where f_s is new sampling rate, f_d is symbol rate and m is the slope of the fractional change.

Time synchronization at application layer can simply be modeled as transmission of master node time stamp to other nodes and correction of phase and frequency when using FTSP. FTSP is a very widely used time synchronization protocol at application layer because of the benefits offered by it. Master node wraps its time stamp in a message and transmit this message to all receivers in the distributed network. This message instruct the receivers to note their clock values and compare it with the time stamp received to perform synchronization.

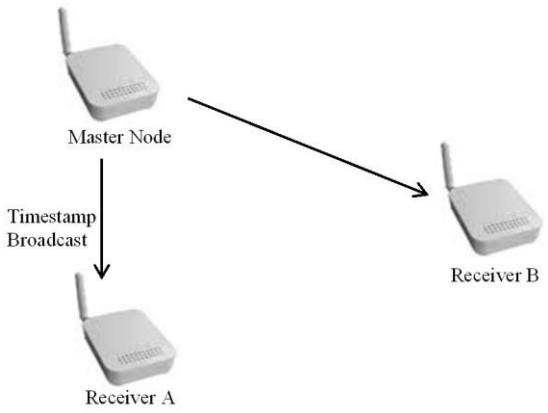

Fig. 2. Timing synchronization at application layer.

Fig. 2 shows time synchronization at application layer using FTSP protocol in a network of two receivers. Master node broadcasts its time stamp to both receivers. On the reception of time stamps, these receivers compare their clock values to the received one and hence synchronized. Synchronization accuracy can be increased by increasing number of time stamp broadcasts. Increasing broadcasts however reduces the energy efficiency of the synchronization protocol.

Single time stamp broadcast can only help for the phase correction. Multiple time stamp broadcasts are required to get frequency synchronization. Frequency error can be calculated by finding the slope of the line created by the time stamps. A simple method to find the slope or frequency error f_e is

$$f_e = \frac{y_n - y_1}{x_n - x_1} \tag{9}$$

Where y_n and y_1 are the nth and $1st$ timestamps of the receiver A, x_n and x_1 are the nth and $1st$ timestamps of the receiver B. Least Square (LS) estimate can give a better estimate of frequency error. LS estimate can be applied to the timestamps exchanged to get the estimate of straight line. Unlike the first method, LS uses all the timestamps to compute the slope and hence more accurate. The slope of the line is frequency offset at the application layer. Timestamps are used as data points in LS estimate and error is minimized between data points and the straight line, given by

$$e = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{10}$$

Where y_i are data points and $\hat{y}_i$ are points of computed straight line given by $y = mx+c$ and to find m and c the least square estimate of a straight line is

$$\begin{bmatrix} m \\ c \end{bmatrix} = (X^T X)^{-1} X^T Y \tag{11}$$

Where X and Y corresponds to the timestamps of receiver A and receiver B.

Time synchronization has to be done at physical layer and application layer of any distributed wireless networks. Whenever the time needs to be synchronized at two different nodes, it starts from the synchronization at physical layer and then synchronize at the application layer. The model proposed in this paper, shown in Fig. 3, enlightens that the heart of both layer's clock is quartz crystal. Once the synchronization at the physical layer is achieved, frequency offset of physical layer clock can be applied to the application layer clock which saves energy required to synchronize at the application layer.

The nodes using the cross layer time synchronization shown in Fig. 3 is assumed to be synchronized in phase or phase synchronization can be achieved by using any application layer protocol like FTSP once or a simple cross layer packet including one timestamp can also provide the time phase synchronization.

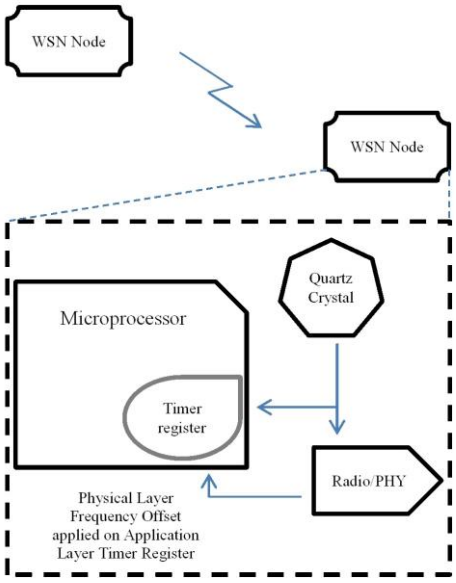

Fig. 3. Cross-Layer Time synchronization

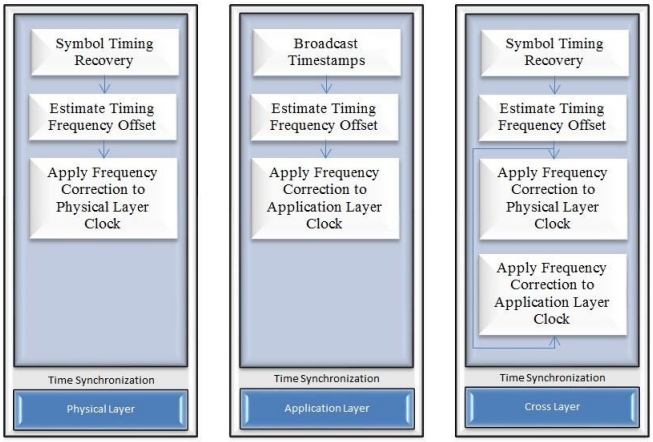

Fig. 4. Flowcharts of time synchronization at different layers

To elaborate the cross layer approach, flowcharts are shown in Fig. 4. At physical layer, symbol timing recovery is applied, frequency offset is estimated and applied to physical layer clock. At application layer timestamps are broadcasted, frequency offset is estimated and applied to application layer clock. In cross layer design, symbol timing recovery is applied, frequency offset is estimated and applied to physical layer clock as well as application layer clock.

IV. EXPERIMENTATION

Experimentation setup for the proposed model is done using two TMS320C6713 DSP Starter Kits (DSKs) [35]. The experimentation consists of two synchronization systems, one

at the physical layer and one at the application layer. The frequency offset computed at both layers comes out to be same which shows that the physical layer frequency offset can be used to adjust the application layer clock. Keeping this in mind, an experimental setup is created that uses two DSKs connected together as shown in Fig. 5.

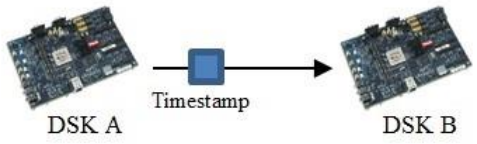

Fig. 5. Experimental setup

DSK A acts as a transmitter and DSK B acts as a receiver. Both DSKs needs to be synchronized in time. DSK A uses binary Pulse Amplitude Modulation (PAM) with symbols $+1$ and -1. Symbols are generated with a symbol rate of 4000 symbols/sec. The sampling rate is set to be 16000 samples/sec. The samples per symbol are computed to be four and total samples are 22000. Binary PAM symbols are up-sampled by four and pulse shaped using a square root raise cosine with fifty percent excess bandwidth and transmitted over the channel to DSK B. DSK B initialize its processing by match filtering the data. Matched filter uses the square root raise cosine with fifty percent excess bandwidth. Symbols are down sampled to two samples/symbol and fed to the system shown in Fig. 1 for time synchronization at the physical layer. Time synchronization at DSK B involves piecewise parabolic interpolator, zero crossing timing error detector, proportional plus integrator loop filter and modulo 1 decrement counter interpolation control. The parameters that are needed for the loop filter for acquisition and tracking efficiency are $B_n T_s = 0.005, \xi = \frac{1}{\sqrt{2}}, K_p = 2.7, K_0 = -1, N = 2$. A graph of the fractional interval is obtained and shown in Fig. 6. The slope of the fractional interval increases with time showing the positive frequency offset. So there exist frequency offset between the clocks of DSK A and DSK B.

The slope of the fractional interval is used to compute the frequency offset. LS estimate is used to find the slope of the fractional interval. This estimate improves by increasing the number of symbols transmitted as shown in the Fig. 7. LS is applied to the fractional interval graph using different number of transmitted symbols. The result shows that estimate of frequency offset improves in accuracy by increasing the number of transmitted symbols. The physical layer offset over 22000 symbols is found to 1.3139 ppm.

Clock jitter of DSK B is found by taking the histogram of the variations in clock and comes out to be Gaussian as shown in Fig. 8. These clock variation are also visible in Fig. 6 and hence this fractional interval results can be used to visualize the clock jitter.

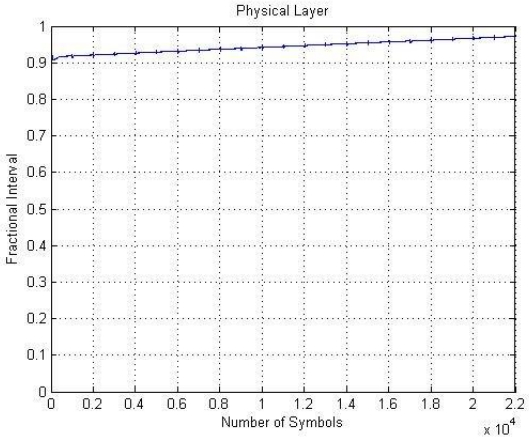

Fig. 6. Timing error and fractional interval over 22000 symbols

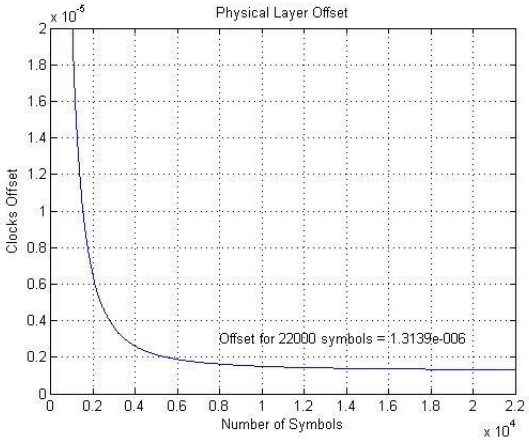

Fig. 7. Physical Layer frequency offset computed for various number of symbols transmitted

DSK A transmits its time stamps to DSK B. On the reception of time stamp message, DSK B also note down its clock value. A total of twenty two timestamps are transmitted from DSK A to DSK B. These timestamps are used to compute the frequency offset at the application layer. Application layer frequency offset is estimated using LS estimate on the timestamps. Fig. 9 corresponds to the application layer clocks of DSK A and DSK B running untethered.

Clock 1 is the clock of DSK A and clock 2 is the clock of DSK B. LS estimate is used to estimate the frequency offset of the application layer clocks. The slope of the line shown in Fig. 9 corresponds to the frequency offset of the application layer clocks running on DSK A and DSK B. Fig. 10 shows the application layer frequency offset between the clocks of DSK A and DSK B computed using different number of timestamps.

The frequency offset is computed here using two to twenty two number of timestamps and as shown, increasing number of time stamp increases the accuracy of the computed frequency offset.

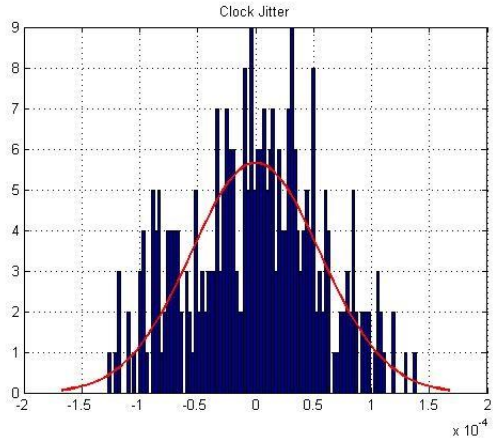

Fig. 8. Clock jitter

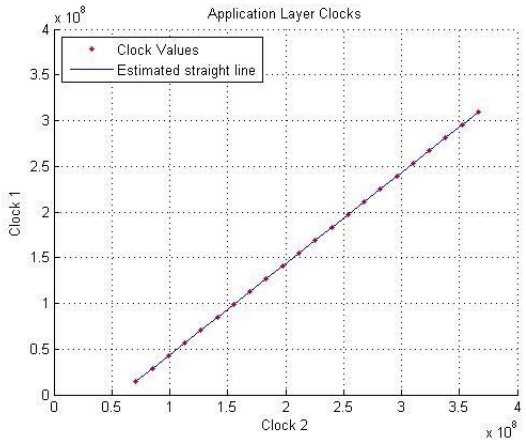

Fig. 9. Application Layer frequency clocks of DSK A and DSK B estimated using LS

The application layer clock offset comes out to be 1.4557 ppm for twenty two timestamps which is nearly same as the physical layer offset that is 1.3139 ppm.

Fig. 11 shows a comparison of the physical layer frequency offset and application layer frequency offset. The frequency offset at the application layer and at physical layers converges to nearly same value. If accuracy needs to be improved then more number of timestamps are required to converge the frequency offset to the physical layer frequency offset. This comparison shows that the frequency offset at both the layers of nodes is same as expected and the physical layer frequency offset can be applied to application layer clocks for the correction of clock frequency. This approach saves the amount of energy required to distribute timestamps and compute frequency offset at the application layer, as physical layer frequency offset can be applied at application layer.

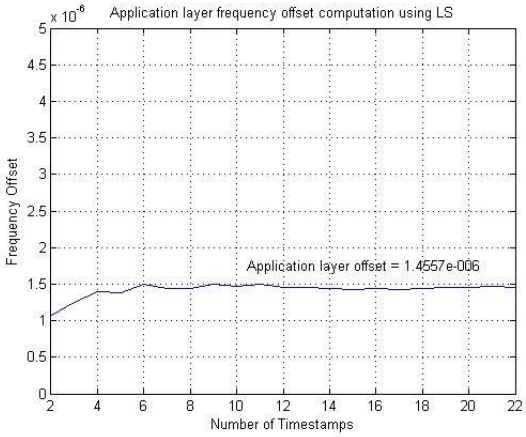

Fig. 10. Application Layer frequency offset computed using various number of timestamps

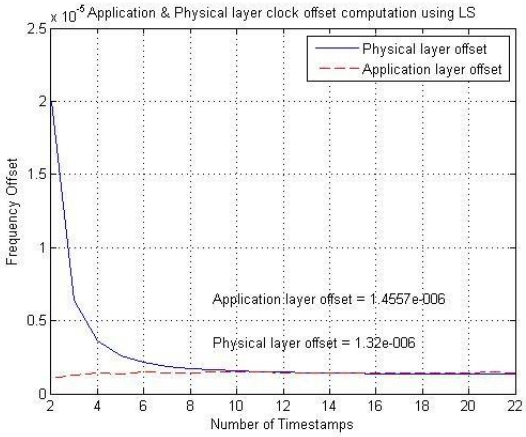

Fig. 11. Comparison of Application Layer and Physical Layer frequency offset

V. MATHEMATICAL ANALYSIS

Cross-layer design helps to make any distributed system energy efficient. In this section energy efficiency of the proposed cross layer design is analyzed.

Energy at the application layer is the energy required to transmit timestamps. A timestamp transmission corresponds to one packet transmission and the energy required to transmit one packet is the same amount of energy required to transmit one timestamp. Keeping this in mind, energy at the application layer can be written as,

$$E_{APP} = E_P \times N_P \quad (12)$$

Where E_{APP} is the energy at application layer, E_P is the energy required to transmit one packet and N_P is the number of packets or timestamps required to achieve frequency error correction at application layer clock. Equation 12 shows that increasing number of packets will increase the energy requirements at the application layer whereas the accuracy of frequency offset computation increases. Energy required per packet can defined in terms of total bits in a packet and total bits in a sample as,

$$E_P = \frac{N_b}{N_m} \times E_S \tag{13}$$

Where N_b are the bits in one packet or time stamp, N_m are the bits in one sample and E_S is the energy required to transmit one sample. N_b holds a direct relation to E_P which means that increasing bits in a packet will increase the energy required per packet. N_m holds inverse relation to E_P reveals that decrease in number of bits per sample will increase energy requirement. Using equation 12 and 13, application layer energy can be written as,

$$E_{APP} = \frac{N_P N_b}{N_m} \times E_S \tag{14}$$

Above equation shows that application layer energy will increase with the increase in energy required to transmit one sample, number of packets and number of bits per packet.

Energy at the physical layer can be defined using energy required to transmit one symbol and total number of symbols need to be transmitted,

$$E_{PHY} = E_{sym} \times N_{sym} \tag{15}$$

Where E_{PHY} is the energy at physical layer, E_{sym} is the energy required to transmit one symbol and N_{sym} are the total symbols. Increase in symbol energy or symbols will increase the energy requirement at physical layer. E_{sym} can be defined as,

$$E_{sym} = \frac{N_{bsym} N_b}{N_m} \times E_S \tag{16}$$

Where N_{bsym} are the number of bits in one symbol. Analyzing above equation yields that energy per symbol also have a direct relation with sample energy and inverse relation with bits in one sample. Whereas increase in bits per symbol increases energy required to transmit one symbol. The above equation can be utilized to compute the energy required to transmit one symbol when bits in one symbol, bits in one packet, bits in one sample and energy required to transmit one sample are known. Using equation 15 and 16, energy at the physical layer can be written as,

$$E_{PHY} = \frac{N_{sym} N_{bsym}}{N_m} \times E_S \tag{17}$$

Above equation shows that physical layer energy will increase with the increase in energy required to transmit one sample, number of symbols and number of bits per symbol. Using this equation, total physical layer energy required to transmit the complete packet can be computed.

Energy efficiency of the proposed cross layer design can be defined as a ratio of energies at the application layer and physical layer,

$$E_{eff} = \frac{E_{APP}}{E_{PHY}} \tag{18}$$

Where E_{eff} is the cross layer energy efficiency that provides the relation of application layer energy to physical layer energy. Using equation 14 and 17 in 18 gives,

$$E_{eff} = \frac{\frac{N_P N_b}{N_m} \times E_S}{\frac{N_{sym} N_{bsym}}{N_m} \times E_S} \tag{19}$$

or

$$E_{eff} = \frac{N_P N_b}{N_{sym} N_{bsym}} \tag{20}$$

Equation 20 shows that energy efficiency will increase if number of packets increases. Similarly decreasing number of symbols also ensure increase in energy efficiency. This equation has the significance importance in computing the cross layer energy efficiency.

Cross layer design ensures usage of one packet to synchronize at physical layer as well as application layer. The packet must have enough symbols so that the receiver can lock the timing frequency offset. This packet also have a time stamp of the sender so that receiver can also synchronize in timing phase. The number of symbols in this packet can be found by exploiting the fact that application layer clock jitter and physical layer clock jitter is same because both have same hardware clock. The Cramer-Rao Lower Bound (CRLB) is the minimum possible variance of the timing frequency error at the physical layer which is given by 21 for the PAM systems [31].

$$\frac{1}{T^2} CRLB(\tau) = \frac{1}{8\pi^2 \xi} \frac{1}{\frac{E_S}{N_0}} \tag{21}$$

Where T is the symbol time, $CRLB(\tau)$ is the variance when timing frequency error τ is estimated, ξ is the loop parameter and E_S is the symbol energy. If R is the symbol rate then equation 21 can be written as

$$CRLB(\tau) = \frac{1}{8\pi^2\,\xi R^2}\frac{1}{\frac{E_S}{N_0}} \tag{22}$$

To estimate the timing frequency offset at the application layer, LS approach can be applied to N time stamps $t[n]$. It gives the minimum LS error which is variance [34]

$$var(t) = \sum_{n=0}^{N-1}(t[n]-\bar{t})^2 - \frac{\left(\sum_{n=0}^{N-1} nt[n] - \frac{N}{2}(N-1)\bar{t}\right)^2}{\frac{N(N^2-1)}{12}} \tag{23}$$

As the cross layer design suggest that variance of clocks at both layers is same, hence equating 22 and 23 gives

$$\frac{1}{8\pi^2\,\xi R^2}\frac{1}{\frac{E_S}{N_0}} = \sum_{n=0}^{N-1}(t[n]-\bar{t})^2 - \frac{\left(\sum_{n=0}^{N-1} nt[n] - \frac{N}{2}(N-1)\bar{t}\right)^2}{\frac{N(N^2-1)}{12}} \tag{24}$$

The above equation can be written in term of symbol rate as

$$R = \frac{1}{\pi}\sqrt{\frac{N(N^2-1)}{\frac{2}{3}\xi\frac{E_S}{N_0}A}} \tag{25}$$

Where,

$A = \frac{N(N^2-1)}{12}\sum_{n=0}^{N-1}(t[n]-\bar{t})^2 - \left(\sum_{n=0}^{N-1} nt[n] - \frac{N}{2}(N-1)\bar{t}\right)^2$

Equation 25 shows that the synchronization accuracy given by N timestamps at the application layer can be achieved using cross layer design with symbol rate R. The required symbol rate decreases by increasing symbol to noise energy $\frac{E_S}{N_0}$ and number of time stamps N as shown in Fig. 12.

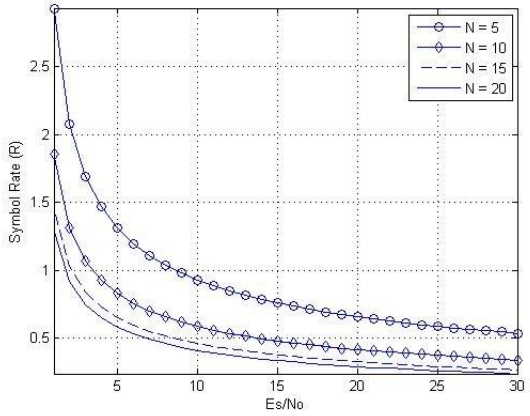

Fig. 12. Required symbol rate for cross layer approach

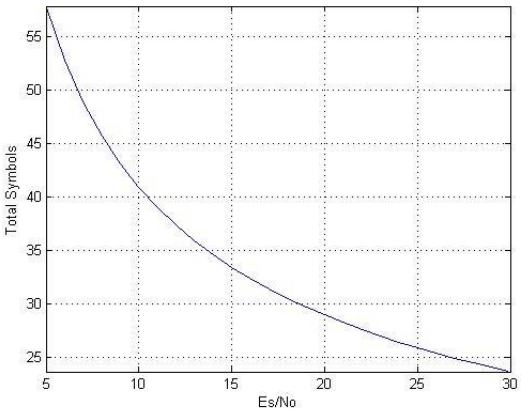

Fig. 13. Required symbols for cross layer packet

Now if total transmission time is T_T, then number of symbols N_{SYM} in one cross layer packet can be computed as,

$$N_{SYM} = R \times T_T \tag{26}$$

Using equation 25 and 26 we have

$$N_{SYM} = \frac{T_T}{\pi}\sqrt{\frac{N(N^2-1)}{\frac{2}{3}\xi\frac{E_S}{N_0}A}} \tag{27}$$

Number of symbols required in cross layer packet decreases by increasing the $\frac{E_S}{N_0}$ which is shown in Fig. 13.

Equation 27 gives the number of symbols in one cross layer packet that are required for timing frequency synchronization which achieves the accuracy of application layer time synchronization using N time stamps. Fig. 14 shows the relation of number of time stamps at the application layer and total symbols required in one cross layer packet.

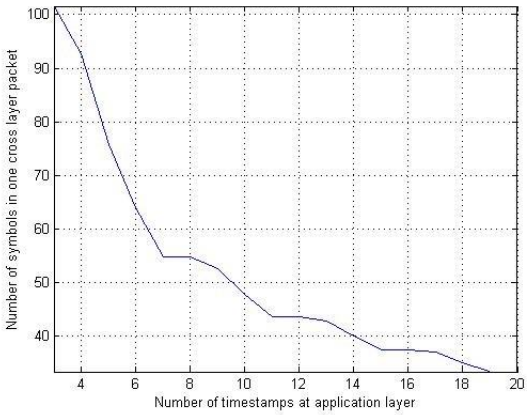

Fig. 14. Required symbols for cross layer packet with respect to time stamps used at application layer

Fig. 15 shows the structure of the cross layer packet.

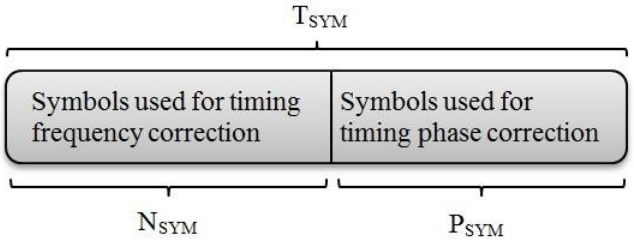

Fig. 15. Cross layer packet

The total symbols in one cross layer packet that can be used for timing frequency synchronization as well as timing phase synchronization can be written as

$$T_{SYM} = N_{SYM} + P_{SYM} \tag{28}$$

where T_{SYM} are total symbols, N_{SYM} are the symbols used for timing frequency synchronization and P_{SYM} are the symbols used for timing phase synchronization which are equal to the number of symbols in one time stamp. Using equation 27 and 28 we have

$$T_{SYM} = \frac{T_T}{\pi}\sqrt{\frac{N(N^2-1)}{\frac{2}{3}\xi\frac{E_S}{N_0}A}} + P_{SYM} \tag{29}$$

VI. CONCLUSIONS

The cross layer time synchronization uses one packet to synchronize in time phase and time frequency (drift) and that eliminates the need of application layer time synchronization protocol and hence it saves energy required for computations and transmission of multiple timestamps exchange and proves good for the energy constrained environments. The experimentation results shows that the heart of both the physical layer and application layer clock is hardware oscillator (clock). The frequency offset (drift) at application layer and physical layer is the same offset that occurs in hardware clock as both of the mentioned clocks are derived from the hardware clock. That's why the frequency offset computed at the physical layer is computed to be same as the frequency offset at the application layer and can be corrected directly using physical layer clock frequency offset. There will be no need to use any multiple message exchange algorithm on the application layer to set the application layer clock frequency offsets. This cross layer design can be used in any distributed wireless network like Wireless Sensor Network. This approach can also estimate offsets in multi user case, such as cooperative communication. Nodes don't have to participate in synchronization procedure sequentially for higher layer synchronization and instead a cross layer approach can estimate all offsets in simultaneous cooperative synchronization.

VII. ACKNOWLEDGMENTS

Thanks to Almighty ALLAH, The Merciful, The Beneficent, whose bountiful blessings and exaltation flourished our thoughts and thrived our ambitions to have the cherish fruit of my modest efforts in the form of this research. We offer our humblest thanks from the core of my heart to the holy Prophet (Peace be upon Him) who is forever a model of guidance and knowledge for humanity. We feel great depth of obligation for our loving parents and wife whose prayers have enabled us to reach at this stage.

I (Usman Hashmi) owe a special debt of gratitude to my reverend supervisor Dr. Qasim Mahmood Chaudhari for their invaluable guidance, expert advices, cooperation, encouraging attitude, positive criticism and healthy suggestions. I also wish to record my sincere appreciations to Dr. Imran Shafi, Dr. Affan Ahmed, Dr. Ismail Shah, Dr. Jamil Ahmed and Engr. Ammar Ajmal for their support and guidance.

NOMENCLATURE

$\mu(k)$	Fractional interval
$\omega(t)$	Angular frequency of oscillator
τ	Timing frequency error
ξ	Loop parameter
$a(k)$	Binary PAM kth symbol
$e(k)$	Timing error signal
E_P	Energy required to transmit one packet
E_S	Energy required to transmit one sample
E_{APP}	Energy at application layer
E_{eff}	Energy efficiency
E_{PHY}	Energy at physical layer
E_{sym}	Energy required to transmit one symbol
f_d	Symbol rate
f_e	Frequency error
f_s	Sampling rate
m	Slope of the fractional change
$m(k)$	Basepoint index
N_b	Bits per packet
N_m	Bits per sample
N_P	Number of packets
N_{bsym}	bits per symbol
N_{SYM}	Symbols used for timing frequency synchronization
N_{sym}	Total symbols
P_{SYM}	Symbols used for timing phase synchronization
R	Symbol rate

T Symbol time

T_T Total transmission time

T_{SYM} Number of symbols per cross layer packet

x_i ith timestamp of receiver

y_i ith timestamp of transmitter

REFERENCES

[1] Ardakani, S. P., Padget, J. and Vos, M. D., HRTS: A Hierarchical Reactive Time Synchronization Protocol for Wireless Sensor Networks,*Ad Hoc Networks. Springer International Publishing*, vol. 129, pp. 47-62 , 2014.

[2] Yildirim, K. and Kantarci, A., External Gradient Time Synchronization in Wireless Sensor Networks, *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 633-641, 2014.

[3] Yildirim, KS. and Kantarci, A., Time Synchronization Based on Slow Flooding in Wireless Sensor Networks, *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no.1, pp. 224-253, 2014.

[4] Goncalves, F. , Suresh, L., Pereira, RL., Trindade, J. and Vazao, T., Light-Weight Time Synchronization For Wireless Sensor Networks, *IEEE Conference on Future Internet Communications (CFIC)*, pp. 18, 2013.

[5] Xu, M. and Xu, W., TACO: Temperature-Aware Compensation for Time Synchronization in Wireless Sensor Networks, *International IEEE Conference on Mobile Ad-Hoc and Sensor Systems (MASS)* , 2013.

[6] He, J., Chen, J., Cheng, P. and Cao, X., Secure Time Synchronization in Wireless Sensor Networks: A Maximum Consensus Based Approach, *IEEE Transactions on Parallel and Distributed Systems*, 2013.

[7] Kumar, S., Lee, Y. and Lee, SR., Time Synchronization in Wireless Sensor Networks: Estimating Packet Delay, *The 1st International Conference on Convergence and it's Application*, vol. 24, pp. 68 - 71, 2013.

[8] Kumar, S., Chandra, AA., Sanjaa, B., Hur, K. and Lee, SR., Estimation of Packet Delay Components for Time Synchronization in Wireless Sensor Networks, *Advanced Science and Technology Letters*, vol.28, pp.104-109, 2013.

[9] Liu, J., Zhou, Z., Peng, Z., Cui, J.-H., Zuba, M. and Fiondella, L., MobiSync: Efficient Time Synchronization for Mobile Underwater Sensor Networks, *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 2, pp. 406 - 416, 2013.

[10] Jiang, Y., Fan, Y. and Chen, X., Time Synchronization Protocol for Wireless Sensor Networks with Node Monitoring, *Journal of Information and Computational Science*, vol. 10, no. 4, pp. 1213 - 1220, 2013.

[11] Wu, Y.-C., Chaudhari, Q. and Serpedin, E., Clock Synchronization of Wireless Sensor Networks, *Signal Processing Magazine, IEEE*, vol. 28, no. 1, pp. 124 - 138, 2011.

[12] Ferrari, F., Zimmerling, M., Thiele, L., and Saukh, O., Efficient network flooding and time synchronization with Glossy, *10th International Conference on Information Processing in Sensor Networks (IPSN)*, Chicago, IL , 2011.

[13] Rahamatkar, S. and Agarwal, A., An Approach towards Lightweight, Reference Based, Tree Structured Time Synchronization in WSN, *Advances in Computer Science and Information Technology*, vol. 131, pp. 189-198, 2011.

[14] Ranganathan, P. and Nygard, K., Time synchronization in wireless sensor network: A survey, *International Journal of UbiComp*, vol. 1 , no. 2, pp. 92-102, 2010.

[15] Lasassmeh, S. and Conrad, J., Time synchronization in wireless sensor networks: A survey, in *IEEE SoutheastCon 2010 (SoutheastCon)*, Concord, NC, 2010.

[16] Rice, M., *Digital communication: A discreet time approach*, Prentice Hall, 2008.

[17] Chaudhari, Q. M., Serpedin, E., Suter, B. W., Kyoung-Lae, N., Novel Clock Phase Offset and Skew Estimation Using Two-Way Timing Message Exchanges for Wireless Sensor Networks, *Communications, IEEE Transactions, Vol. 55, No. 4*, pp. 766-777, 2007.

[18] Chakrabortyy, I., Lynch, N., Fan, R. , Clock Synchronization for Wireless Networks, *Principles of Distributed Systems, Vol. 3544/2005 , No. 900*, 2005.

[19] Buy, U., Kshemkalyani, A. D., Sundararaman, B., Clock Synchronization for Wireless Sensor Networks: A Survey, *Ad Hoc Networks, Vol. 3 , No. 3*, pp. 281-323, 2005.

[20] Li, Q., Rus, D., Global Clock Synchronization in Sensor Networks, *INFOCOM, Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004.

[21] Maroti, M., Kusy, B., Simon, G., Ledeczi, A., The flooding time synchronization protocol, *SenSys '04 Proceedings of the 2nd international conference on Embedded networked sensor systems* , New York, USA, 2004.

[22] Liu, K., Thesis on Architectures for symbol timing synchronization, Brigham Young University, 2004.

[23] Eskelinen, P. , Ono, S., Kihara, M., *Digital Clocks for Synchronization and Communications*, Artech House, 2003.

[24] Veerarittiphan, C., Sichitiu, M. L., Simple, accurate time synchronization for wireless sensor networks, *Wireless Communications and Networking*, pp. 1266-1273, 2003.

[25] Ganeriwal, S., Kumar, R. and Srivastava, M. B., Timing-sync protocol for sensor networks, in *SenSys '03 Proceedings of the 1st international conference on Embedded networked sensor systems*, New York, USA, 2003.

[26] Abdel-Ghaffar, H. S., Analysis of synchronization algorithms with time-out control over networks with exponentially symmetric delays,*Communications, IEEE Transactions*, vol. 50, no. 10, pp. 1652 1661, October 2002.

[27] Girod, L., Estrin, D., Elson, J., Fine-grained network time synchronization using reference broadcasts, *ACM SIGOPS Operating Systems Review - OSDI 02: Proceedings of the 5th symposium on Operating systems design and implementation*, New York, USA, 2002.

[28] Romer, K., Time synchronization in ad hoc networks, *MobiHoc 01 Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing*, New York, USA, 2001.

[29] Harris,F. J. and Rice, M., Multirate digital filters for symbol timing synchronization in software defined radios,*Selected Areas in Communications*, vol. 19, no. 12, pp. 2346 - 2357, December 2001.

[30] Moeneclaey, M., Fechtel, S. A., Meyr, H., *Digital Communication Receivers: Synchronization, Channel Estimation and Signal Processing* (2nd ed.) USA: Wiley-Interscience, 1998.

[31] D'Andrea, A. N., Mengali, U., *Synchronization Techniques for Digital Receivers (Applications of Communications Theory)* (1st ed.) Pisa, Italy: Springer, 1997.

[32] Gardner, F. M., Gardner, R.C., Interpolation in digital modems. I. Fundamentals, *Communications, IEEE Transactions, Vol. 41, No. 3*, pp. 501-507, 1993.

[33] Erup, L., Gardner, F. M., Harris, R.A., Interpolation in digital modems. II. Implementation and performance, *Communications, IEEE Transactions, Vol. 41, No. 6*, pp. 998-1008, 1993.

[34] Kay, S. M., *Fundamentals of Statistical Signal Processing, Estimation Theory* (1st ed.), Prentice Hall, 1993.

[35] TMS320C6713 DSP Starter Kit (DSK), *Texas Instruments*, 1995-2014. [Online]. Available: http://www.ti.com/tool/tmdsdsk6713.