Time Synchronization in Network-Centric Sensor Networks

Sejal Raje and Qilian Liang
Department of Electrical Engineering,
University of Texas at Arlington, 416 Yates Street, Nedderman Hall, Rm 518, Arlington, TX 76019
Email: sejalraje@yahoo.com, liang@uta.edu

***Abstract*— Time Synchronization is a crucial component of infrastructure for wireless sensor networks (WSN). The design of new synchronization methods for WSNs needs to satisfy the unique requirements and constraints in terms of precision, lifetime, energy, and scope of the synchronization. In this paper we have designed and compared three techniques for achieving synchronization in network centric sensor networks, using Sequential least squares, Kalman filter and Fuzzy logic systems. These techniques achieve a highly accurate, long-term and adaptive synchronization by forming a time conversion scale between clocks of two nodes in the network. A synchronization protocol based on any of these techniques will be very energy-efficient, lightweight, multimodal, tunable, and even scalable.**

I. Introduction

A. Overview

In Wireless Sensor Networks (WSN), time synchronization is needed by many applications, for e.g., data fusion, temporal delivery of events, duplicate detection, target detection and tracking etc. Although many solutions have been designed and implemented for problem of time synchronization in traditional networks, many factors in sensor networks, such as *energy constraint*, *dynamic topology*, *variety of applications* and *cost and form factor*, make traditional methods unsuitable for WSNs. After these observations, some design requirements have been formulated to address the challenging task of WSN time-synchronization [1], wherein it is better for any scheme suggested to be energy efficient, adaptive to application's needs, robust and scalable, and provide only the necessary and sufficient synchronization.

The many factors causing errors in clocks of sensor nodes or in synchronization algorithm can be divided into two categories: ***1. Oscillator Characteristics :*** Since sensor nodes' clocks run on very cheap oscillators, the two characteristics of clock oscillators that are the main sources of error are *accuracy* (or resolution), which is a measure of difference between oscillator's expected (ideal) frequency and actual frequency, and *stability*, which is oscillator's tendency to stay at the same frequency over time. ***2. System and Network issues :*** These sources of non-determinism in the message delivery latency have been categorized in four type of delays [4]: '*Send Time*', '*Access Time*', '*Propagation Time*' and '*Receive Time*'. All the above factors finally result in time or phase offset, frequency bias (skew) and frequency drift between the clocks of two nodes. In this paper, we mainly consider the resulting factors due to oscillator problems.

B. Outline of A Synchronization Protocol

Our interests in synchronization arise from a recently suggested approach of synchronizing sensor nodes by forming a time conversion scale between them [2] [3] [4]. As found in many applications, consider a sensor network consisting of many small clusters of nodes; each of them consisting of a clusterhead and many nodes that are within the broadcast range (*single hop* distance) from it. The goals here is to get all the nodes in a cluster to synchronize with the clusterhead. In this scenario, the clusterhead, also referred to as the *sender* (S) node, will broadcast timestamps - containing time in it's own clock - at regular intervals to all other nodes in its cluster, which are the *receivers* (R). Then, using these timestamps and one of the methods suitable for existing conditions, each receiver forms a time-conversion-scale between itself and sender node according to its own clock, and thus predicts the time in the senders clock at any point of time. By having such a reference scale, a node is synchronized with sender, because it can convert the time in it's own clock into other node's time.

Even though similar approaches have been tried before for WSN, the problem with these techniques is that they do not take into account the frequency drift of clocks or environmental effects, which can affect the accuracy of synchronization, especially over a longer period of time. We have tried designing methods that can address these issues while fulfilling the other requirements of WSN time-synch problem.

In the rest of the paper, section II, III, and IV respectively present the designs of Sequential least squares, Kalman filter and Fuzzy logic approaches. Simulations and observations shown in section V and section VI concludes the paper.

II. Sequential Least Squares Approach

Assume that the clocks of S and R are of the same type, so under ideal conditions, they should run at the same frequency, without any initial phase error. But in reality, each clock will drift from its ideal conditions. From [7], [5], an approximation characterizing a clock oscillator, and the resulting eq for drift of a single clock from ideal conditions, are given as:

$$f = f_{nom} + \Delta f_0 + f_d.(t - t_0) + \Delta f_n(t) + \Delta f_e(t) \quad (1)$$

$$\Delta\Phi(t) = \Delta\Phi(t_0) + \Delta f_0.(t - t_0) + \frac{f_d}{2}.(t - t_0)^2 + \Delta f_n(t) + \int_{t_0}^{t} \Delta f_e(\tau) \quad (2)$$

1-4244-0445-2/07/$20.00 ©2007 IEEE

where t_0 = starting time ; f_d = frequency drift or aging rate;
f_{nom} = ideal frequency; Δf_0 = initial frequency error;
Δf_n = short-term frequency instability (noise) term;
Δf_e = environmental term.

From the above equations, we can derive the relative clock drift (or time offset) between clocks of two nodes as referred to an ideal clock ($\Delta t_{12}(t)$). Using that expression the offset between the receiver's clock and the sender's clock (Δt_{SR}), at time t_R in the receiver's clock is given as,

$$\Delta t_{SR}(t_R) = (t_{0S} - t_{0R}) + \frac{\Delta f_{0SR}}{f_{nom}}(t_R - t_{0R}) + \Psi_n(t_R) + \frac{f_{d_{SR}}}{2f_{nom}}(t_R - t_{0R})^2 + \Delta t_{eSR}(t_R) \quad (3)$$

and t_{R0} can be assumed to be 0. Thus, the phase or time offset between two clocks at any given time results from a combination of initial phase offset, frequency bias, frequency drift, noise due to the environmental terms, and the random clock jitter. The clock jitter does not lead to accumulated time errors. But as more time passes from the synchronization point, the drift and environmental terms become more significant.

Here, the SLS estimates the clock parameters between a sender and a receiver node, and using those predicts the time-offset between them at the next time step. So node R seeks to predict Δt_k^{SR}, given past m observations of time-offset, $\Delta t_{k-1}^{SR}, ..., \Delta t_{k-m}^{SR}$ from consecutive timestamps received from sender node. In this design, effect of clock drift is considered, but not the environmental terms. Refereing to equation 3, our model for this sequential least squares approach is of the form,

$$\tilde{y_k} = H_k x + v_k \quad (4)$$

where, $\tilde{y}_k = \Delta t_k^{SR}$ denotes the k^{th} measurement,
$H_k = [1 \quad t_k \quad t_k^2]$ is the basis function matrix,
and $x = [x_1 \quad x_2 \quad x_3] = [(t_{0S} - t_{0R}) \quad \frac{\Delta f_{SR}}{f_o} \quad \frac{f_{dSR}}{2f_o}]$ is the parameter matrix. Clock parameters in estimation are initial offset, frequency bias, and frequency drift respectively. The clock jitter term, $\Psi_n(t_R)$, is assumed to be zero mean, white gaussian noise and is used as the measurement noise, v, at any instance of time. Hence the measurement error covariance is given as, $R^{-1} = cov\{\Psi_n(t_R)\}$. The basis functions are independent, and weight matrix remains block diagonal, because the samples Δt_k^{SR} at various times t_k are i.i.d. So as a new timestamp is available, we calculate an updated $(k+1)^{th}$ estimate of clock parameters, using the k^{th} estimates of them and the $(k+1)$ measurements and associated side calculations. We use the covariance recursion form of SLS algorithm. Reader can refer to [9] for further details on working of SLS. The sequential process can be started at any step by initial values for clock parameters - taken either from theory or previous experiments or derived from batch estimation of a few initial measurement samples - i.e., an *a priori* estimate x_0, and the *a priori* estimation error covariance matrix $P_0 = Q = cov(\mathbf{w})$, $\mathbf{w}$ being the errors in *a priori* estimates. Since this design is a minimum-variance (MV) estimator, we get an unbiased estimates of the clock parameters.

III. Kalman Filter Algorithm

Viewed as an extension to the SLS algorithm, Kalman filter approach with it's dynamic system parameters, model for system state estimation and it's *predictor-corrector* form, can prove useful to make the synchronization adaptive to the environmental noise and the dynamic clock drift. Our approach is motivated from various attempts in past of using Kalman filter as a tool in the formation of stable timescales like GPS composite clocks [6]; but the time-synch problem worsens in case of WSN due to cheap oscillators of sensor nodes and severe environmental conditions.

The problem formation is same as that of SLS. For simulating a sensor-network scenario, we have assumed that the clock drift changes after every few seconds due the environmental and other noise. The measurement model will be given as,

$$y_k = x_S(t_k) - x_R(t_k) = H.x_k + v_k \quad (5)$$

where y_k is the phase or time offset between S and R, and $x_S(t_k)$ and $x_R(t_k)$ are the phase states of the sender and the receiver respectively, at time t_k. Even though these expressions are in terms of phase offset, they can be easily converted in terms of time offset, by dividing each term by the ideal frequency of nodes' clocks. Doing so will not change the nature of the problem. For the system model, let x_{SR} be the phase-offset state, y_{SR} the frequency bias state and z_{SR} as the relative drift state between the clocks of sender and receiver. Here, $x_{SR}(t)$ is same as x_k in the equation 5, and $y_{SR}(t)$ should not be confused with y_k. Then the evolution of this offset between S & R from time step $(t-\tau)$ to t is given by following equations:

$$x_{SR}(t) = x_{SR}(t-\tau) + \tau.y_{SR}(t-\tau) + \frac{1}{2}\tau^2.z_{SR}(t-\tau) + w_{x_{SR}}(t) \quad (6)$$

$$y_{SR}(t) = y_{SR}(t-\tau) + \tau.z_{SR}(t-\tau) + w_{y_{SR}}(t) \quad (7)$$

$$z_{SR}(t) = z_{SR}(t-\tau) + w_{z_{SR}}(t) \quad (8)$$

where the process noise vector, which specify the level of relative noise components between the two clocks, is given as

$$W_{SR}(t) = [\ w_{x_{SR}}(t) \quad w_{y_{SR}}(t) \quad w_{z_{SR}}(t)\ ]^T \quad (9)$$

This leads us to a basis vector : $H = [\ 1 \quad 0 \quad 0\ ]$

The random clock jitter is used as the measurement noise, v, at any instance of time. Hence the measurement error covariance is given as, $R^{-1} = cov\{\Psi_n(t_R)\}$ which is assumed to be a zero-mean Gaussian random variable. Hence the state vector is: $x_k = [\ x_{SR}(t) \quad y_{SR}(t) \quad z_{SR}(t)\ ]$ which is updated at every re-synchronization instance, using Φ and W. The transition matrix, showing the transition of states from time step t_k to t_{k+1} is,

$$\Phi = \begin{pmatrix} 1 & \tau & \frac{\tau^2}{2} \\ 0 & 1 & \tau \\ 0 & 0 & 1 \end{pmatrix}$$

Thus the next value of the relative drift between two nodes is obtained by adding some noise to it's previous value; which affects the frequency bias between the two clocks,

which in turn changes the phase offset between them. Here, a multiplication factor, τ, which reflects the change in drift over a short time, is taken as the time between subsequent synch pulses. The sources of all the noises lie well in the environmental effects, aging of the oscillator, etc. and it also accounts for the errors in system model. The vector W is derived from the Hadamard and Allan variances - the complete description of which are out of the scope of this work. For a detailed working of Kalman filter, readers can refer to [9].

IV. Fuzzy Logic Algorithm

The rule-based Fuzzy logic systems (FLS) are extensively used for forecasting of time-series [8] [10]. For the FLS solution, first the trend is calculated by a sliding window technique and data is detrended, which is then fed as input to the FLS. Given a collection of these N data points, they are first partitioned into a *training* data subset - used for learning period of FLS - with D data points, $x(1), x(2), ..., x(D)$, and a *testing* data subset - for actual prediction - with $(N - D)$ data points, $x(D+1), ..., x(N)$. For both these subsets, we use the previous 4 data samples as antecedents for forecasting the next sample of data (consequent). I.e. $x(k-3), x(k-2), x(k-1), x(k)$ were used to predict $x(k+1)$. Number of training points were about 30 to 50 for a dataset of 100 samples.

We have chosen *singleton fuzzification*, *product implication* and *Height Defuzzifier*. Each antecedent has two membership functions (MFs) - High and Low, chosen to be Gaussian. The number of rules are $2^4 = 16$. Now, if x_1 is an input value from input set X_1, F_p^l is MF of rule l that is of type p, $\star$ and T are the chosen t-norm, and G^l is rule output fuzzy set. Then the rules are designed as:

R^l : IF $x(k-3)$ is F_1^l and $x(k-2)$ is F_2^l and $x(k-1)$ is F_1^l and $x(k)$ is F_2^l, THEN $x(k+1)$ is G^l.

For singleton fuzzification, the firing level of l^{th} rule is,

$$\mu_{F_1^l}(\acute{x}_1) \star \mu_{F_2^l}(\acute{x}_2) ... \star \mu_{F_p^l}(\acute{x}_p) = T_{i=1}^{p} \mu_{F_i^l}(\acute{x}_i) \tag{10}$$

and the output fuzzy set B^l for rule R^l is such that

$$\mu_{B^l}(y) = [T_{i=1}^{p} \mu_{F_i^l}(x_i)] \star \mu_{G^l}(y) \tag{11}$$

The output of the height defuzzifier, which replaces each G^l by a crisp value at the point, $\bar{y}^l$, having maximum membership grade (height) in l^{th} output set, is given as:

$$y_h(\mathbf{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l \mu_{B^l}(\bar{y}^l)}{\sum_{l=1}^{M} \mu_{B^l}(\bar{y}^l)} \tag{12}$$

For the antecedent MFs, initial mean for 'High' MF was $m_t + \sigma_t$ and that for 'Low' MF was $m_t - \sigma_t$, and the std for both MFs was σ_t, where m_t and σ_t are the mean and std of training data set. The initial consequent fuzzy set comprises of random numbers in the given data range. So, we have total 144 parameters. After setting up the initial design, a steepest descent (back-propagation) algorithm was used to train these parameters using the training data. After training, the rules are fixed and then this FL forecaster is used for the testing subset.

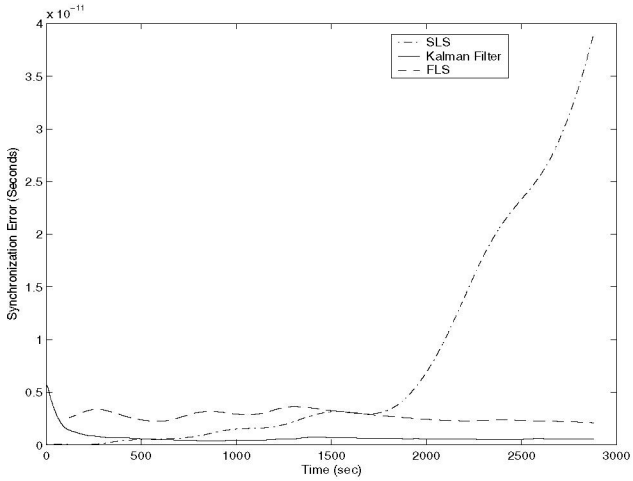

Fig. 1. Mean square synchronization error for dataset 2

The design described above works well with batch mode of prediction; but a better online performance is obtained in this case by a sequential design. For sequential design, after setting the architecture and initial rules, when stored samples are equal to the number of antecedents, it is fed to FLS to predict the next sample. When the next data sample is actually received, the error between predicted and measured value of data is used in the back-propagation algorithm to train the parameters of FLS online, which are then used for prediction at the next time-step.

V. Simulations and Comparisons

We have tested and compared the performances of all the three algorithms, on various datasets, which comprise of the time offset between the sender node and a single receiver node, taken at the sampling rate of 60 to 360 seconds. The first set has constant clock parameters and no environmental noise. In the second dataset, the clock parameters are made dynamic, but environmental noise term is kept low. Both these datasets can be considered to be similar to an indoor-scenario, where the clock parameters and hence the offset between 2 nodes' clock will vary in a non-linear pattern, yet without any drastic random changes. In the third dataset, both variation in the clock parameters and environmental noise are high, which results in highly random changes in clock drift. This dataset is an ideal example of time-offset variation in an outdoor scenario. The mean squared error (mse), i.e. synchronization error, between the predicted value of time-offset and the actual measurement is calculated and plotted for all three approaches. (But due to constraints of space, only few are presented here).

All the methods achieve very good precision on most datasets. The mse is in the range of 10^{-12} or lower, which means that the nodes will be synchronized within an accuracy of less than 10 microseconds. All methods achieved a much higher sampling period (360 seconds) as compared to that obtained from short-term timescale formation of current techniques (10 to 60 seconds). The broadcast nature of the

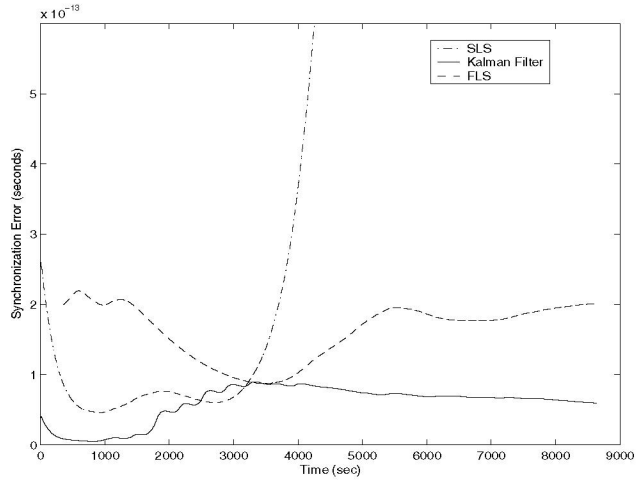

Fig. 2. Mean square synchronization error for dataset 3

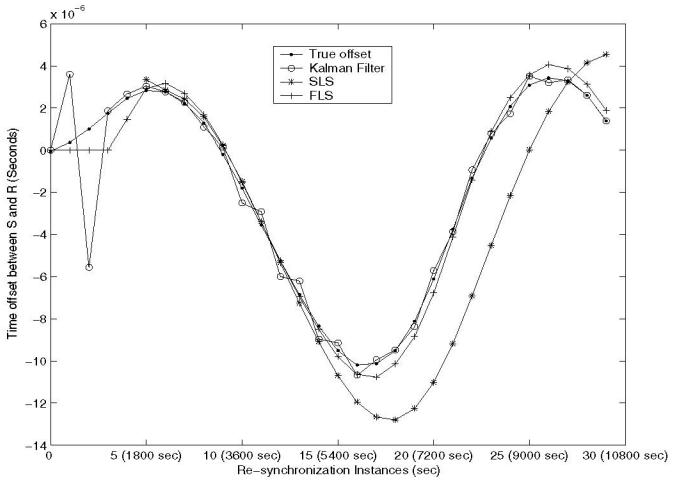

Fig. 3. Measured and estimated values of time offset between Sender and Receiver for Dataset 3

protocol provides a uniform precision over the cluster. All the methods converge very fast, which leaves us room for approximate choice of the initial conditions. FLS and Kalman filter are adaptive to environmental effects on nodes' clocks and hence preserve the accuracy under dynamic conditions. The advantage of FLS over the other two is in its model-free nature, which is very useful when the offset between clocks vary in a highly random fashion. The storage requirement of all techniques is very small, a few samples, though FLS needs more space than other two. SLS and FLS have much less complexity and processing time than Kalman filter approach.

Since the synch pulses are coming after a considerable duration of time, the nodes can sleep in this time and wake-up just to receive the pulses, thus saving a lot of energy. The S-R nature of protocol and a minimum window size keeps the messaging overhead very less. The protocol is flexible and it can adapt easily to provide the 'necessary and sufficient' synchronization for a given application. It is possible to tune it to required accuracy by choosing appropriate values of parameters such as, transition matrix, covariance matrices, mean and std of membership function, window size, and sampling period. The prediction of sender's clock can go on even in the sleep mode, so it will always assure some accuracy in case of both '*Always On*' and '*Post-facto*' synchronization. The performance of these techniques does not depend on the number of nodes in the cluster (density), node failures, or a change of clusterhead. Also, nodes can be synchronized to physical time without much stress on resources, by incorporating only the clusterhead with a GPS receiver.

VI. Conclusion

All the three approaches have been successful in achieving high accuracy of a few microseconds even at a high sampling interval for synchronization beacons, at the same time making the synchronization scheme adaptive to the environmental effects. Owing to it's simpler design, the Sequential Least Squares algorithm can be more useful in an indoor-scenario, whereas because of their recursive and dynamic nature, Kalman Filter and Fuzzy Logic algorithms yield a very good performance for an outdoor-scenario. The nature of the algorithms also provides many other advantages such as: energy-efficiency, multimodal and tunable to the needs of application, scalability, and low overhead. These factors make our schemes applicable in a variety of applications. We suggest some future research to provide in-depth analysis of issues like network-wide synchronization, or adding a correction for network delays, combined with these techniques.

Acknowledgement

This work was supported by the U.S. Office of Naval Research (ONR) Young Investigator Program Award under Grant N00014-03-1-0466.

References

[1] Jeremy Elson, Kay Römer, "Wireless Sensor Networks: A New Regime for Time Synchronization" *ACM SIGCOMM Computer Communication Review*, vol 33, no. 1, January 2003, pp. 149-154

[2] J. Elson, D. Estrin, "Time Synchronization for Wireless Sensor Networks" *Proc. 15th International Parallel and Distributed Processing Symposium*, April 2001, pp. 1965-1970

[3] S. Ganeriwal, D. Ganesan, M. Hansen, M. Srivastava, D. Estrin, "Rate-Adaptive Time Synchronization for Long-lived Sensor Networks" *ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, June 2005, pp. 374-375

[4] Jeremy Elson, Lewis Girod, Deborah Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts" *ACM SIGOPS Operating Systems Review: Special issue on Physical Interface*, vol 36, SI Winter 2002, pp. 147-163

[5] An-swol Hu, Sergio D. Servetto, "Asymptotically Optimal Time Synchronization in Dense Sensor Networks" *2nd ACM Int. Conference on Wireless Sensor Networks and Applications* , Sept. 2003, pp. 1-10

[6] Charles A. Greenhall, "Forming stable timescales from the JonesTryon Kalman filter" *IoP Journals: Metrologia: IV International Time-Scale Algorithms Symposium*, vol 40, no. 3, June 2003, pp. 335-341

[7] Douglas Arnold, "Stochastic Model Estimation of Network Time Variance" *Tech. Rep., TRUETIME*,

[8] Qilian Liang, "Ad hoc wireless network traffic-self-similarity and forecasting" *Communications Letters, IEEE*, vol 6, no. 7, July 2002, pp. 297-299

[9] John L. Crassidis, John L. Junkins, "Optimal Estimation of Dynamic Systems" CRC Press, 2004

[10] Jerry M. Mendel, "Uncertain Rule-Based Fuzzy Logic Systems" Prentice Hall PTR, 2000