On the Impact of Time Synchronization on Quality of Information and Network Performance

XXX,YYY
{xxx,yyy}@honeywell.com
Honeywell Labs
3660 Technology Drive, Mailstop: MN65-zzz
Minneapolis, MN 55418
phone: 612 951 xxx 612 951 yyy

***Abstract*—Recent new hardware and technology enable low-power inexpensive distributed sensor networks. To realize certain applications such as real time event detection, target tracking and system monitoring, time synchronization is essential. In the literature, a book of time synchronization techniques have been proposed. In general, the goal of a time synchronization mechanism is to devise a scheme that improves accuracy with lower energy consumption. Yet, there is clearly a trade off between accuracy and energy consumption. It would not be always desirable to deploy the most accurate time synchronization as it might drain the power. The choice of a time synchronization mechanism will depend on application's demand on timing accuracy as well as the energy budget. To estimate application's demand and select a proper time synchronization mechanism, the relationship between application's performance or QoI (Quality of Information) and time synchronization services are yet to be investigated. This paper formalizes the relationship between the performance of applications and time synchronization services based on analytic frameworks using representative applications, namely, event detection and estimation. The analysis shows the impact of timing errors for different event duration, target moving speed, number of sensors, sampling frequencies, etc. The analysis framework can also be used to extract the maximum synchronization errors that each application can sustain to achieve the desired QoI.**

Moreover, time synchronization plays a key role in energy-saving requirements. For example, accurate time synchronization can improve the efficiency of duty-cycling and thus save energy. In this paper, we will investigate the savings in power and bandwidth by employing different stabilities of clock sources.

I. Introduction

Recent new hardware and technology enable low-power inexpensive distributed sensor networks. To realize certain applications such as real time event detection, target tracking and system monitoring, time synchronization is essential. In the literature, a book of time synchronization techniques have been proposed including Time-Sync Protocol for Sensor Network (TPSN) [3], Reference Broadcast Synchronization (RBS) [2], elapsed time [5], etc. In general, the goal of a time synchronization mechanism is to devise a scheme that improves accuracy with lower energy consumption. Yet, there is clearly a trade off between accuracy and energy consumption. It would not be always desirable to deploy the most accurate time synchronization as it might drain the power. The choice of a time synchronization mechanism will depend on application's demand on timing accuracy as well as the energy budget. To estimate application's demand and select a proper time synchronization mechanism, the relationship between application's performance or QoI (Quality of Information) and time synchronization services are yet to be investigated. In this paper, we aim to formalize the relationship between the performance of applications and time synchronization services based on analytic frameworks using representative applications, namely, event detection and estimation.

This work was supported by US-UK ITA Grant

Moreover, time synchronization may play a key role in energy-saving requirement. For example, accurate time synchronization can improve the efficiency of duty-cycling and thus save energy. Even though our study implies that energy saving by improving duty cycling through better clock accuracy would not be significant with low-power sensor platforms (e.g., motes?). We will show the analysis with regard to energy saving in Section VI.

II. Related Work

Time synchronization techniques – RBS, TPSN, Tiny-sync, mini-sync, elapsed time,

III. Application QoI and Time Synchronization

A. Background

Questions that we like to answer through this analysis includes the followings. 1. What will happen if timing errors are not bounded - a sample may have very inaccurate timing value - can we detect this unusual event (or fault)?

2. What could be the upper bound of timing error versus sample rate for estimation application - Is there any relationship between sample rate and timing error (such as if timing errors are high, do we need more frequent samples?)

3. Can we do some fault detection and correction. Such as maintaining history of samples from each sensor, we can detect some abnormal behaviors/samples? Or among samples from sensors at certain duration, we can do also some filtering? 4. If we want to do this kind of filtering, do we need better time-sync accuracy?

5. Does the error characteristic known to the filter help the accuracy of filtering? Let's say, we can provide a fairly accurate error model, can filter achieve highly accurate tracking?

1) Time synchronization error: In our paper, we assume that a sensor is equipped with an internal hardware oscillator that provides vibration frequency to keep the local clock continuously running. The vibration frequency will depend on the size, thickness and cutting edge of a crystal inside the oscillator and the condition of sensor's surrounding physical environment.

Generally speaking, we have two sources of synchronization errors for distributed algorithms: (1) local time change due to clock drift; (2) message delay uncertainty. These two error sources will differently impact on the synchronization error depending on the synchronization technique [8].

Considering a single node, the local clock of a sensor 'X' at Coordinated Universal Time (UTC) t can be represented as [10]

$$t_x = a_x t + t_{0x} + \text{Drift}_x(t),$$

where a_x is the clock skew, i.e., difference between X's oscillator and the ideal perfect one, and t_{0x} is the initial time offset. Drift$x(t)$ is the clock drift at time t. Clock drift changes with environment. It is an environment-dependent phenomenon and hard to be modeled accurately. More detailed discussion on clock drift will be shown in Section **??**.

If we consider two nodes, then we also consider message latency and other communication-related delay. It can be classified as the followings further [10].

- **P**rocessing delay: the time spent at node S (sender) to prepare and process the packet, and transfer it to the networking component.
- **A**ccess delay: the delay to acquire the wireless medium
- **P**ropagation delay: the delay to transmit the propagate over the medium
- **R**eceive delay: time for the node R (receiver) to process the packet and get the reading of its local clock

In total, the delay to include all the communication can be represented as $PAPR_{StoR}$, and the time of R relative to S can be shown as $t_r = a_{rs}(n)t_s + t_{0rs}(n) + PAPR_{StoR}$,$nT < t < (n+1)T$, where T is the sampling rate, and a_{rs} is the offset difference between R and S. Detailed information can be found in [10].

Based on this representation, [10] derives the limitation on errors between any networked two nodes in terms of value and distribution. The paper showed that the distribution of limitation of errors is uniform if communication-related delay is uniformly bounded. The results from the paper are used in our analysis.

B. Analysis Framework

1) Event Detection Application: Section 2.2.1.1 Event detection framework Section
2.2.1.2 Analysis on performance of event detection versus various timing error models Section
2.2.2 Target Estimation Application Section
2.2.2.1 Target Estimation Filtering Algorithm Section
2.2.2.2 Analysis on performance of target estimation versus various timing error models Section

C. Discussion on relationship between time synchronization service and application QoI

IV. A Simple Probabilistic Detector

Consider a system of N sensors that are identical and detect an event of time period τ such as a gunshot or an explosion. Let the probability that an event E has occurred, given that the sensor measurement s_i exceeds a threshold T be given as:

$$P(E|s_i > T) = p \quad \forall i, \tag{1}$$

where $i = 1 \ldots N$ goes over all the sensors, and $0 < p \leq 1$. Similarly, let the probability of threshold crossing when the event has not occurred, E^c be given as:

$$P(E^c|s_i > T) = q \forall i, \tag{2}$$

and $0 < q \leq 1$. Let the number of samples produced by a sensor in the course of the event equal $M = \lfloor \tau f_s \rfloor$, where f_s is the sampling frequency of all the sensors. Considering the probabilities identical for all of the sensors is realistic if the sensors respond to low frequency acoustics or thermal radiation signature, which does not attenuate very much with distance. Furthermore, we can also assume that the detection is at the same instant for all of the sensors if the sensors are not separated by distances larger than the signal would travel in a sampling interval (e.g., for an acoustic sensor system 10ms or 100Hz sampling gives around 3 meters separation for the sensors).

Now consider a detection function defined as follows that combines data from all of the sensors to detect an event and determine its time:

$$f = \frac{1}{MN} \sum_{i=1}^{MN} d_i \tag{3}$$

$$d_i = \begin{cases} 1 & s_i > T \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The relevant questions that can be asked at this point are–

- What is the expected value of f if an event occurs?
- What is the expected value of f if the event does not occur?

If there are no bandwidth limitations and there is perfect time synchronization between the sensors, we will have

$$\begin{aligned} E\langle f|E\rangle &= p \\ E\langle f|E^c\rangle &= q. \end{aligned} \tag{5}$$

The detection rate will be smaller and the false positive rate will be higher if there are bandwidth limits and synchronization is not perfect if we threshold the detection function f at the value p.

A. Detection Rate under Synchronization Error

Let us consider the detection system when the different sensors have different detection probabilities p_i and different

false positive probabilities q_i. The detection function can be expressed as

$$\begin{aligned} f &= \frac{1}{N}\sum_{i=1}^{M} d_i \\ d_i &= \frac{1}{M}\sum_{k=1}^{M} d_{i,k} \\ d_{i,k} &= \begin{cases} 1 & s_{i,k} > T_i \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{6}$$

We can set thresholds using standard Bayesian methods if we know the distributions of noise in the different sensors along with the signal levels produced by the events. To consider the effect of synchronization error on the detector, let us assume that it follows a uniform distribution over the interval $[0,t_s]$ or almost equivalently, $[0,L]$, where $L = \lfloor t_s f_s \rfloor$, and that this distribution is identical for all the sensors. Now the probability that the sample of length MN from which f is computed was not produced in the interval $[t-\tau,t]$, or in the sampled case, $[k-M,k]$ can be calculated. We examine the most probable case that the event occurs in the interval $[k-M,k]$, and there is no event in the preceding interval. We assume that all detection results from individual sensors are transmitted to the node which produces f–we later generalize to the case where some of the bits are lost in transmission. Consider a sample received at time l in the interval $[k-M,k]$. The probability that it came from a sensor before the event started is

$$p_{se,l} = \begin{cases} \frac{L-(l-k+M)}{L} & L-M+k-l \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

and the probability that it came from during the event is $1-p_{se,l}$. Without loss of generality, we can take $k = M$ or any multiple of it, so that l goes from 0 to M. Hence, the expected value of the composite detector f can be calculated from the individual values of the $d_{i,l}$ as follows:

$$\begin{aligned} E\langle f\rangle &= \frac{1}{N}\sum_{i=1}^{N}\langle d_i\rangle \\ \langle d_i\rangle &= \frac{1}{M}\sum_{l=1}^{M} p_i(1-p_{se,l}) + q_i p_{se,l} \\ &= p_i\left(\frac{M+1}{2L}\right) + q_i\left(1-\frac{M+1}{2L}\right) \end{aligned} \tag{8}$$

We can use these calculations in conjunction with the distributions in [10] to obtain realistic detector performance.

V. A Robust Tracking Filter for Bearings Measurements

In this section, we detail the design and performance of a target tracking filter that uses bearings measurements. The Kalman-like hybrid nonlinear filter is stable and robust to synchronization error on the network and asynchronous transmissions of bearings data to the central processor (or any processor). The filter does not need very much more computation than a Kalman filter. We make the following assumptions upon the sensor network:

Assumption 1: Sensors are laid out randomly on the terrain–in particular, their coordinates follow a 2D Gaussian distribution.

Assumption 2: Each sensor's location is known through GPS or prior localization.

Assumption 3: The sensors are omnidirectional, i.e., they have a 360° field of view.

To model general motion of an object on the plane, we use a constant acceleration point mass model for both the x and y coordinates of the object. This permits tracking both the motion of vehicles and human beings or animals with the same filter.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{a} \\ \dot{\mathbf{a}} &= \nu_{2\times 1} \\ \mathbf{x} &= \begin{pmatrix} x \\ y \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} a_x \\ a_y \end{pmatrix} \\ \nu_{2\times 1} &= \begin{pmatrix} \nu_x \\ \nu_y \end{pmatrix}, \quad \nu_x \sim N(0,\sigma_x), \nu_y \sim N(0,\sigma_y) \end{aligned} \tag{9}$$

We rewrite the model in discrete time taking the sampling time T of the sensors into account, stacking up the x and y dynamics on top of each other. From this point, boldface $\mathbf{x}$ will denote the overall state of the tracked object–$(x v_x a_x y v_y a_y)^T$.

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \nu(k) \tag{10}$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_1 \end{pmatrix}, \quad \mathbf{A}_1 = \begin{pmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \tag{11}$$

A. Constructing a Measurement

The sensors output bearing to the tracked object which is also equivalent to measuring the slope $m_i(\mathbf{x}(k)) = \frac{y(k)-y_i}{x(k)-x_i}$ of a line joining the sensor location (x_i,y_i) to the tracked object at the sensing instant. We create a measurement function that consists in the y-intercept of the line joining the sensor to the tracked object.

$$\begin{aligned} z(k) &= y_i - m_i(\mathbf{x}(k))x_i \\ &= y(k) - m_i(\mathbf{x}(k))x(k) = \mathbf{C}(\mathbf{x}(k))\mathbf{x}(k), \end{aligned} \tag{12}$$

where $\mathbf{C}(\mathbf{x}(k)) = (-m_i(\mathbf{x}(k)) \quad 0 \quad 0 \quad 1 \quad 0 \quad 0)$. Before we move to filter construction, we assume the slope measurement follows a normal distribution, which implies a normal distribution for the intercept when the sensor positions are known, which suggests a Kalman-filter structure.

B. Kalman-like Nonlinear Hybrid Filter

We construct our filter using the above measurement. We make assumptions standard in Kalman filtering: the initial condition of the moving object is a random variable uncorrelated to the state covariance $\mathbf{Q}$; the measurement noise and state covariance are uncorrelated. The state covariance is a 6×6

matrix with only two non-zero elements $q_{33} = \sigma_x^2, \quad q_{66} = \sigma_y^2$. The measurement noise covariance has the form

$$\mathbf{R}_i(k) = x_i^2 \mathbf{R}, \tag{13}$$

where (x_i, y_i) is the location of the sensor that sends data at time k. With all of these assumptions, the filter equations for the state estimate $\hat{\mathbf{x}}(k)$, the Kalman gain $\mathbf{K}(k)$, and the state covariance $\Sigma(k)$ can be written as:

$$\hat{\mathbf{x}}(k+1) = \mathbf{A}\hat{\mathbf{x}}(k) + \mathbf{K}(k)\left(y(k) - \mathbf{C}\left(\mathbf{x}(k)\right)\hat{\mathbf{x}}(k)\right) \tag{14}$$

$$\mathbf{K}(k) = \left(\mathbf{A}\Sigma(k)\mathbf{C}\left(\mathbf{x}(k)\right)\right) \times \left(\mathbf{C}\left(\mathbf{x}(k)\right)\Sigma(k)\mathbf{C}\left(\mathbf{x}(k)\right)^T + \mathbf{R}_i(k)\right)^{-1} \tag{15}$$

$$\Sigma(k+1) = \mathbf{A}\Sigma(k)\mathbf{A}^T + \mathbf{Q} - \mathbf{K}(k)\left(\mathbf{C}\left(\mathbf{x}(k)\right)\Sigma(k)\mathbf{C}\left(\mathbf{x}(k)\right)^T + \mathbf{R}_i(k)\right)\mathbf{K}(k)^T \tag{16}$$

VI. Impact of Clock Stability on Duty Cycling, Bandwidth, and Power Consumption

It is generally assumed that using a more stable clock will improve the duty cycling capabilities of an embedded system and saves bandwidth in time synchronization since less frequent resynchronizations are necessary. Dutta showed in [1] that the lower bound of a clock with a stability of $\pm$50ppm is a duty cycle of 0.01% for a scheduled communication MAC protocol. But how much additional power and bandwidth could be saved by employing a more stable clock, considering that a more stable clock consumes more energy?

A. Clock Stability and Duty Cycling

Let's compare two systems, A and B. Both systems have the same sleep power consumption P_S and active power consumption P_A. Each node has a local clock source with stability s_1 and s_2 that consume P_{C1} and P_{C2} respectively. Additionally, we assume that both platforms have the same duty cycle ratio between sleep (T_S) and active time (T_A). However, in order to communicate with their peers, both systems have to set a guard band that is proportional to their local clock source stability $T_{Gx} = 2 \cdot T_S \cdot s_x$, where s_x is the system's local clock stability. This guard band allows the nodes to wake up ahead of time and thus start the synchronization process early to assure that both nodes are awake when the active time starts.

Given these definitions, we can calculate the average power consumption of a system as

$$P_X = \frac{T_S \cdot (P_S + P_{Cx}) + (T_{Gx} + T_A) \cdot (P_A + P_{Cx})}{T_S + T_A + T_{Gx}}. \tag{17}$$

Let's now assume that $s_1 > s_2$ and thus $P_{C1} < P_{C2}$. In order for system B to be more efficient than system A we have to show that

$$P_A > P_B, \tag{18}$$

and thus

$$\frac{T_S \cdot (P_S + P_{C1}) + (T_{G1} + T_A) \cdot (P_A + P_{C1})}{T_S + T_A + T_{G1}} > \frac{T_S \cdot (P_S + P_{C2}) + (T_{G2} + T_A) \cdot (P_A + P_{C2})}{T_S + T_A + T_{G2}}. \tag{19}$$

We now assume that the clock of system A is a cheap low frequency crystal as found in many embedded systems. These crystals consume very little power, and thus $P_{C1} \sim 0$. Using this assumption, Equation 19 can be simplified to

$$P_{C2} < \frac{2 \cdot T_s^2 \cdot (P_A - P_s) \cdot (s_1 - s_2)}{(T_S \cdot (1 + 2s_1) + T_A)(T_S \cdot (1 + 2s_2) + T_A)}, \tag{20}$$

where we replaced T_{G1} and T_{G2} with its respective definition. We can now observe that $DC = T_A/(T_A + T_S)$ is the duty cycle of the system, and if we assume that $s_1, s_2 << 1$, we get

$$P_{C2} < 2 \cdot [1 - DC]^2 \cdot (P_A - P_S) \cdot (s_1 - s_2). \tag{21}$$

Further assuming that $DC << 1$, $P_S \to 0$, and $s_1 >> s_2$ we find that

$$P_{C2} < 2 \cdot P_A \cdot s_1. \tag{22}$$

This shows that in order for system B to be more power efficient than system A, the clock system used in system B has to use less power than the active power consumption multiplied by twice the precision of system A's clock stability. For example, if we assume that $P_A = 1W$, $s_1 = 50ppm$, and $s_2 = 1ppm$ (in order to satisfy $s_1 >> s_2$) then $P_{C2} < 100\mu W$. We currently don't know of a technology that can achieve a clock stability of $1ppm$ with a power budget of less than $100\mu W$. The closest we could find on the current market is the MAXIM DS32BC35 [7] which is an RTC with integrated 32kHz TCXO, and consumes about 600mW for a stability of $\pm 3.5ppm$. However, we also ignored the fact that time synchronization protocols, such as FTSP [6], can estimate the current clock drift very accurately and thus, as long as the temperature doesn't change too quickly, can improve the duty cycling of an embedded system without using a higher stability clock source.

B. Bandwidth Benefits

It is trivial to see that a time synchronization protocol based on a less temperature stable clock needs to resynchronize more often, than one based on a more stable clock. But how much is the difference? In the simplest case, which represents an upper bound, the time synchronization algorithm assumes the worst case drift $s_{\max}$ and calculates the necessary resynchronization time interval T_r based on a maximum allowed time synchronization error ε

$$T_r < \frac{\varepsilon}{s_{\max}}. \tag{23}$$

This is a very conservative estimate and doesn't include the fact that a time synchronization protocol can calculate the current clock drift. Additionally, if we have knowledge of how fast temperature changes in the environment where the system is located, then a much better estimate of T_r can be found.

The question arises on what would be the optimal resynchronization interval while still guaranteeing that the synchronization error is smaller than ε? Assuming the system has access to an oracle that tells the system the current synchronization error, and that the system does not compensate for local clock drift, it is clear that a system with a lower drift

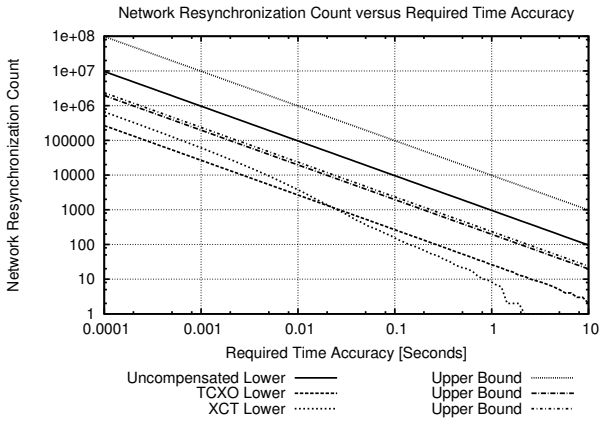

Fig. 1. Upper and lower bounds for the number of resynchronizations necessary to achieve a given time synchronization error. The temperature data comes from a 3 year dataset. XCT is a new compensation technology described in [9].

clock will resynchronize less often than one with a higher drift. Figure 1 illustrates this by using a 3 year temperature dataset collected at the James Natural Wildlife Reserve [4]. It shows the upper and lower bounds by calculating how often a system would have to resynchronize experiencing the temperature changes recorded in the dataset. Looking at one specific example, we can calculate the bandwidth savings a better clock stability can achieve. Let's assume that the application needs a synchronization accuracy of $\varepsilon < 1ms$. Over the 3 year period, an uncompensated clock would have to resynchronize at least 900'000 times using the oracle, whereas a compensated TCXO only 25'000 times. Now, assuming that each resynchronization consists of 2 messages of 16 bytes each results in an average bandwidth of 2.35 bit/s for the uncompensated clock, and only 0.065 bit/s for a compensated clock.

Even though this shows that a compensated clock can lower the number of necessary resynchronizations, it does not mean that a time synchronization system needs to employ a TCXO to achieve this compensation. Maroti et al. showed in [6] that FTSP can estimate the current clock drift below an accuracy of 0.1ppm. Thus, as long as the environment temperature doesn't change too often, drift compensation can be achieved in software to a very high accuracy.

REFERENCES

[1] P. Dutta, D. Culler, and S. Shenker. Procrastination might lead to a longer and more useful life. *HotNets-VI*, 2007.

[2] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts, 2002.

[3] S. Ganeriwal, R. Kumar, and M. Srivastava. Timingsync protocol for sensor networks, 2003.

[4] James Reserve. Mosscam. http://www.jamesreserve.edu/mosscam/, 2007.

[5] B. KUSY, P. DUTTA, P. LEWIS, M. MAR, A. EDECZI, and D. CULLER. Elapsed time on arrival: A simple and versatile primitive for canonical time synchronization services, 2005.

[6] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. The flooding time synchronization protocol. *SenSys '08*, pages 39–49, 2004.

[7] MAXIM. DS32B35, accurate I^2C RTC with integrated TCXO/crystal/FRAM. http://www.maxim-ic.com/quick_view2.cfm/qv_pk/5283, April 2008.

[8] Kay Römer, Philipp Blum, and Lennart Meier. Time synchronization and calibration in wireless sensor networks. In Ivan Stojmenovic, editor, *Handbook of Sensor Networks: Algorithms and Architectures*, pages 199–237. John Wiley and Sons, September 2005.

[9] Thomas Schmid, Zainul Charbiwala, Jonathan Friedman, Young H. Cho, and Mani B. Srivastava. Exploiting manufacturing variations for compensating environment-induced clock drift in time synchronization. *Sigmetrics*, 2008.

[10] Qing Ye, Yuecheng Zhang, and Liang Cheng. A study on the optimal time synchronization accuracy in wireless sensor networks. *Comput. Netw.*, 48(4):549–566, 2005.