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*—New hardware and technologies enable low-power low-cost distributed sensing systems. To realize certain applications such as real-time event detection, target tracking, and system monitoring, time synchronization is essential. The goal of a time synchronization mechanism is to improve timing accuracy with low energy consumption. Upon closer inspection, however, one identifies a trade off between accuracy and energy consumption. Thus, it may not always be desirable to deploy the most accurate time synchronization as it might either drain the batteries prematurely or clog the network with time synchronization messages. The choice of a time synchronization mechanism will depend on the application's requirement of timing accuracy as well as its energy budget. To select appropriate time synchronization parameters, the relationship between an application's performance or QoI (Quality of Information) and time synchronization services needs to be investigated. This paper formalizes this relationship based on an analytical framework using representative applications, namely, event detection and estimation. The analysis shows the impact of timing errors for different event durations, target speeds, number of sensors, and sampling frequencies. The analysis framework can also be used to estimate the maximum synchronization error each application can sustain while still achieving the desired QoI.**

Moreover, time synchronization plays a key role in energy-saving requirements. The intuitive assumption that using higher stability clocks will automatically improve duty cycling performance, and thus decrease power consumption, does not always hold true. In this article, we will present the link between clock stability, impact on duty cycling, and the possible bandwidth savings that can be achieved by using temperature compensated clocks or clock drift estimation techniques.

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) [**?**], Reference Broadcast Synchronization (RBS) [**?**], elapsed time [**?**], etc. In general, the goal of a time synchronization mechanism is to devise a scheme that improves accuracy with lower energy consumption. However, there is clearly a trade off between accuracy and energy consumption. It would not always be desirable to deploy the most accurate time synchronization as it might drain the batteries too fast. The choice of a time synchronization mechanism will depend on the application's demand on timing accuracy as well as the available energy budget. To estimate an application's demand and select a proper time synchronization mechanism, the relationship between application 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 an analytic framework using representative applications, namely, event detection and estimation.

This work was supported by US-UK ITA Grant

Time synchronization may play a key role in energy-saving requirements. For example, accurate time synchronization can improve the efficiency of duty-cycling and thus save energy. However, different clock sources consume different amounts of energy. In general, more temperature stable oscillators consume more energy. Our analysis of clock stability and duty cycling performance will show how much energy a duty-cycled system can gain by employing a higher stability clock, and vice-versa it shows the maximum energy such a clock system is allowed to use, in order not to offset the gained energy. Additionally, the local clock stability is tightly bound to the number of times a time synchronization service has to resynchronize with its peers. We will investigate upper and lower bounds on the number of resynchronization requests using a 3 year temperature data set and modeling crystal drifts.

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 [**?**].

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

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

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. $\text{Drift}_x(t)$ 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 [?].

- **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 [?].

Based on this representation, [?] 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:

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

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

Figure 1 shows the detection probability $E\langle f\rangle$ with time synchronization error bound of 1ms and 1s drawn in solid and dotted line respectively. Here, we assume that p_i = 0.9 and q_i = 0.05 with 10 randomly deployed sensors. The event duration is 1 second for the analysis. In the figure, we can observe three important results. First, the impact of timing error on detection probability is significant. If the timing error is bounded within 1 second, the detection probability is considerably lowered compared to the case where timing error is bounded within 1ms. Thus, reasonable timing error bounds are mandatory for good QoI. Secondly, the application QoI is not linearly improved with sampling rate. It shows a tendency that the detection probability is slightly impaired with $2 \sim 5$ samples in the event duration period ($0.5 \sim 0.2$ sampling frequency with 1 second event duration). Lastly, opposite to our intuition that the detection probability would be improved as the sampling rate increases, the sampling rate or sampling frequency has minimal impact on detection application's QoI. Rather, the limit of timing error plays more critical role in terms of QoI.

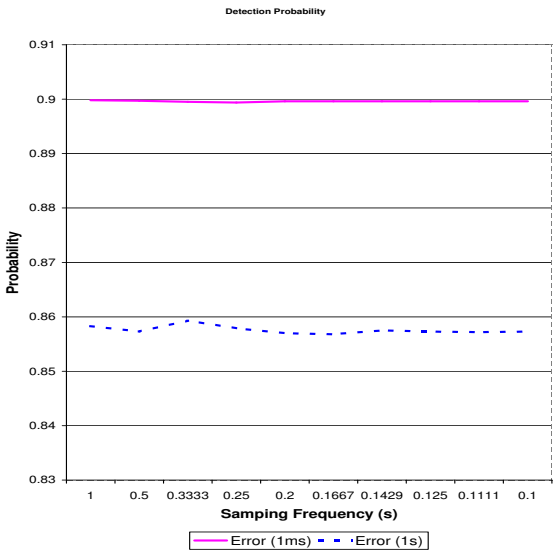

Fig. 1. Detection Probability with various sampling frequencies

B. Dependence of Detection upon Synchronization Error: A More Precise Analysis

The preceding analysis of the dependence of detection accuracy on synchronization error assumes there is no actual synchronization process going on–that the sensors are just left unsynchronized and the clocks drift from each other within a certain bound. This gives conservative results compared to a scheme where there is some synchronization and the error distribution is not uniform but rather, bell shaped. There are two reasons for this–when no time stamps are passed around, measurements that arrive can be simply be taken to be from the past. But when there is a synchronization algorithm that is attempting to synchronize all the clocks, we get measurements with time stamps and we have to concern ourselves with the error in the time stamp to determine detector performance.

A distribution of the time error between two networked nodes, combining process delay, access delay, propagation delay, and receive delay is developed in [?]. Assuming a

uniform distribution of this error, they show that a maximum absolute error of $b > 0$ in the time error results in a maximum synchronization error of $3b$ and a probability distribution of synchronization error over the interval $[-3b, 3b]$ as follows:

$$f_s^b(t) = \begin{cases} \frac{9}{8b} - \frac{9}{8b^2}t + \frac{3}{8b^3}t^2 - \frac{1}{24b^4}t^3, & 2b < t < 3b \\ \frac{11}{24b} - \frac{1}{8b^2}t - \frac{1}{8b^3}t^2 + \frac{1}{24b^4}t^3, & b < t < 2b \\ \frac{5}{12b} - \frac{1}{4b^3}t^2 + \frac{1}{12b^4}t^3, & 0 < t < b \\ \frac{5}{12b} - \frac{1}{4b^3}t^2 - \frac{1}{12b^4}t^3, & -b < t < 0 \\ \frac{11}{24b} + \frac{1}{8b^2}t - \frac{1}{8b^3}t^2 - \frac{1}{24b^4}t^3, & -2b < t < -b \\ \frac{9}{8b} + \frac{9}{8b^2}t + \frac{3}{8b^3}t^2 + \frac{1}{24b^4}t^3, & -3b < t < -2b \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The question we answer with this probability distribution is– what is the probability $p_{in,i}$ that a measurement time stamped as being in an interval came from within it? The answer is obtained through integrating $f_s^b(t)$ from $-t_i$ to $\tau - t_i$ where the duration of the event is τ and t_i is the time already elapsed during the event. The probability that it didn't come from within the interval is simply $p_{out,i} = 1 - p_{in,i}$

$$p_{in,i} = \int_{-t_i}^{\tau - t_i} f_s^b(t)dt. \tag{10}$$

To write the expression for $p_{in,i}$ in a compact form, we first create the notation needed for the integrals of the the probability distribution over the different intervals–we denote them in terms of the interval of integration and the time to which the integration is performed $c_{[ab],t}$denotes the integral of the pdf within the interval to the point t.

$$\begin{aligned} c_{[2b,3b],t} &= \frac{9}{8b}(t-2b) - \frac{9}{16b^2}(t^2-4b^2) \\ &\quad + \frac{1}{8b^3}(t^3-8b^3) - \frac{1}{96b^4}(t^4-16b^4) \qquad (11) \\ c_{[b,2b],t} &= \frac{11}{24b}(t-b) - \frac{1}{16b^2}(t^2-b^2) \\ &\quad - \frac{1}{24b^3}(t^3-b^3) + \frac{1}{96b^4}(t^4-b^4) \\ c_{[0,b],t} &= \frac{5}{12b}t - \frac{1}{12b^3}t^3 \\ &\quad + \frac{1}{48b^4}(t^4) \\ c_{[-b,0],t} &= \frac{5}{12b}(-t) - \frac{1}{12b^3}(-t^3) \\ &\quad + \frac{1}{48b^4}(-t^4) \\ c_{[-2b,-b],t} &= \frac{11}{24b}(-b-t) + \frac{1}{16b^2}(b^2-t^2) \\ &\quad - \frac{1}{24b^3}(-b^3-t^3) - \frac{1}{96b^4}(b^4-t^4) \\ c_{[-3b,-2b],t} &= \frac{9}{8b}(-2b-t) + \frac{9}{16b^2}(4b^2-t^2) \\ &\quad + \frac{1}{8b^3}(-8b^3-t^3) + \frac{1}{96b^4}(16b^4-t^4), \end{aligned}$$

We calculate the probabilities in two parts–integrating over the time ahead of the current time stamp and integrating over the time before the current time stamp to obtain $p_{in,i}^p$ and $p_{in,i}^f$ respectively:

$$p_{in,i}^f = \begin{cases} c_{[2b,3b],3b} + c_{[b,2b],2b} + c_{[0,b],b}, & \tau - t_i > 3b \\ c_{[2b,3b],\tau-t_i} + c_{[b,2b],2b} + c_{[0,b],b}, & 2b < \tau - t_i < 3b \\ c_{[b,2b],\tau-t_i} + c_{[0,b],b}, & b < \tau - t_i < 2b \\ c_{[0,b],\tau-t_i}, & 0 < \tau - t_i < b \end{cases} \tag{12}$$

$$p_{in,i}^p = \begin{cases} c_{[-b,0],-t_i}, & -b < -t_i < 0 \\ c_{[-2b,-b],-t_i} + c_{[-b,0],-b}, & -2b < -t_i < -b \\ c_{[-3b,-2b],-t_i} + c_{[-2b,-b],-2b} + c_{[-b,0],-b}, & -3b < -t_i < -2b \\ c_{[-3b,-2b],-3b} + c_{[-2b,-b],-2b} + c_{[-b,0],-b}, & -t_i < -3b \end{cases} \tag{13}$$

We now have $p_{in,i} = p_{in,i}^p + p_{in,i}^f$ as the sum of the past and future components. We can now use the $p_{in,i}$ and $p_{out,1} = 1 - p_{in,i}$ to calculate the expected value of our detection function from 3:

$$\begin{aligned} E\langle f \rangle &= \frac{1}{N}\sum_{i=1}^{N} E\langle d_i \rangle \\ E\langle d_i \rangle &= \frac{1}{M}\sum_{l=1}^{M} p_i p_{in,l} + q_i(1 - p_{in,l}) \end{aligned} \tag{14}$$

The above expression would help set realistic thresholds for our detector based upon the performance of the synchronization algorithm in the network. The effect of synchronization is significant only if the temporal separation between events is comparable to the sum of event duration τ and maximum synchronization error $3b$. If events are isolated occurrences, correlations constructed between the sensors will yield event timing. However, if we want to go further and localize an event, or estimate any of its parameters in real time, time synchronization again becomes important.

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.

$$\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{v} \\
\dot{\mathbf{v}} &= \mathbf{a} \\
\dot{\mathbf{a}} &= \nu_{2\times 1} \\
\mathbf{x} &= \begin{pmatrix} x \\ y \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} a_x \\ a_y \end{pmatrix} \\
\nu_{2\times 1} &= \begin{pmatrix} \nu_x \\ \nu_y \end{pmatrix}, \quad \nu_x \sim N(0,\sigma_x), \nu_y \sim N(0,\sigma_y) \quad (15)
\end{aligned}$$

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–$(xv_xa_xyv_ya_y)^T$.

$$\begin{aligned}
\mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \nu(k) \quad (16) \\
\mathbf{A} &= \begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_1 \end{pmatrix} \\
\mathbf{A}_1 &= \begin{pmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \quad (17)
\end{aligned}$$

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.

$$\begin{aligned}
z(k) &= y_i - m_i(\mathbf{x}(k))x_i \\
&= y(k) - m_i(\mathbf{x}(k))x(k) = \mathbf{C}(\mathbf{x}(k))\mathbf{x}(k), \quad (18)
\end{aligned}$$

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

$$\mathbf{R}_i(k) = x_i^2\mathbf{R}, \quad (19)$$

where (x_i, y_i) is the location of the sensor that sends data at time k. If there is uncertainty in sensor positions, as when localization is being executed, the corresponding covariances can be added to the $\mathbf{R}_i(k)$. With all of these assumptions, the

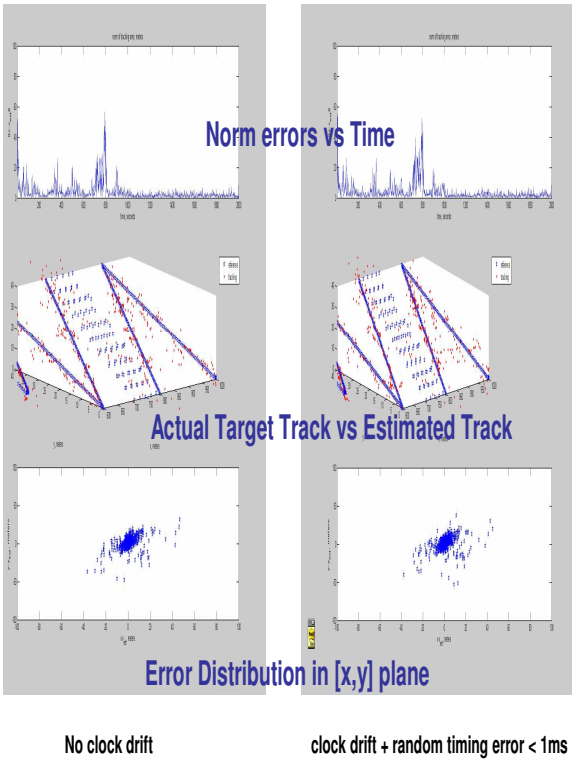

Fig. 2. Target Tracking in a low mobility with timing errors. 10 sensors are randomly placed in 50x50 m^2 field. Each sensor reports the sample to the target every 5 seconds. The target moves with speed less than 10m/s

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:

$$\begin{aligned}
\hat{\mathbf{x}}(k+1) &= \mathbf{A}\hat{\mathbf{x}}(k) + \mathbf{K}(k)\left(y(k) - \mathbf{C}(\mathbf{x}(k))\hat{\mathbf{x}}(k)\right) \quad (20) \\
\mathbf{K}(k) &= (\mathbf{A}\Sigma(k)\mathbf{C}(\mathbf{x}(k))) \\
&\quad \times \left(\mathbf{C}(\mathbf{x}(k))\Sigma(k)\mathbf{C}(\mathbf{x}(k))^T + \mathbf{R}_i(k)\right)^{-1} \quad (21) \\
\Sigma(k+1) &= \mathbf{A}\Sigma(k)\mathbf{A}^T + \mathbf{Q} \\
&\quad -\mathbf{K}(k)\left(\mathbf{C}(\mathbf{x}(k))\Sigma(k)\mathbf{C}(\mathbf{x}(k))^T\right. \\
&\quad \left.+\mathbf{R}_i(k)\right)\mathbf{K}(k)^T \quad (22)
\end{aligned}$$

The figures 2 and 3 demonstrate the performance of target tracking using the proposed filtering algorithm with or without timing errors. Each figure has three sub-figures. First, it shows the normalized error $sqrt(x'-x_t)^2 + (y'-y_t)^2$ (where x' is the estimated x location of the target, x_t is the actual location x of the target) with time. The second graph shows the actual target's position in blue solid line and the estimated positions in (x,y) plane in red dotted line. The last diagram shows $([x'-x_t, y'-y_t])$. The results indicate that the filtering is robust with timing errors less than 1ms. Generally, the synchronization error through networked time synchronization mechanisms can achieve less than a few milliseconds error. With high mobility where target moves faster, the performance degradation with timing error becomes noticeable. Yet, our

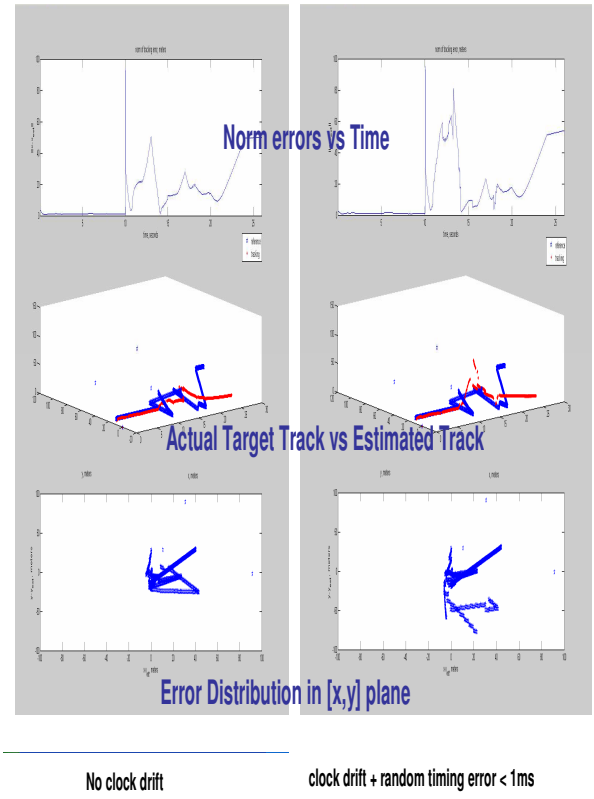

Fig. 3. Target Tracking in a high mobility with timing errors. 10 sensors are randomly placed in 50x50 m^2 field. Each sensor reports the sample to the target every 5 seconds. The target moves with speed of $10 \sim 20$ m/s.

scheme is much more robust to timing errors compared to Least Squares mechanisms shown in Section V-D.

C. Motivation

The motivation for the above nonlinear hybrid filter was to obtain a filter with globally stable performance (rather than the local stability obtained for an extended Kalman filter), robustness to synchronization errors, inexpensive distributable computation (unlike, for example a particle filter), and the ability to handle uncertainty in sensor location directly. Robustness is accomplished through using a filter with a minimum number of parameters–all of which have a physical significance and do not have to be chosen for specific circumstances of installation of the sensor network. through study of the filter's structure, we see that it is instantaneously unobservable, indeed, the observability matrix for the pair $(\mathbf{C}(\mathbf{x}(k)), \mathbf{A})$ is

$$\mathbb{O} = \begin{pmatrix} \mathbf{C}(\mathbf{x}(k)) \\ \mathbf{C}(\mathbf{x}(k))\mathbf{A} \\ \mathbf{C}(\mathbf{x}(k))\mathbf{A}^2 \\ \mathbf{C}(\mathbf{x}(k))\mathbf{A}^3 \\ \mathbf{C}(\mathbf{x}(k))\mathbf{A}^4 \\ \mathbf{C}(\mathbf{x}(k))\mathbf{A}^5 \end{pmatrix} \tag{23}$$

$$= \begin{pmatrix} -m_i & 0 & 0 & 1 & 0 & 0 \\ 0 & -m_i & 0 & 0 & 1 & 0 \\ 0 & 0 & -m_i & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} (), \tag{24}$$

with three non-zero eigenvalues $m_i^2 + 1$. The intuition for the working of this filter comes from Monte-Carlo methods for thresholding extreme phenomena, where there is short term divergence and long term convergence. We can be reasonably sure of being able to prove the convergence of this filter given that it's tracking error converges to zero in simulations over a wide range of sensor placement, number and tracking object speed. The tools envisioned for this are direct analysis of the sort that is performed for Kalman filters [**?**], [**?**], Barrier certificates to show reliable performance [**?**], [**?**], or stochastic hybrid system tools.

D. Sensitivity to Synchronization Error of Batch Least Squares over the Network

This sensitivity analysis applies to least squares or any equivalent recursive method that attains the least squares solution. We perform the analysis for a constant velocity model of the tracked object simply for the purpose of illustration. Suppose that the cartesian coordinates of a moving object are given the equations

$$x(k) = x_0 + v_x kT \tag{25}$$

$$y(k) = y_0 + v_y kT, \tag{26}$$

then we can construct an array of sequential measurements for the least squares estimation of object position and velocity, with the object of extracting x_0, v_x, y_0 and v_y. It would have the following form:

$$\mathbb{A}\mathbf{X} = \mathbb{B} \tag{27}$$

$$\mathbb{A} = \begin{pmatrix} 1 & kT & -m_{k,j} & -m_{k,j}kT \\ 1 & (k+p_1)T & -m_{k+p_1,l} & -m_{k,j}(k+p_1)T \\ \vdots & \vdots & \vdots & \vdots \\ 1 & (k+p_n)T & -m_{k+p_n,n} & -m_{k+p_n,n}(k+p_n)T \end{pmatrix}$$

$$\mathbb{B} = \begin{pmatrix} y_j - m_{k,j}x_j \\ y_j - m_{k,j}x_j \\ \vdots \\ y_n - m_{k+p_n,j}x_n \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} y_0 & v_y & x_0 & v_x \end{pmatrix}^T,$$

where T is the sample period, $j...n$ are sensor labels, (x_j, y_j) is a sensor position on the plane, $m_{k,j}$ is the slope measurement

from sensor j at time kT, k going from $k...k+p_n$. From the structure of this least squares problem, if we assume that there are no errors in sensor locations, the synchronization error comes additively into the matrix $\mathbb{A}$ as

$$\delta\mathbb{A} = \begin{pmatrix} 0 & a_0 & 0 & -m_{k,j}a_0 \\ 0 & a_1 & 0 & -m_{k,j}a_1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_n & 0 & -m_{k+p_n,n}a_n \end{pmatrix} \quad (28)$$

$$= \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 0 & -m_{k,j} \\ 0 & 1 & 0 & -m_{k,j} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & -m_{k+p_n,n} \end{pmatrix} \quad (29)$$

$$= \begin{pmatrix} a_0 & a_1 & \dots & a_n \end{pmatrix} \delta\mathbb{A}' \quad (30)$$

where $a_1,\dots,a_n$ are the timing errors for the different sensors. For a least squares problem, the first order error in estimation of the unknown given the uncertainty in the regression matrix $\mathbb{A}$ is available from standard references [**?**]:

$$\delta\mathbf{X} = \mathbb{A}^\dagger(\delta\mathbb{B} - \delta\mathbb{A}\mathbf{X}) + \left(\mathbb{A}^T\mathbb{A}\right)^{-1}\delta\mathbb{A}^T\mathbf{r}, \quad (31)$$

where $\mathbf{r} = \mathbb{B} - \mathbb{A}\mathbf{X}$, and $\mathbb{A}^\dagger = \left(\mathbb{A}^T\mathbb{A}\right)^{-1}\mathbb{A}^T$ is the More-Penrose inverse of $\mathbb{A}$. If we assume a mean synchronization error of $\bar{a}$ for all of the sensors, we have an estimation error that is directly proportional to it, in the form:

$$\delta\mathbf{X} = -\bar{a}\left(\mathbb{A}^\dagger\left(\delta\mathbb{A}'\mathbf{X}\right) - \left(\mathbb{A}^T\mathbb{A}\right)^{-1}\delta\mathbb{A}'^T\mathbf{r}\right), \quad (32)$$

showing that the expected estimation error (assuming that the residual $\mathbf{r}$ is zero mean) is proportional to the product of the synchronization error and the estimate $\mathbf{X}$. Thus, faster object motion or more distant objects will increase the estimation error induced by synchronization error.

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 [**?**] 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

$$P_X = \frac{T_S\cdot(P_S+P_{Cx}) + (T_{Gx}+T_A)\cdot(P_A+P_{Cx})}{T_S+T_A+T_{Gx}}. \quad (33)$$

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

$$P_A > P_B, \quad (34)$$

and thus

$$\frac{T_S\cdot(P_S+P_{C1}) + (T_{G1}+T_A)\cdot(P_A+P_{C1})}{T_S+T_A+T_{G1}} > \frac{T_S\cdot(P_S+P_{C2}) + (T_{G2}+T_A)\cdot(P_A+P_{C2})}{T_S+T_A+T_{G2}}. \quad (35)$$

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 35 can be simplified to

$$P_{C2} < \frac{2\cdot T_s^2\cdot(P_A-P_s)\cdot(s_1-s_2)}{(T_S\cdot(1+2s_1)+T_A)(T_S\cdot(1+2s_2)+T_A)}, \quad (36)$$

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

$$P_{C2} < 2\cdot[1-DC]^2\cdot(P_A-P_S)\cdot(s_1-s_2). \quad (37)$$

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

$$P_{C2} < 2\cdot P_A\cdot s_1. \quad (38)$$

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 [**?**] 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 [**?**], 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.

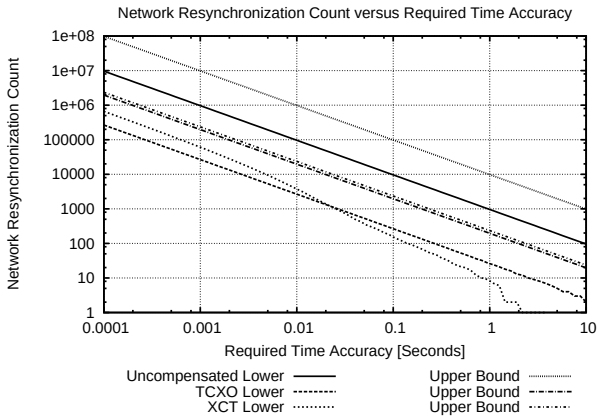

Fig. 4. Upper and lower bounds for the number of resynchronizations necessary to achieve a given time synchronization error. The temperature data comes from a 3 year dataset.

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

$$T_r < \frac{\varepsilon}{s_{\max}}. \tag{39}$$

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 clock will resynchronize less often than one with a higher drift. Figure 4 illustrates this by using a 3 year temperature dataset collected at the James Natural Wildlife Reserve [**?**]. 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 [?] 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.