

Accuracy and Duty-Cycle of FTSP on a LPL-MAC

TIK Report 319

Federico Ferrari, Marco Zimmerling, and Lothar Thiele
ETH Zurich, Switzerland
{ferrari,zimmerling,thiele}@tik.ee.ethz.ch

February 16, 2010

Abstract

Synchronization accuracy and energy efficiency are two of the most important requirements of a synchronization protocol for wireless sensor networks (WSNs). Typically, synchronization protocols for WSNs like FTSP rely on periodic message exchange in order to achieve a sufficient level of accuracy. A higher accuracy therefore comes at the expense of an increased energy consumption. In this paper, we introduce an analytic framework aimed at analyzing how accuracy and energy consumption are affected by the synchronization period and the radio duty cycle, using the example of FTSP running on top of a low-power listening (LPL) MAC protocol. The model exposes the hidden dependencies between protocol parameters and application requirements, which allows for using standard tools to generate feasible as well as optimal protocol configurations.

1 Accuracy Model

1.1 Hardware Clock Model and Assumptions

The local time provided by the HW clock of a sensor node can be expressed as a function $h(t)$ of the real-time t [2]. The HW time is usually the value of a counter that counts intervals of an ideally fixed length. The clock drift $\rho(t)$ is defined as the deviation of the rate with which the counter is actually incremented by the oscillator and the ideal rate 1: $\rho(t) = (dh(t)/dt) - 1$. An ideal clock would therefore have drift $\rho \equiv 0$. We assume $\rho(t) > -1$: a clock never stops and never runs backwards. Due to changes in supply voltage, temperature, and aging, the drift of a real clock fluctuates over time. Different models try to represent this behavior.

The *bounded-drift* model assumes that the drift $\rho(t)$ is bounded: $|\rho(t)| < \hat{\rho}$, $\forall t$. A special case of this model is the *constant-drift* model, where the clock drift $\rho(t) = \rho$ is constant yet still bounded: $|\rho| \leq \hat{\rho}$. Reasonable values for standard sensor nodes equipped with inexpensive oscillators are $\hat{\rho} \in [10, 100]$ ppm¹ [2].

¹ppm: parts per million, *i.e.*, 10^{-6} .

The *bounded-drift-variation* model assumes that the variation of the drift $\vartheta(t) = d\rho(t)/dt = d^2h(t)/dt^2$ over time is bounded: $|\vartheta(t)| < \hat{\vartheta}$, $\forall t$. Assuming a bounded drift variation is very reasonable for most settings, since environmental conditions (*i.e.*, temperature, battery voltage, age of oscillator) change gradually. Furthermore it should be noted that drift estimation assumes a bounded drift variation.

The bounded-drift and the bounded-drift-variation models can be unified into one *combined* model that covers them both. For instance, setting $\hat{\vartheta} = \infty$ in the combined model results in a bounded-drift model. If not otherwise stated, we assume a maximal drift of $\hat{\rho} = 100$ ppm and a maximum drift-variation of $\hat{\vartheta} = 10^{-8} \text{ s}^{-1}$ in the remaining of the paper, which are common values for (inexpensive) clocks equipping sensor nodes.

In FTSP [1], all nodes try to estimate the amount of drift of their HW clock with respect to the reference clock, provided by the root of the network. In order to retrieve information about the drift, nodes periodically send time-stamped messages, *i.e.*, messages containing the synchronized time as estimated by the sender. This period is denoted by T_f . Each node stores the received time-stamps, together with the offset from the local (HW) time at which they are received, in a table of N entries (see Table 1). Time-stamps are usually performed at the radio stack, so that the delay between the time-stamp at the sender and at the receiver is minimized and depends mostly on the transmission time, that can be considered roughly constant. However, delays at the radio stack may always occur, causing random jitters on the time-stamps and thus introducing errors in the drift estimation. Since in our model we are only interested in evaluating the maximum error in the drift estimation as a function of the synchronization period T_f , we assume that there are no delays at the radio stack, *i.e.*, time-stamps are assumed to be perfect.

The maximum error in the drift estimation (ε_{\max}) is thus affected only by non-constant drifts of the HW clocks (maximum drift variation $\hat{\vartheta}$), the length of the FTSP table (N), the synchronization period (T_f), and the maximum hop-distance from the root (H).

$$\varepsilon_{\max}(\hat{\vartheta}, N, T_f, H) = |\text{estimatedDrift} - \text{realDrift}| \quad (1)$$

If all the nodes have a constant drift ($\hat{\vartheta} = 0$), then the estimation is perfect ($\varepsilon_{\max} = 0$): two reference points in the table are already sufficient for performing a perfect linear regression on a linear function.

1.2 1-hop Model: Worst-Case Scenario

The 1-hop model aims at computing the maximum error in the drift estimation ε_{\max} made by a node (node 1) when receiving time-stamped messages from the root node (node 0). Since we are assuming that all time-stamps are perfect, the error in the drift estimation is generated by non constant drifts of the HW clocks. The maximum error ε_{\max} in the drift estimation made by node 1 (node 0 does not perform any drift estimation since it is the root) occurs thus when there is a maximum variation between the drifts of nodes 0 and 1. This happens when one node has $\theta(t) = -\hat{\vartheta}$ and the other node has $\theta(t) = +\hat{\vartheta}$. Since only one of the two nodes performs drift estimation (based on the time-stamps received from the other node), the above situation is equivalent to the one in which node 0 has

a perfect clock with no variation on the drift and node 1 has a drift variation of $-2\hat{\vartheta}$.

It is important to notice that the absolute values of the drifts $\rho_0(t)$ and $\rho_1(t)$ do not influence the worst-case, even if nodes 0 and 1 have different drift values, as long as they are far from the bounds $\pm\hat{\rho}$. This is due to the fact that, under the assumption of perfect time-stamps, FTSP perfectly estimates constant drifts of the clock. The simplest case occurs thus when both node 0 and node 1 have the same initial drift $\rho_0(0) = \rho_1(0) \equiv 0$. Let us also suppose for simplicity that at the beginning the two clocks are perfectly synchronized and that the first time-stamp is both sent by node 0 and received by node 1 at time 0 (hypothesis of perfect time-stamps). The HW times, drifts, and drift variations for nodes 0 and 1 are thus:

$$\begin{aligned} h_0(t) &= t, & h_1(t) &= t - \hat{\vartheta}t^2 \\ \rho_0(t) &= 0, & \rho_1(t) &= -2\hat{\vartheta}t \\ \theta_0(t) &= 0, & \theta_1(t) &= -2\hat{\vartheta}. \end{aligned} \quad (2)$$

1.2.1 Received Time-Stamps and Corresponding Table Content

Since its clock is perfect, node 0 sends a message exactly every T_f time units: the time-stamps (*globalTime*) received by node 1 have values $0, T_f, 2T_f, \dots, (N-1)T_f$. The clock of node 1, however, runs slightly faster due to the fact that its drift varies over time (see (2)). The *localTime* values that node 1 stores in the table differ thus by a certain *offset* from the received time-stamps, as shown in (3) for the generic n -th element, $0 \leq n \leq N-1$.

$$\begin{aligned} \text{globalTime}[n] &= nT_f \\ \text{localTime}[n] &= n(1 - n\hat{\vartheta}T_f)T_f \approx nT_f \\ \text{offset}[n] &= \hat{\vartheta}n^2T_f^2 \end{aligned} \quad (3)$$

The approximation on *localTime*[n] holds when:

$$\hat{\vartheta}T_f(N-1) \ll 1 \quad \Rightarrow \quad T_f \ll \frac{1}{\hat{\vartheta}(N-1)}. \quad (4)$$

With standard values $\hat{\vartheta} = 10^{-8} \text{ s}^{-1}$ and $N = 8$ we have the following constraint on the synchronization period: $T_f \ll 1.4 \cdot 10^7 \text{ s}$. A reasonable (wide) range for T_f is [1 s, 1000 s], so that the approximation in (3) can always be considered valid.

1.2.2 Drift Estimation

Table 1 schematically shows the time-stamps received by node 1 and the corresponding table content; these values are used to calculate conversions between *globalTime* and *localTime* values and finally estimate the drift on the local HW clock. The first step is to compute *localAverage*, *i.e.*, the average of the *localTime* values corresponding to the last N synchronization periods.

$$\text{localAverage} = \text{localTime}[0] + \frac{1}{N} \sum_{n=1}^{N-1} (\text{localTime}[n] - \text{localTime}[0])$$

Received <i>globalTime</i>	Table	
	<i>localTime</i>	<i>offset</i>
0	0	0
T_f	$T_f - \hat{\vartheta}T_f^2 \approx T_f$	$\hat{\vartheta}T_f^2$
$2T_f$	$2T_f - 4\hat{\vartheta}T_f^2 \approx 2T_f$	$4\hat{\vartheta}T_f^2$
$3T_f$	$3T_f - 9\hat{\vartheta}T_f^2 \approx 3T_f$	$9\hat{\vartheta}T_f^2$
$4T_f$	$4T_f - 16\hat{\vartheta}T_f^2 \approx 4T_f$	$16\hat{\vartheta}T_f^2$
...
$(N-1)T_f$	$(N-1)T_f - (N-1)^2\hat{\vartheta}T_f^2 \approx (N-1)T_f$	$\hat{\vartheta}(N-1)^2T_f^2$

Table 1: Time-stamps received by node 1 and corresponding table content.

$$\approx \frac{1}{N} \sum_{n=1}^{N-1} (nT_f) = \frac{T_f}{2}(N-1) \quad (5)$$

$$\text{offsetAverage} = \text{offset}[0] + \frac{1}{N} \sum_{n=1}^{N-1} (\text{offset}[n] - \text{offset}[0]) \quad (6)$$

$$= \frac{1}{N} \sum_{n=1}^{N-1} (\hat{\vartheta}n^2T_f^2) = \hat{\vartheta} \frac{T_f^2}{6} (N-1)(2N-1) \quad (7)$$

$$\text{localSum} = \sum_{n=0}^{N-1} (\text{localTime}[n] - \text{localAverage})^2 \quad (8)$$

$$\approx \sum_{n=0}^{N-1} (nT_f - \frac{T_f}{2}(N-1))^2 \quad (9)$$

$$\approx T_f^2 \sum_{n=0}^{N-1} \left(n - \frac{N-1}{2} \right)^2 \quad (10)$$

$$\equiv T_f^2 \cdot k_1(N) \quad (11)$$

$$\text{offsetSum} = \sum_{n=0}^{N-1} (\text{localTime}[n] - \text{localAverage})(\text{offset}[n] - \text{offsetAverage}) \quad (12)$$

$$\approx \sum_{n=0}^{N-1} (nT_f - \frac{T_f}{2}(N-1))(\hat{\vartheta}n^2T_f^2 - \hat{\vartheta} \frac{T_f^2}{6}(N-1)(2N-1)) \quad (13)$$

$$\approx \hat{\vartheta}T_f^3 \sum_{n=0}^{N-1} \left(n - \frac{N-1}{2} \right) \left(n^2 - \frac{(N-1)(2N-1)}{6} \right) \quad (14)$$

$$\equiv \hat{\vartheta}T_f^3 \cdot k_2(N) \quad (15)$$

$$\frac{k_2(N)}{k_1(N)} = \frac{\sum_{n=0}^{N-1} \left(n - \frac{N-1}{2} \right) \left(n^2 - \frac{(N-1)(2N-1)}{6} \right)}{\sum_{n=0}^{N-1} \left(n - \frac{N-1}{2} \right)^2} \quad (16)$$

$$= \frac{\frac{N}{12}(N-1)^2(N+1)}{\frac{N}{12}(N-1)(N+1)} \quad (17)$$

$$= N - 1 \quad (18)$$

$$estimatedDrift = \frac{offsetSum}{localSum} \approx \frac{\hat{\vartheta}T_f^3 \cdot k_2(N)}{T_f^2 \cdot k_1(N)} = \hat{\vartheta}T_f(N-1) \quad (19)$$

$$2T_f\hat{\vartheta}(N-1) \leq realDrift \leq 2T_f\hat{\vartheta}N \quad (20)$$

$$T_f\hat{\vartheta}(N-1) \leq \varepsilon \leq T_f\hat{\vartheta}(N+1) \quad (21)$$

$$\varepsilon_{\max} = T_f\hat{\vartheta}(N+1) \quad (22)$$

2 Energy model

We use the fraction of time D the radio is on during a synchronization period T_f as a metric of energy consumption. This is reasonable since the radio device typically consumes most of a node's energy, and other activities (*e.g.*, sensing and processing) constitute an energy offset that is roughly the same for all nodes.

Using FTSP each node broadcasts one synchronization message within a synchronization period of length T_f . To ensure that all neighbors can receive the broadcast, nodes transmit for a period that is slightly longer than their sleep interval T_s . Hence, we approximate the fraction of time spent for sending synchronization messages by

$$D_{tx} = \frac{T_s}{T_f}. \quad (23)$$

The rest of the time nodes wake up regularly to poll the channel. Within these active periods of length T_l nodes also hear the short synchronization messages from their neighbors. Assuming that T_l is long enough to accommodate such a message, receiving synchronization messages comes for free. That is, the fraction of time the radio is on besides broadcasting is given by the duty cycle:

$$D_{cycle} = \left(1 - \frac{T_s}{T_f}\right) \frac{T_l}{T_l + T_s}. \quad (24)$$

Thus, the fraction of time D the radio is on during a synchronization period T_f is

$$D = D_{tx} + D_{cycle}, \quad (25)$$

where D_{tx} and D_{cycle} are given by (23) and (24).

References

- [1] M. Maróti et al. The flooding time synchronization protocol. In *SenSys'04*, New York, USA, 2004.
- [2] K. Römer, P. Blum, and L. Meier. *Sensor Networks*, chapter Time Synchronization and Calibration in WSNs. John Wiley & Sons, New York, jul 2005.