Fine-Grained Time Synchronization For Mission-Critical DTSNs

Seng-Yong Lau*, Ling-Jyh Chen†, Yu-Te Huang†, Po-Yen Lin*, Yi-Hsuan Chiang*, Jyh-How Huang‡, Kun-chan Lan§, Hao-hua Chu*, Polly Huang*
*National Taiwan University, Taiwan
†Academia Sinica, Taiwan
‡National Taiwan College of Physical Education, Taiwan
§National Cheng Kung University, Taiwan

***Abstract*—It is generally considered a trivial task to synchronize time in a distributed sensor network having a GPS module on board for each sensor node. This is not quite true depending on the application at hand. Commissioned to implement YuShanNet, a delay-tolerant sensor network (DTSN) enabling encounter information collection for hiker search and rescue, we find that the naive use of standard GPSs time acquisition functionalities gives a synchronization error at the scale of 10-100s milliseconds, which is not sufficient given two application-specific considerations: (1) data goodput and (2) energy efficiency. The two requirements are crucial in that the matter is life or death and the hiking trips usually take days. Towards efficient use of the intermittent connectivity in YuShanNet, we design, implement, and evaluate a time synchronization mechanism that improves the synchronization accuracy by approximately two orders of magnitude. This enables TDMA-fashioned MAC for high-goodput data transmission while the GPS is duty-cycled for energy efficiency.**

I. Introduction

The trail to the highest peak in East Asia, Yushan and a.k.a. Mt. Jade, is extremely popular and attracts visitors from all around the world. Working together with the Yushan National Park to address the hiker safety issue, we set out to design and implement a hiker tracking system, named YushanNet [1]. In the YushanNet system, each hiker carries a hiker node that consists of a GPS receiver, a short-range radio and a limited amount of memory. When hikers pass each other on the trail, the hiker nodes automatically record the encounter information (i.e., the node's ID, the GPS location, and the timestamp) in their memory, and then exchange with the other hiker nodes the encounter information collected previously. When a hiker reaches one of the base-stations installed at frequently visited spots on the trail, the encounter information is uploaded automatically to the base station which is connected to the Internet via a long-range radio, such as a 3G transmitter.

In such a network, the wireless connectivity is opportunistic and the end-to-end data delivery delay can be unpredictably long. We refer to this genre of sensor networks as the delay-tolerant sensor networks (DTSNs). Having examined actual hiker traces before system implementation, we find that (1) hikers tend to move in clusters; and (2) hiker encounters tend to be short. Compounded by the property that the amount of encounter information to exchange scales to the number of hikers, the number of encounters over time, as well as the specific encounter information relay strategy, the wireless channel is likely well contended during the data exchanges. This is the case generally considered better handled by TDMA-fashioned MACs with fine-grained time synchronization.

The GPS's basic global time acquisition function, i.e., the GPGGA messages, produces a millisecond-level synchronization error which is insufficient. We turn to the Pulse Per Second (PPS) signals that provide a very low microsecond-level error [2]. The issue is that it takes a certain amount of time for the PPS signals to stabilize when the GPS module is just waking up. Duty-cycling GPS is essential for YushanNet given that the hiking trips can go as long as a few days and energy efficiency is one of the major system requirements.

To address this problem, we consider both the PPS signals and readings from hiker nodes' local clocks. These two time sources are used to calibrate each other collaboratively. Specifically, when the PPS signals are stable, they are used to correct the local clocks. After tracking the local clocks' drifts for a while, we use the adjusted local clocks to filter the PPS signal outliers as the GPS module comes back from sleep. As a result, we achieve in reducing the time synchronization error to the 10s-microsecond-level while attaining the energy efficiency via duty-cycling the GPS module.

978-1-4673-1786-3/12/$31.00 © 2012 IEEE

II. Time Synchronization By Collaborative Calibration

The basic idea is to exploit the accuracy of PPS signals in estimating the clock drift on the hardware, and calibrate the hardware clock continuously when the GPS receiver is turned off. When the GPS receiver is turned on, the calibrated hardware clock is then used to filter outliers of PPS signals and leave accurate PPS signals for re-synchronization. There are two phases in the proposed collaborative calibration mechanism:

Self-Calibration phase: The self-calibration phase continuously measures the clock drift between PPS signals and the hardware clock wherever a valid PPS signal is available. There are two steps in this phase:

Initialization: The initialization step ensures that accurate PPS signals are used to calculate the initial clock drift of the hardware, i.e., Drift Per Second (DPS). More precisely, when a hardware module is powered on and its GPS receiver achieves a 3D-fix, we capture N contiguous PPS signals and calculate the time offsets between the N signals and the hardware clock.

We let Δt_i be the time offset in ticks between the i-th PPS signal and the hardware clock, and calculate the histogram of the Δt_i values ($1 \leq i \leq N$). We pick the most frequent value from the N time offsets, and expand both upward and downward to include X of the N time offsets. For simplicity, we set $X = 90\%$ in this study. Then, we calculate the initial

time skew of the hardware module (i.e., DPS_0) by

$$DPS_0 = \frac{\sum_{i=1}^{N} \delta_i \Delta t_i}{\sum_{i=1}^{N} \delta_i}; \quad (1)$$

where $\delta_i = 1$ when Δt_i is included in the chosen X of the N time offsets, and $\delta_i = 0$ otherwise. DPS_0 is then used to compensate the time skew of the hardware module every second.

Update: This step updates the clock drift estimate continuously in order to respond to the varying clock drifts caused by the temperature and other environment factors. We let DPS'_i be the time drift measured at time i, which is the time offset between the new valid PPS signal and the hardware clock at time i divided by the elapse time between two consecutive valid PPS signals. We then update the clock drift estimate to obtain DPS_i by

$$DPS_i = (1 - \alpha) * DPS_{i-1} + \alpha * DPS'_i; \quad (2)$$

where α is a weighting coefficient between [0,1]. The value of α will be determined based on actual measurements.

PPS Filter phase: Noises in PPS signals are inevitable when a GPS receiver is operated in the duty-cycled manner. The PPS filter phase identifies PPS outliers and disregards them before the valid ones can be passed on to the Self-Calibration phase for clock drift estimation. Let Δt be the clock drift between two consecutive PPSs, and the time elapsed be ΔT seconds. We consider the latter PPS signal as an outlier if $\frac{\Delta t}{\Delta T} > \epsilon$, where ϵ is a pre-determined threshold. A small ϵ might result in over-filtering and hinder the tracking of clock drift in the Self-Calibration phase, whereas a large ϵ may include some PPS outliers as valid signals by mistake.

III. Experimentation

GPGGA Experiments. The error of the GPGGA-message-based approach is contributed essentially by the delay in between the time the GPS module obtaining the 3D-fix and the time the microcontroller receiving the corresponding GPGGA message. To measure the delay, we stamp the time the microcontroller receives a GPGGA message and subtract it by the 3D-fix time indicated in the GPGGA message. To ensure that the experiment nodes have an identical clock source, we use a function generator to generate a square wave and feed it into the nodes. We also enforce all nodes to start ticking their timers at the same time.

PPS Experiments. In this set of experiments, we compare the proposed PPS solution to two baseline mechanisms. The *PPS naive* case uses the duty-cycled PPS signals which might be unstable and the *Reference PPS* case uses the continuous and stable PPS signals to calibrate the time. 3 variants of the proposed time synchronization mechanism are implemented. The *PPS Initial* implements the initial clock drift derivation. The *PPS Update* adds the continuous updates of the clock drift and the *Collaborative Calibration* adds further the PPS outlier filtering.

We sample the 3D-fix, PPS, and the clock ticks by the Saleae logic analyzer from all the nodes available and then use the same set of traces to compare the various PPS-based solutions. The synchronization errors are derived from discrete-event simulations. The simulator walks through the events and executes the actions described previously and records the adjusted time of each node per second. After the simulations are completed for all nodes, we calculate the pair-wise differences which give rise to the synchronization errors.

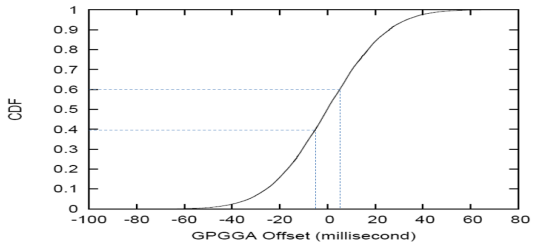

Fig. 1. The CDF of GPGGA-Message Synchronization Errors

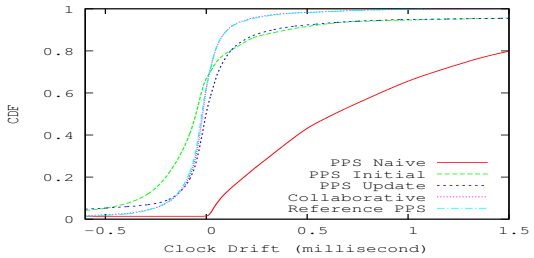

Fig. 2. The result of Δt using 1.PPS naive, align when valid PPS signal acquired 2. PPS self-calibration, 3. Collaborative Calibration

IV. Results

GPGGA-Message Synchronization. Figure 1 plots the CDF of the time drifts among nodes synchronizing via the GPGGA messages. Our experimental result shows that 80% of the time difference is larger than ± 5 ms and the maximum can be as high as 84ms. With this level of accuracy, GPGGA-message-based time synchronization is too coarse-grained for efficient TDMA-fashioned data transmission.

PPS Synchronization. Figure 2 shows the result of the 5 PPS-based solutions. We can see that *PPS naive* perform the worst. By calibrating the local clock during the GPS-off period, *PPS Initial* and *PPS Update* improve synchronization error significantly. By filtering out the noisy PPS signals, *Collaborative Calibration* takes the synchronization accuracy to the next level and the performance is very close to the best case. The 80-percentile synchronization error is ± 0.135 ms and the maximum is 1.924 ms. This is approximately two orders of magnitude better than that of the GPGGA-message-based time synchronization.

V. Conclusion

We design, implement, and evaluate a time synchronization mechanism that improves the synchronization precision by approximately two orders of magnitude. With the success of the time synchronization, we look forward to bring the YuShanNet into realization and promote hiker safety in the Yushan region.

References

[1] Y.-T. Huang, Y.-C. Chen, J.-H. Huang, L.-J. Chen, and P. Huang. Yushannet: A delay-tolerant wireless sensor network for hiker tracking in yushan national park. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM '09. Tenth International Conference on*, pages 379 –380, may 2009.

[2] J. Mannermaa, K. Kalliomaki, T. Mansten, and S. Turunen. Timing performance of various GPS receivers. In *IEEE International Frequency Control Symposium*, 1999.