μs-Scale Time Synchronization For Energy-Constrained 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

Being a regional landmark, ecologically diverse, and indescribably scenic, 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. Despite having limited the number of hikers per day and continuous efforts to promote safety, the extreme elevation and difficult terrain cause a relatively high accident rate – one search-rescue mission per month. 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. In the YushanNet system, each hiker carries a mote-like device that consists of a GPS receiver, a short-range radio and a limited amount of memory. This is referred to as the hiker node. When hikers pass each other on the trail, the hiker nodes, serving as an e-witness, automatically record the encounter information (i.e., the encountered 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. The data exchanged include not only the encounters of the node itself to other nodes, but also the encounters of other hiker nodes through previous exchanges. When a hiker node reaches one of the base-stations installed at frequently visited spots on the trail, the encounter information stored in the node is uploaded automatically to the base station which is connected to the Internet via a long-range radio, such as a 3G transmitter.

978-1-4673-1786-3/12/$31.00

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). DTSNs have been realized in a number of application domains. For example, DieselNet [3], ZebraNet [8], RealityMining [9], and RollerNet [14] are developed to track wild animals, motor vehicles, athletes, and pedestrians respectively. What differentiates our system from previous work is the stringent data requirement and the unique characteristics in hiker node mobility.

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 first technical challenge to address is the time synchronization granularity. Having GPS modules on board, this task appears trivial. We, however, find that GPS's basic global time acquisition function, i.e., the GPGGA messages, produces a 10s-millisecond-level synchronization error. We turn to the Pulse Per Second (PPS) signals, a functionality provided by higher-end GPS modules. In the stable state, these signals are periodic pulses with a very low microsecond-level error [10]. 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. Each of the two timing sources alone does not support sub-millisecond time synchronization. Using the two to calibrate each other collaboratively, however, suffices. More 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. By exploiting the knowledge of what time data is more reliable at a particular time, the proposed solution improves upon the naive GPS global time acquisition method by approximately two orders of magnitude.

Our contribution in this paper is two-fold: (1) a mechanism for microsecond-level time synchronization with energy-efficient use of GPS; and (2) a working reference time synchronization implementation for mission-critical DTSNs. The remainder of the paper is organized as follows. In Section II we detail the motivating application and hiker traces collected in-situ that leads to the need of fine-grained synchronization; In Section III, we describe the three methods, naive GPGAA, naive PPS, and the proposed calibrated PPS for time synchronization. The implementation and the experimental methodologies are described in Section IV and V. Section VI presents a quantitative comparison of the three methods and highlights the benefits of the proposed method. Finally, in Section VII, the paper ends with a close examination of previous work in DTSNs and time synchronization.

II. Motivating Application and Rationales

In this section, we provide in detail the motivating application, the YushanNet system, and the two major design requirements: (1) data goodput and (2) energy efficiency. An analysis of hiker data collected in situ is presented, which gives rise to the need of TDMA-based MAC and in turn the need of fine-grained time synchronization.

A. The YushanNet

Sitting in between the Eurasian and the Philippine Plate, Taiwan is mountainous. In fact, a good two third of this island are covered by mountains. Yushan is the highest of them all. Yushan is Mt. Jade pronounced in Mandarin. The name appeared first in the journal of a scholar in Ching dynasty (1,674 BC). "*Up the high land, above the heavenly cloud, reflecting the rays off the sun, the great white stone shines like a magnificent piece of jade.*" The description is poetic and yet articulate. Yushan, infamous of frequent thunder showers in early afternoon, does glow like the precious stone when the sun reveals itself from the heavy clouds. The air is refreshing and the view is mysteriously beautiful.

Yushan′s main summit, 3,952m above sea level, is the highest peak of East Asia. With dramatic elevation changes as such, the area is blessed with meteorological, biological, and geological diversity. The weather characteristics range from subtropical, temperate, to arctic. The land bears one of the most heterogeneous animals and vegetation habitats. Embedded within the rocks are layers of geological evolution. Last but not the least, among all the trails endeared by the hikers, the national park preserves a famous historical relic - the Patungkuan old passage, constructed 438 years ago.

Being a regional landmark, ecologically diverse, and indescribably scenic, the trail to the highest peak is extremely popular and attracts visitors from all around the world. Despite having limited the number of hikers to 200 per day and continuous efforts to promote safety, the extreme elevation and difficult terrain cause a relatively high accident rate – one search-rescue mission per month.

Fig. 1. The Yushan Peak Trail.

Commissioned by the national park to address the hiker safety issue, we set out to design and implement a hiker tracking system, named YushanNet. In the YushanNet system, each hiker is required to carry a matchbox-size mote-like device 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 motes, serving as an e-witness, automatically record the encounter information (i.e., the encountered node's ID, the GPS location, and the timestamp) in their memory, and then exchange 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 stored in his/her device is uploaded automatically to the base station, which is connected to the Internet via a long-range radio, such as a 3G transmitter.

B. System Requirements

The encounter information collected is critical in that it tells the rescue team where and when each hiker was last seen [7]. By integrating encounter information from multiple hikers during the missing period, the rescue team can estimate the whereabouts of the missing hiker and enable a focused search, as opposed the current approach – querying other hikers and hoping their memories serve them right. On such a matter of life or death and for optimal use of the system, the first system requirement is high-goodput data transmission.

The YushanNet system is deployed on the Yushan Peak Trail, as shown in Figure 1. The trail is 10.9km long from the entrance to the summit and with a 1,302m altitudinal shift, and it usually takes hikers two days to complete the round trip. The Paiyun Lodge, which is about 2.4km from the summit at an elevation of 3,402 meters, provides very basic overnight accommodation for hikers. It is not uncommon to have hikers spending one or two more days on the surrounding trails. Having GPS and RF radio on board, energy efficiency of the hiker mote is the second system requirement.

TABLE I
SUMMARY OF YUSHANNET MOBILITY TRACES.

Date	Total number of hikers
2009-05-28	23
2008-11-20	15
2008-11-21	10
2008-11-22	13
2008-11-23	12

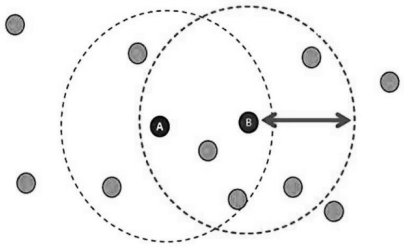

Fig. 2. An encounter between A and B

C. TDMA-like MAC in Favor

To enable high-goodput data transmission, the reservation-based TDMA approach out-weighs its random-access-based CSMA counterpart. This is particularly the case when the hiker mobility shows a distinctive clustering phenomenon where CSMA-like mechanisms are prone to have collision losses.

Table I summarizes the 5 data sets collected from Yushan Peak Trail at the initial stage of the project. In each data set, hikers are recruited to carry commercially available GPS trackers, (Model: Genie GT-311). Unlike the hiker motes, these GPS trackers do not have short-range communication capability for opportunistic data exchange. These GPS trackers are programmed to densely sample and log hikers′ location for the whole day. Figure 2 depicts an encounter of node A and node B. The neighborhood of an encounter is defined as the set of nodes that overhear A′s or B′s radio broadcast. As shown in Figure 3, the size of encounter neighborhood ranges from 2 to 13 nodes. The majority of neighborhood size, number of nodes competing for the wireless access, is in fact higher than 2. Furthermore, we observe that the hiker encounters are typically short. As shown in Figure 4, the median is about 15-40 seconds. A significant amount of encounters are as short as a few seconds.

The LAN that hiker nodes opportunistically form at an encounter is likely large in size and small in channel capacity. 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. Under the circumstance, CSMA-based approaches will result in collision losses and have a hard time meeting the high-goodput data transmission requirement.

D. Time Synchronization for TDMA

TDMA requires that the nodes are synchronized in time. Having a GPS module on board makes the issue seemingly

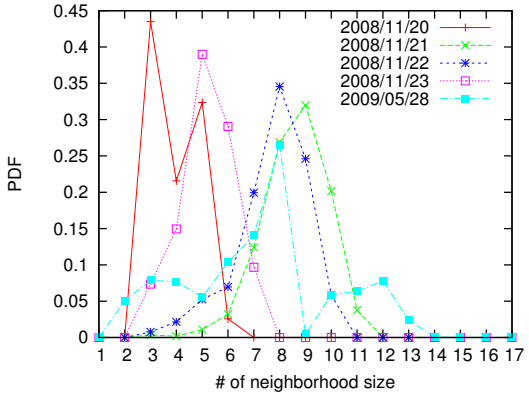

Fig. 3. PDF distribution of encounter neighborhood size

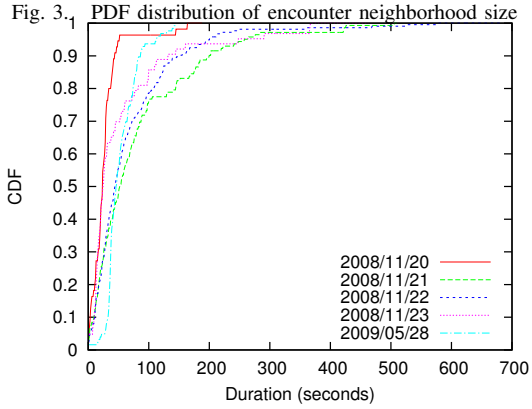

Fig. 4. CDF distribution of encounter duration

trivial, but a closer examination of synchronization error and hiker mobility gives rise to the need of fine-grained time synchronization. Depending on the hardware design, software-based global time query from the GPS module may results in synchronization error that is not suitable for the application at hand. Tested on the GPS module selected for our hiker mote, the time synchronization error ranges from -60 to 60 milliseconds. That is, to avoid packets from 2 adjacent time slots to collide at all, a time slot needs to be at least 120 milliseconds long, whereas sending an encounter packet takes only a few milliseconds. The resulting channel utilization will be too low to provide a high data-goodput service if the time synchronization error is not controlled well below the packet sending time. This motivates the need of a fine-grained time synchronization mechanism for YuShanNet.

III. GPS TIME SYNCHRONIZATION

In this section, we explain the existing GPS-based time synchronization solutions and describe the proposed *collaborative calibration* approach.

A. GPGGA Message-based Time Synchronization

Every time when a GPS receiver is turned on, it acquires the satellite signals and determines the position. With four or more satellite signals in view, a GPS receiver can obtain a 3-dimensional position fix, called 3D-fix. The GPS receiver outputs the location and other information in NMEA 0183

[1] message format. There are different types of the message defined in the NMEA 0183 format, and the most frequently used one is GPGGA, which describes the detailed GPS position information. A sample GPGGA message looks like:

`$GPGGA,123519.876,4807.038,N,01131.000,E,...`

where $GPGGA is the message header, 123519.876 means the position fix is taken at 12:35:19.876 UTC, and the rest are the latitude and longitude of the GPS receiver's location. A first look at the timing information provided by the GPGGA message reveals that the millisecond precision might be insufficient for efficient data transmission.

In addition, it is not clear whether the 3D-fix time on the GPGGA message indicates the beginning or the end of obtaining the position fix. If it is the former, the time required for a GPS receiver to calculate the 3-D location may vary depending on several environmental and hardware factors. If it is the latter, the time required to pass the GPGGA message to the sensor node may also vary depending on the code running on the GPS receiver and the node's microcontroller. In either case, the delay from taking the 3D-fix time to synchronizing the local clock based on the value is non-deterministic, which suggests that the GPGGA-based solution is not appropriate for fine-grained time synchronization.

B. PPS-based Time Synchronization

In addition to using the GPGGA messages for time synchronization, some high-end GPS receivers provide an alternative solution that utilizes the Pulse Per Second (PPS) output. The PPS output is available after a GPS receiver obtains a 3D-fix, and it has been shown that the timing offset of continuous PPS signals between different receivers is less than one microsecond [10].

However, GPS receivers are widely regarded as power-hungry; thus, keeping them "always-on" is unaffordable for mission-critical DTSNs because the lifespan of the latter is constrained by the battery capacity of the associated sensor nodes. To prolong the lifespan of mission-critical DTSNs, GPS receivers must be used in a duty-cycled manner. Specifically, we suppose that a GPS receiver is turned on for T_{on} seconds within a cycle period T_{cycle}. Since the time required for a GPS receiver to acquire a 3D-fix may vary depending on the quality of satellite signal reception, an upper bound of T_{on} is set at $MaxT_{on}$, such that $MaxT_{on} \leq T_{cycle}$; and the GPS receiver is turned off when 1) it obtains a 3D-fix, or 2) the turn-on time reaches $MaxT_{on}$.

With this duty-cycling mechanism, the simplest way to implement time synchronization in DTSNs is to synchronize with accurate PPS signals every time after a 3D-fix is acquired. However, when the GPS receiver is in OFF mode, the local clock will continue to drift away. Thus, the re-synchronization interval is determined by the time between two successful 3D-fixes, which depends on the value of T_{cycle} and the quality of the satellite signal. The interval increases when the value of T_{cycle} increases or the quality of the satellite signal is poor. Moreover, the less stable the hardware clocks used, the higher the time errors between the two synchronizations.

Furthermore, in a number of GPS modules we have tested, the PPS signals received after 3D-fixes are not stable when GPS receivers are just turned on. The amount of time required for the PPS signals to stabilize is also uncertain. To cope with the lack of references in between two successful 3D-fixes issue, as well as the PPS signal stability issue, we propose a *collaborative calibration* solution in the following subsection.

C. 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, as shown in Figure 5.

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, as shown in Figure 6. 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.

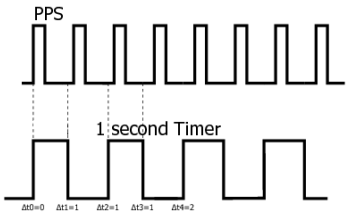

Fig. 5. Illustration of clock drift measurements between PPSs and the hardware clock

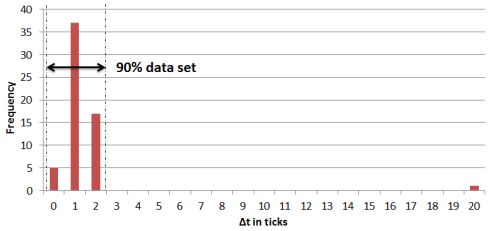

Fig. 6. Illustration of the Initialization step in the Self-Calibration phase ($X = 90\%$)

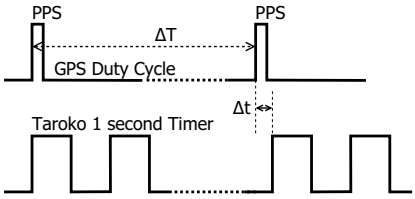

Fig. 7. Illustration of the PPS Filter phase

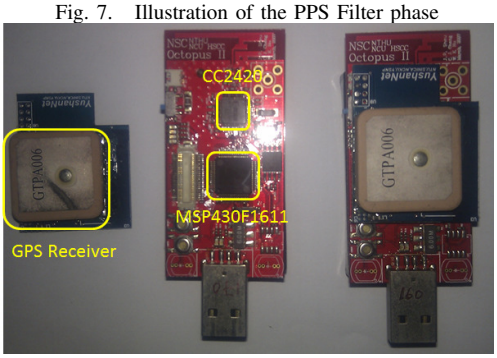

Fig. 8. GPS extansion board and Octopus mote

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, as illustrated in Figure 7. We consider the latter PPS signal as an outlier if $\frac{\Delta t}{\Delta T} > \epsilon$, where ϵ is a predetermined 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.

IV. Implementation

A. Hardware

We implemented our mechanism on a Telos clone, called Octopus [13]. The Octopus mote features a TI MSP430F1611 microcontroller and a TI CC2420 radio compliant with the IEEE 802.15.4 standard. The GPS receiver, GlobalTop FGPMMOPA5, is connected to the Octopus mote via a custom-made extension board, named YushanNet. Displayed in Figure 8, from left to right, are the GPS extension board, the Octopus mote, and the complete prototype with two boards connected as one. Below we detail the relevant specifics of the hardware choices.

Clock on-board the MCU: The clock on the microcontroller is sourced by a 32,768Hz crystal oscillator. The actual running frequency of the crystal oscillator varies depending on the manufacturing process, board design, and a number of other factors which may change over time. The particular oscillator on TI MSP430F1611 may drift 20 ticks faster or slower every one million ticks, denoted 20ppm. The accuracy deteriorates over time as the crystal decays as well. There is an additional aging drift at 5ppm per year. In addition to the manufacturing factors, the PCB board layout, load capacitor, and temperature can all incur significant drifts. The temperature, in particular, may change drastically in a day in the YushanNet environment. To ensure fine-grained time synchronization, tracking of the clock drift is critical.

GPS receiver: Arbitrary GPS receiver will not do to support fine-grained synchronization. Although, the GPS functionalities required for our design are specified in the standard documents, that does not necessarily mean they are well-provided by all GPS receivers manufactured. The manufacturing decision is largely market-driven. Some GPS receivers output only the signals necessary depending on the target applications. The GPS extension board thus went through a few iterations of design. The GlobalTop FGPMMOPA6B GPS receiver was initially selected due to its low price. The module allows querying and returning of the 3D-fix result in NMEA 0183 format. In the second generation, we turn to GPS receivers that output PPS signals. The main board can therefore receive the constant rate pulses from the satellite after a 3D-fix. While having PPS input should enable fine-grained time synchronization, this is still not quite ideal supporting an energy-efficient use of GPS. In the third generation, we finalize on GlobalTop FGPMMOPA5. The module contains another 3D-fix output which signals the main board when the module starts collecting satellite signals for a 3D-fix and signals the main board as well after a 3D-fix is obtained. The functionality saves the main board a few cycles probing the status of the 3D-fix and allows a cleaner and yet more efficient implementation.

B. Software

The time synchronization among nodes are enabled by a timer on the MCU, which is essentially a counter that increments to a preset value at certain frequency. The time to increment is administered by the 32,768Hz crystal oscillator. When counter counts to the preset value, a timeout event is triggered, which is used to indicate the start of a time slot in TDMA. The counter is, in the meantime, reduced to zero and the counting process continues. When the GPS receiver is on and quality PPS signals are available, we align the value of

the counter via PPS signals. The offset is computed to track the clock drift, DPS, as described in III-C.

The timer is also adjusted by the tracked clock drift with or without quality PPS input. Using a 32,768Hz clock source means the minimum unit of time is approximately 30.5 μs. Incrementing the counter by one means an approximately 30.5 μs time elapsed. The clock drift value DPS might be smaller than 30.5 μs which is less than the minimum unit to increment. In this case, we accumulate the DPS value and adjust the counter value when the accumulated time $\geq$ 30.5 μs.

V. Experimentation

Setting the duty cycle of the GPS module to wake up for a 3D-fix every 3 minutes, we conducted two sets of experiments. The first set is to measure the error synchronizing via the GPGGA messages. The second set is comparing the error of the PPS time synchronization with and without the proposed collaborative calibration. The experimental methodologies are detailed in the following.

A. 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.

The precaution we take in this set of experiments is that the clocks on individual nodes are not perfect in synchronization. Instead of attempting to synchronize the local clocks, which is the research problem that leads us here, we provide the nodes in the experiments with a single clock source and enforce all nodes to start ticking its timer at the same time. As shown in Figure 9, a Tektronic AFG3022 function generator is set to generate a 32,768Hz square wave. The square wave is fed into the nodes to stamp the time each GPGGA message is received. A switch is connected to all the nodes. When the switched is pressed, an interrupt will be generated and trigger all the timers to start simultaneously. All the timestamps and GPGGA messages received are logged via the USB port to a PC. The GPS receivers are programmed to output GPGGA messages every second. Each of the four Octopus nodes runs for 2.5 hours which generates 9000 GPGGA-message events. The pair-wise delay differences are calculated which result in 54000 data points to present.

B. PPS Experiments

In this set of experiments, we compare the proposed PPS solution to two baseline mechanisms. The first case works as the worst case where the GPS receiver is duty-cycled and the PPS signals are used naively to calibrate the time on the sensor node. This is referred to as the *PPS naive* case. The second case serves as the best case in which the GPS receiver is always on and the PPS signals are continuous and stable. The *Reference PPS* case estimates the clock drift based on such continuous and stable PPS input and calibrates the

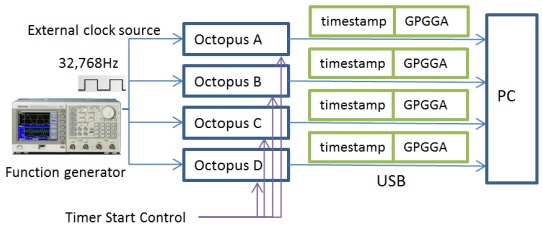

Fig. 9. Setup for GPGGA experiments

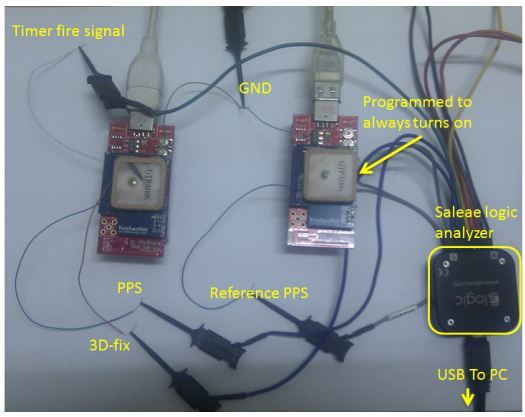

Fig. 10. Setup for PPS experiments

software timer accordingly. For a better understanding of the PPS filtering and continuous self-calibration components in our design, 3 variants of the proposed time synchronization mechanism are implemented. The first variant implements the self-calibration with only the initial clock drift derivation. The second variant adds the continuous updates of the clock drift and the third the PPS outlier filtering. These three variants are referred to as the *PPS Initial*, *PPS Update*, and *Collaborative Calibration*.

We take a trace-driven simulation approach for this part of experimentation. This is due to the error scale of PPS-based solutions being relatively fine. Note that the GPS receiver might not be able to obtain a 3D-fix in cloudy days. It is indeed a concern that the weather effect might overshadow the benefit of design. We, however, would like to focus on the effectiveness of design in this work and leave the satellite signal quality as an issue to pursue in the future. Therefore, as opposed to having nodes running one mechanism at a time and measure the synchronization errors of different variants under different weather conditions, we sample the 3D-fix, PPS, and the clock ticks from all the nodes available and then use the same set of traces to compare the various PPS-based solutions.

Using an always on node's PPS signals as the reference clock, the 3D-fix, PPS, and clock tick signals of 36 nodes are densely sampled at 1MHz. The events are logged by the multi-port Saleae logic analyzer [2]. The length of data collection varies from 2.33 hours to 13.23 hours. Figure 10 shows a simple case of connecting an always-on and a duty-cycled node to the analyzer. The synchronization errors are

derived from discrete-event simulations. For each per-node trace, the 3D-fix, PPS, and clock tick events are sorted by their timestamps. The simulator walks through the events and executes the actions in accordance to the PPS-based solutions described in Section III-C. In the meantime, the simulator records the adjusted time of each node per second. After the simulations are completed for all nodes, we calculate the pair-wise differences that give rise to the synchronization errors to be presented in Section VI-B.

VI. Results

Lastly, we compare and contrast the synchronization accuracy of the GPGGA-message and PPS-based mechanisms. Presented also is an analysis of the parameters chosen to obtain the synchronization accuracy results.

A. GPGGA-Message Synchronization

Figure 11 plots the CDF of the time drifts among nodes synchronizing via the GPGGA messages. Although the timing information embedded in the GPGGA message has a granularity of millisecond, our experimental result shows a much larger offset error. 80% of the time difference is larger than ± 5 ms and the maximum can be as high as 84ms. With this level of accuracy, the GPGGA-based approach is only suitable for sensing event time stamping. It is too coarse-grained for efficient TDMA-fashioned data transmission.

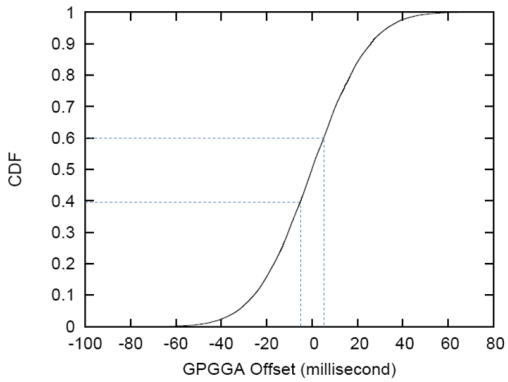

Fig. 11. The CDF of GPGGA-Message Synchronization Errors

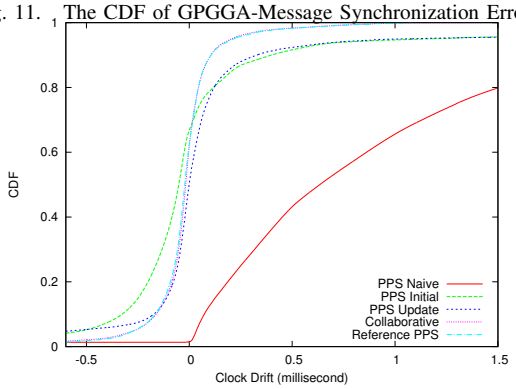

Fig. 12. The result of Δt using different approaches

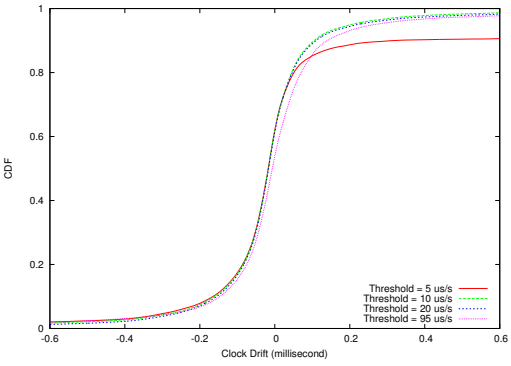

Fig. 13. Result of using different ϵ values to filter PPS outliers.

B. PPS Synchronization

Figure 12 shows the result of the 5 PPS-based solutions. We can see that *PPS naive* perform the worst. The local clocks on the sensor nodes continue to drift away during the time the GPS receiver is off. By calibrating the local clock during the GPS-off period, *PPS Initial* and *PPS Update* improve synchronization error significantly. The situation temperature at the time the experiments are conducted is stable. The clocks on the sensor nodes drift at about the same rate throughout the experiments. We thus do not see the *PPS Initial* vs. *PPS Update* versions being significantly different. 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.

C. Parameters Choices

Two parameters influence the performance of the implementations. First one is the threshold ϵ that is used to filter the PPS outlier, and the second one is the weighting coefficient α in the exponential moving average equation.

Figure 13 shows the performance of the time synchronization with different ϵ settings. PPS outliers will contaminate the calculation and decrease the performance. We want to set a small threshold to effectively filter the outliers, but a small value could possibly filter out the valid PPS. As we can see in Figure 13, 5 μs/s threshold is worse than 10 μs/s threshold setting. In our experiment, a threshold larger than 10 μs/s also decreases the performance, since it increases the chance to accept a PPS outlier. The outlier filter threshold is thus set to 10 μs/s for the *Collaborative Calibration* experiments.

The choice of weighting coefficient α depends on the accuracy of the DPS' estimation in Equation 2. If there is no PPS outlier, setting a large α will produce a better result. If the DPS' is polluted by the PPS outlier, a larger α will result in over-calibration. This is saying the choice of α depends whether PPS outliers are filtered. We run the simulator with different α values and find the best setting for *PPS Update* and *Collaborative Calibration* are 0.05 and 0.85 respectively.

VII. Related Works

Our work builds on prior work in the literature related to Delay Tolerant Sensor Networks (DTSNs) and time synchronization. Several DTSNs have been deployed for a variety of applications. For instance, RollerNet [14] studied a class of DTSNs that follow a pipelined shape. They logged contacts between participants of the roller tour using Bluetooth devices, and introduced a random back-off timer to avoid synchronization of two Bluetooth nodes around the same cycle clock. The Reality Mining project [9] attempted to capture proximity, location, and activity information from 100 subjects at MIT over one year. They utilized the participants' mobile phones to record their proximity with others through periodic Bluetooth scans and location information provided by their phones. ZebraNet [8] is a system designed to investigate the moving patterns of zebras. The UMass DieselNet [3] studied the mobility pattern, AP-to-bus connectivity and bus-to-bus data throughput on a real-world system of 40 buses. The Haggle project [4] studied different data forwarding strategies and explored networking possibilities by utilizing intermittent peer-to-peer connectivity.

NTP [12] is widely used in maintaining time synchronization over the network, and its performance depends on the real-time property of operating systems used on the system. However, NTP is generally considered infeasible for DTSNs due to the non-deterministic transmission time caused by intermittent connectivity in the network. In the Reference Broadcast Synchronization (RBS) [5] protocol, a signal node broadcasts a reference beacon to its neighbors, and the neighboring nodes use the arrival time of this beacon as a point of reference for time synchronization. In the Timing-sync Protocol for Sensor Networks (TPSN) [6], the network is first organized into a tree, and a node is selected as the master node that becomes the root of this tree. Then, the nodes are synchronized with their parents by means of cyclic synchronization. The synchronization consists of several rounds and is initiated by the root, which broadcasts synchronization request to its descendants. Each descendant performs cyclic synchronization with the root. This process is repeated until the tree depth is exhausted. Finally, in the Flooding Time-Synchronization Protocol (FTSP) [11], the node with the least identifier is selected to become the source of standard time and periodically sends synchronizing messages. Each node maintains both a local and a global time, and the global time is synchronized to the local time of the elected node. The clock offset and skew between the local and global clocks are estimated using linear regression. The multiple timestamps on the sender and receiver sides are normalized, statistically processed, and a single timestamp is made. Each node then broadcasts the synchronizing message to their neighbors with the renewed timestamp.

VIII. Conclusion

Commissioned by the Yushan national park, we set out to design and implement a hiker tracking system, named YushanNet. With the stringent application requirements and the nature of the mobility model in the DTSN, a naive and straightforward GPS message time synchronization is insufficient. To fulfill the demands of transmission reliability and energy efficiency, 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] NMEA Format. http://www.gpsinformation.org/dale/nmea.htm.

[2] Saleae PC Logic Analyzer. http://www.saleae.com/home/.

[3] J. Burgess, G. D. Bissias, M. Corner, and B. N. Levine. Surviving Attacks on Disruption-Tolerant Networks without Authentication. In *ACM MobiHoc*, 2007.

[4] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, June 2007.

[5] J. Elson, L. Girod, and D. Estrin. Fine-grained Network Time Synchronization Using Reference Broadcasts. In *OSDI*, 2002.

[6] S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync Protocol for Sensor Networks. In *ACM SenSys*, 2003.

[7] J.-H. Huang, S. Amjad, and S. Mishra. Cenwits: A Sensor-based Loosely Coupled Search and Rescue System Using Witnesses. In *ACM SenSys*, 2005.

[8] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. EnergyEfficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. *ACM Computer Architecture News*, 30(5):96–107, 2002.

[9] A. Lindgren, C. Diot, and J. Scott. Impact of Communication Infrastructure on Forwarding in Pocket Switched Networks. In *ACM SIGCOMM*, 2006.

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

[11] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The Flooding Time Synchronization Protocol. In *ACM SenSys*, 2004.

[12] D. L. Mills. Network Time Protocol Specification, Implementation and Analysis. IETF RFC 1305, March 1992.

[13] J.-P. Sheu, B.-K. Hsu, P.-C. Lin, and C.-J. Chang. Design and Implementation of a Lightweight Operating System for Wireless Sensor Networks. In *International Computer Symposium*, 2006.

[14] P. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. D. de Amorim, and J. Whitbeck. The Accordion Phenomenon: Analysis, Characterization, and Impact on DTN Routing. In *IEEE Infocom*, 2009.