Data-Driven Synchronization for Internet-of-Things Systems

TERRELL R. BENNETT and NICHOLAS GANS, University of Texas at Dallas
ROOZBEH JAFARI, Texas A&M University

The Internet of Things (IoT) is fueled by the growth of sensors, actuators, and services that collect and process raw sensor data. Wearable and environmental sensors will be a major component of the IoT and provide context about people and activities that are occurring. It is imperative that sensors in the IoT are synchronized, which increases the usefulness and value of the sensor data and allows data from multiple sources to be combined and compared. Due to the heterogeneous nature of sensors (e.g., synchronization protocols, communication channels, etc.), synchronization can be difficult. In this article, we present novel techniques for synchronizing data from multi-sensor environments based on the events and interactions measured by the sensors. We present methods to determine which interactions can likely be used for synchronization and methods to improve synchronization by removing erroneous synchronization points. We validate our technique through experiments with wearable and environmental sensors in a laboratory environment. Experiments resulted in median drift error reduction from 66% to 98% for sensors synchronized through physical interactions.

CCS Concepts: • **Networks** → **Sensor networks**; • **Computer systems organization** → *Sensor networks*

Additional Key Words and Phrases: Internet of things, time synchronization, sensor networks

ACM Reference Format:
Terrell R. Bennett, Nicholas Gans, and Roozbeh Jafari. 2017. Data-driven synchronization for Internet-of-Things systems. ACM Trans. Embed. Comput. Syst. 16, 3, Article 69 (April 2017), 24 pages.
DOI: http://dx.doi.org/10.1145/2983627

1. INTRODUCTION

The number of sensors and connected devices in the environment is increasing rapidly, as the Internet of Things (IoT) becomes a reality. We may soon be dealing with trillions of sensors and actuators globally [Lee et al. 2014]. As this multitude of sensors produce data, understanding the relationship between data from multiple sensors and systems is critical to infer the state of the system and to derive valuable information. Sensors generally produce data based on a regular timing interval. Fusing sensor data and other signal processing tasks requires that sensors have an accurate sense of timing

This work was supported in part by the National Science Foundation, under grants CNS-1150079 and CNS-1012975, and the TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.
Authors' addresses: T. R. Bennett and N. Gans, Department of Electrical Engineering, University of Texas at Dallas; 800 W. Campbell Road, Richardson, TX 75080; emails: tbennett@alum.mit.edu, ngans@utdallas.edu; R. Jafari, Department of Biomedical Engineering, Computer Science and Engineering, and Electrical and Computer Engineering, Texas A&M University; 5045 Emerging Technologies Bldg. 3120 TAMU, College Station, TX 77843-3120; email: rjafari@tamu.edu.

DOI: http://dx.doi.org/10.1145/2983627

between them. Properly aligning sensor data in time from multiple sensors requires synchronization.

Many synchronization techniques are based on wireless communication between sensors. Each sensor in the system uses a radio to transmit and receive synchronization messages, and these messages are used to synchronize the clocks. Synchronization of this type requires that all sensors in the system follow the same communication and synchronization protocols to synchronize the entire system. Additionally, long-running, low-power sensors may be power constrained, and wireless communication may not be feasible. These sensors may store data locally or only transmit wirelessly when there is a sufficient power source. Using wireless communication with these sensors could lead to shorter battery life and reduce the usefulness of the sensor. Therefore, real-time synchronization is not an option in systems with these sensors. Other systems depend on an accurate internal real-time clock (RTC) or high-quality oscillators to keep sensors synchronized without wireless communication. In low-power embedded systems, adding an additional chip for an RTC or a higher accuracy oscillator may not be a feasible design decision due to the added cost and power.

There may be instances when a single event triggers a notable measurement in multiple sensors. For example, a person opening a door could be detected by the twisting motion on a wrist-worn wearable as well as an environmental sensor on the door. We call this concept of shared measurements a "coupling." If multiple sensors record or measure the same event, then their data streams are coupled at this global time. This coupling could be physical, as it was in the door/person example, or cyber when data or messages are sent directly between sensors such as a neighbor discovery message. If the local times of the coupled sensors are not the same, then there is some drift (i.e., error) in one (or both) of the sensor clocks, and this error needs to be corrected. One essential step in correction is realigning or shifting the data in time (i.e., synchronizing) to ensure the two data streams are aligned when the shared measurements are acquired and the coupling occurs.

In datasets that have been collected and stored without proper synchronization, some of the valuable information in the data can be lost, such as the ability to determine correlation and possible causation between multiple measurements. Being able to synchronize these datasets offline can greatly increase the value and usefulness of the stored data. In our prior work, we presented an offline method to align and synchronize data [Bennett et al. 2015a]. This technique is applied to data from heterogeneous sensor networks to reduce synchronization errors between sensors. In simple systems, it can be straightforward to determine which sensors had couplings with others. As systems become more complex, determining the correct coupling from many possibilities becomes more difficult. For example, if there is only one person in an apartment and a pill bottle is opened, then we can be certain that the person in the environment opened the bottle. If there are five people in the apartment, then more information is necessary to determine which of the five opened the bottle. Additionally, detecting and ignoring erroneous couplings can help to further improve the synchronization in the system. An erroneous coupling could occur due to choosing the wrong person in the previous example or choosing the correct person but selecting the wrong segment of the relevant data for the coupling.

In this work, we present methods to determine proper couplings in ambiguous scenarios, with an experimental focus on body-worn and environmental inertial measurement unit (IMU) based sensors that include a 3-axis accelerometer and a 3-axis gyroscope. Additionally, we present methods to determine couplings that are erroneous and might decrease the accuracy of the synchronization technique. The primary contribution of this article is our novel method of synchronization for multiple sensors based on physical or cyber couplings between the sensor data streams. This method allows for synchronization of previously collected data streams where synchronization had

not and would not have previously been possible. We present three novel approaches that compose this method. First is synchronization between two sensors, Second is a novel graph model to represent couplings in a multi-sensor system and, third, methods to remove (i.e., prune) couplings that are incorrect or ambiguous from the graph. These three techniques combine to improve sensor synchronization.

We present related works on synchronization in Section 2. This is followed by background on sensor timing and our methods for data-driven synchronization in Section 3. Section 4 covers algorithm updates and techniques for resolving ambiguous couplings as well as outlier rejection. Finally, we present our experiments and analysis in Section 5 and conclusions in Section 6.

2. RELATED WORKS

Due to the importance of synchronizing sensor data, there has been a lot of research in synchronization of wireless sensor networks (WSN). Three prevalent techniques are the timing-sync protocol for sensor networks (TPSN) [Ganeriwal et al. 2003], reference broadcast synchronization (RBS) [Elson et al. 2002], and the flooding time synchronization protocol (FTSP) [Maróti et al. 2004]. TPSN builds a network hierarchy that sends synchronization messages one level at a time. RBS uses a reference broadcast and communication between local receiver nodes to synchronize. FTSP uses media access control (MAC) layer timestamping of a synchronization message and linear regression to reduce the effects of non-deterministic transmit errors in WSNs. These techniques have been used as the basis for other synchronization techniques that focus on energy efficiency, resilience, or message reduction [Guidoni et al. 2010; Huang and Wu 2010; Jain and Sharma 2011; lae Noh and Serpedin 2007; Yildirim and Kantarci 2014b].

Many other WSN synchronization concepts have been presented, based on a variety of ideas and goals. Su and Akyildiz present a technique designed to work well with a changing network topology by electing new master and leader nodes to diffuse synchronization data [Su and Akyildiz 2005]. There are techniques based on gradient synchronization to ensure higher levels of synchronization between neighboring sensor nodes [Pinho et al. 2012; Sommer and Wattenhofer 2009; Yildirim and Kantarci 2014a]. Other researchers focus on techniques that not only synchronize the network locally but also synchronize with external clocks (e.g., coordinated universal time (UTC)) [Swain and Hansdah 2011; Yildirim and Kantarci 2014a; Yildirim and Kantarci 2014b]. Intelligent reduction in synchronization messages and related overhead, while keeping a high synchronization accuracy, is also explored [Lamonaca et al. 2014; Qian et al. 2010]. All of the methods discussed thus far require either one-way or two-way communication between the sensor nodes in the network. Additionally, many of the techniques require a consistently connected network topology for the synchronization. Our technique does not require wireless communication between sensor nodes, but it can take advantage of any communication to assist in the synchronization of data.

There are some synchronization techniques that do not require sensor communication. These techniques use the sensor data to assist in the synchronization of the system. An offline data-driven approach is used to synchronize the data from 100 seismic sensors [Lukac et al. 2009]. These sensors suffered delay due to common faults in environmental systems. This approach relied on regularly occurring events and a model of the event propagation to allow correction of the time information. This technique is limited in that it requires a known regularly occurring seismic event, and the model is based on specific characteristics of the sensor deployment (i.e., *sensor* locations and distances).

Environmental signals are also used to assist in the synchronization of a WSN [Harashima et al. 2012]. Based on the noise-induced synchronization theory, the environmental signals being measured are used as an additive noise that synchronizes the

sensors. Because it is expected that the sensors in a limited range will see the same environmental signals, these signals can be used to aid the synchronization.

Our proposed technique allows for timing synchronization with no communication between the sensors and no synchronization overhead at the sensor. Therefore, heterogeneous low-power sensors can be synchronized in a system based on the events measured by the sensors. Additionally, the proposed technique does not require a fixed sensor topology.

3. DATA-DRIVEN SYNCHRONIZATION

We will provide background on some of the causes of clock inaccuracies. This will be followed by the description of the synchronization algorithms.

3.1. Clock Oscillators

Ideally, all sensor nodes would have access to global system for mobile communication (GSM), global positioning system (GPS), or another high-accuracy common clock. Because of power concerns and operating conditions, this is not always possible. Without a perfect absolute/global clock for each sensor, synchronization techniques are necessary.

The synchronization and timing issues in sensor nodes are due, in part, to the accuracy of hardware oscillators. The stability (i.e., accuracy) of an oscillator can be affected by many factors, including the type of oscillator, the operating temperature, and frequency shifts (i.e., jitter). Oscillator stability is typically measured in parts per million (*ppm*), which is calculated as

$$ppm = \frac{\Delta t}{T} \times 1{,}000{,}000, \tag{1}$$

where Δt is the difference between the actual time passed and the measured time passed (i.e., delay or drift) and T is the total time passed. The accuracy of the oscillators used in sensors typically vary from $\pm$20ppm for a crystal oscillator to $\pm$5,000ppm or higher for digitally controlled oscillators, voltage controlled oscillators, and relaxation oscillators. Oscillators with higher accuracy generally cost more and consume more power [Broman et al. 2013]. To save on the costs and reduce power concerns, sensors may not be designed with the most accurate clock sources. While these tradeoffs enable more long-term, low-cost sensors, they also increase the need for sensor synchronization techniques.

As an example of typical clock drift, Figure 1 shows data from an experiment designed to measure delays between multiple clock sources on a single custom sensor used for our experiments. The sensor has a software-generated real-time clock. The RTC (i.e., local clock) is based on a 32MHz crystal oscillator with $\pm$25ppm stability, which generates a small drift; but the update of this RTC is scheduled by the Contiki operating system (OS) running on the chip. This scheduling delays the RTC update and the restart of the counter and causes errors in the clock. The plot shows the difference between timestamps for the RTC versus the UTC clock that the sensor received from the server.

Considering the UTC time from the server as the most accurate in the system, Figure 1 shows that the relative drift changes as time passes. There is a monotonically increasing drift between sensor clocks based on how long they have been running. In this figure, the drift for the local clock increases at varying rates. This variation is due to the scheduling of the clock update in the OS. Because it is a non-preemptive update, the scheduling of this update can vary. We use the information from this figure to determine drift estimates and build the delay models generated by the expected stability (i.e., *ppm*) for the sensor clock.

The total difference in the time measurements is greater than 36min ($\sim$2,200s) for the local clock after about 10.25h of operation. The effects of this inaccuracy could be amplified in a system that has multiple sensors with low-accuracy clocks and long runtimes.

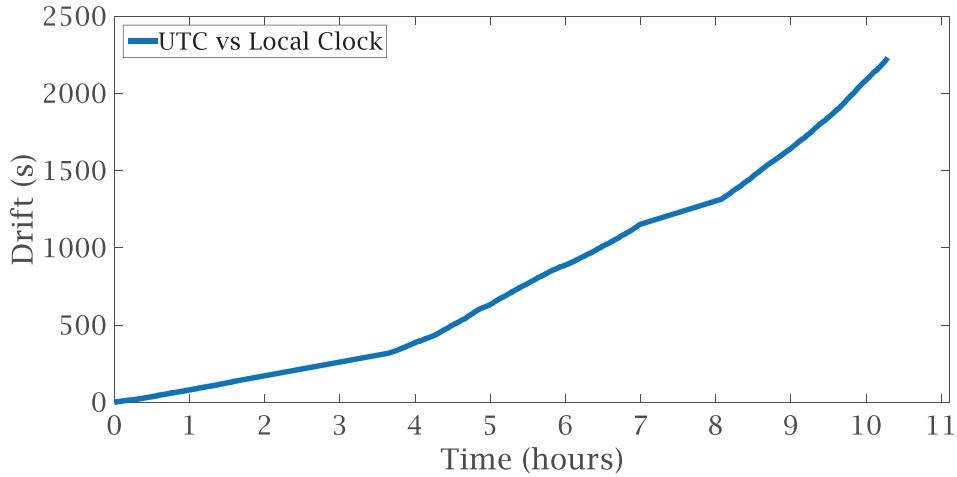

Fig. 1. Clock source differences for over ~10.25h.

Additionally, sensors with high-accuracy clocks can still have frequency fluctuation due the factors mentioned, which further increases the need for synchronization.

One possible solution to the system-based clock error is to design the system with the most accurate clock (i.e., atomic clock) or provide access to GPS or some other reference. However, in a heterogeneous IoT system, these design decisions are not likely to be made by a single party and are likely to differ for all sensors. Furthermore, radio interference and environmental considerations may make GPS synchronization difficult or impossible for some sensors. Another solution could be to timestamp the data when it is received at some central processing node or server. This assumes little to no delay in the reception of all data and that there is no additional delay in the system that handles the timestamps. These assumptions also cannot be guaranteed in a heterogeneous system.

3.2. Synchronization Formulation

In the IoT, there will be a network of sensors providing data. Let the set of all sensors in a network be denoted as

$$\boldsymbol{S} = \{s_1, s_2, \ldots, s_n\}, n \in \mathbb{N} \tag{2}$$

Each of the n sensors in the network generates a data stream consisting of observations that include a data value (i.e., measurement) and a corresponding timestamp. This timestamp could be from a local clock source/oscillator, from an off-chip source (e.g., GPS), or inferred in cases where the sensor outputs at a specific data rate but does not timestamp the data. These observations for the sensor data are described as

$$o_i^n = \{x_i^n, t_i^n\}, i \in \mathbb{N}. \tag{3}$$

where x_i^n and t_i^n are the data value and timestamp at the i^{th} index from sensor s_n, respectively.

When multiple sensors in $\boldsymbol{S}$ experience a physical interaction, shared event measurement, or wireless communication, we can leverage the fact that these couplings occurred to these sensors at the same global clock time, irrespective of the sensor timestamps. We can use these events to synchronize the timing of the sensor data streams by searching the data streams for evidence of couplings that are observed by multiple

sensors. For example, two microphones in a room would both have a response in their data when a person starts to speak. Based on this idea, we define alignment points.

Definition 3.1 (*Alignment Point*). An alignment point is a representation of a physical or cyber event in a sensor data stream that can be accurately distinguished and directly related to the same event in the data stream of another sensor (i.e., coupling).

For example, a sensor measuring the movement of a light switch and a light sensor would have measurements in their respective data streams related to the events of turning on and turning off the lights. For motion sensors, interactions between people and objects can be determined. When sensors are physically coupled through proximity of location, we will search for events in their data streams that can be used as an alignment point. We formally describe alignment between two sensors as

$$o_i^k \equiv o_j^l \text{ where } k \neq l. \tag{4}$$

In an ideal system, the times of the observations would be the same. When they are not the same, a drift exists. We determine the times related to the data points selected to generate a set of alignment points denoted as

$$\boldsymbol{A} = \left\{ \left(o_i^k, o_j^l\right) | \exists i, k, j, l \textit{ sucht that } o_i^k \equiv o_j^l \right\}. \tag{5}$$

Each alignment point in $\boldsymbol{A}$ includes the relevant observations from a pair of sensors. If more than two sensors measure the same event, then each coupling will be listed separately in the set based on sensor pairs. Once alignment points are found in the sensor data streams, the sensor times can be adjusted to correct drift and other timing errors. We previously presented methods to determine alignment points [Bennett et al. 2015a] and briefly review the template- and entropy-based methods for finding physical coupling-based alignment points below.

3.3. Template-Based Alignment Point Selection

If there is a pre-existing synchronized template that records the interaction between two sensors when measuring an event (i.e., a two-signal template), then this two-signal template can be used to find alignment points in the dual-sensor data streams and synchronize the data. Consider a person wearing a sensor on their leg doing a stand-to-sit action onto a chair with a pressure mat. In ideal conditions, the data from the wearable and the pressure mat could be recorded to generate a well-synchronized two-signal template.

We use the two-signal template to find alignment points in other datasets. This is done in two parts. Using data from the first sensor's template (e.g., the stand-to-sit movement), we apply the dynamic time warping (DTW) algorithm [Berndt and Clifford 1994] to the data from the wearable sensor to find all matching instances (i.e., observations) in the sensor data. DTW is used because it is not sensitive to speed variations. With wearable sensors, people may perform actions at different speeds, and this allows a general template to match in these scenarios.

Once the observations are found in the first sensor's data stream, the corresponding instances from the second sensor data stream must be found to form the alignment points. The algorithm used for matching the first sensor's template to the second sensor is covered in Section 3.5.

3.4. Entropy-Based Alignment Point Selection

Entropy (i.e., Shannon entropy) is a measurement of the amount of information in a signal [Cover and Thomas 1991] and is calculated as

$$H\left(X^n\right) = -\sum p\left(x_i^n\right) \log_2 p\left(x_i^n\right), \tag{6}$$

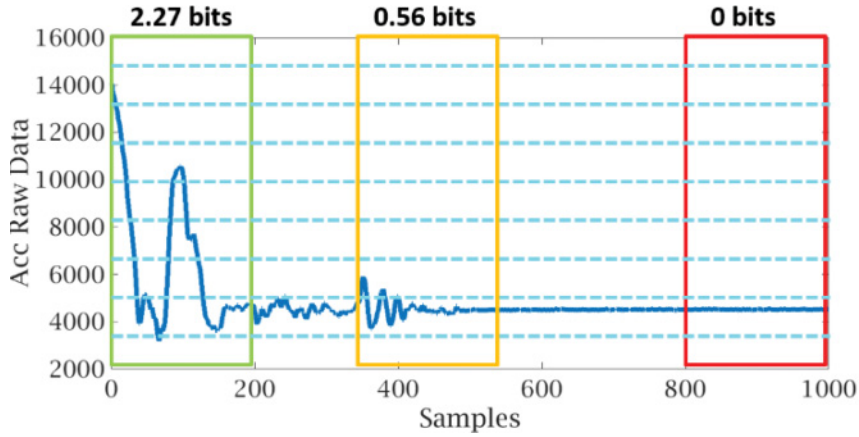

Fig. 2. Entropy calculations for various windows in a data stream.

where $p(x_i^n)$ is the probability of a given x_i^n in the signal's distribution based on a histogram. Using a sliding window across a data stream, the entropy is calculated for each window. The highest entropy values indicate segments of the data stream with the most information or most "interesting" distributions. For example, if a person with a wrist-worn wearable with a motion sensor picks up a coffee mug with a motion sensor, then there will be corresponding high entropy measurements in the data streams of the coffee mug and the wearable sensor.

Figure 2 shows an example of entropy calculations for a data stream using a seven-bin histogram. The horizontal dashed lines show the histogram bin levels. The boxes show three different windows of the data stream. The histogram bins are based on the maximum and minimum values of the entire signal. Note that using histograms reduces the entropy measurements for noisy signals. Noise is inherently high entropy, and the histogram bins dampen this by capturing the high-frequency noise in a single histogram bin and therefore reducing its effect. This assumes that the noise satisfies

$$A_{noise} \times N < A_{signal}, \tag{7}$$

where A_{noise} and A_{signal} are the peak amplitude of the noise and signal, respectively, and N is the number of histogram bins.

Because of this, we see that segments of the signal with a larger range of values will have a higher entropy (e.g., 2.27 bits for the first segment). Segments of the signal with no movement will have a lower entropy (e.g., 0 bits for the third segment). Based on this concept, the entropy peaks are used to select data points in the first sensor's data stream that will be used for matching with a shared event in the second sensor's data stream.

3.5. Mutual Information Matching Algorithm

Once the observation, o_i^k, on the first data stream is selected through the template method or the entropy method, the corresponding observation, o_j^l, on the second data stream must be determined. There are numerous signal matching algorithms that can be used, including cross correlation and earth mover's distance [Ling and Okada 2007]. Through comparison of different approaches, we selected an information theory concept

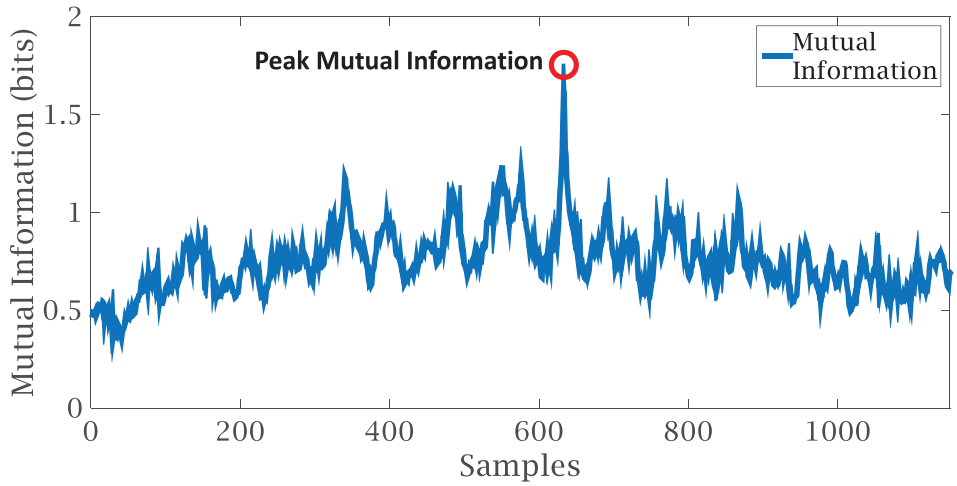

Fig. 3. Matching of template determined by finding the maximum mutual information value.

called mutual information to match the signals. Mutual information is defined as

$$I(X^k; X^l) = \sum p\left(x_i^k, x_j^l\right) \log_2 \frac{p\left(x_i^k, x_j^l\right)}{p\left(x_i^k\right) p\left(x_j^l\right)}, \tag{8}$$

where $p(x_i^k)$ and $p(x_j^l)$ are the probability distributions (estimated by histograms) of signal x_i^k and signal x_j^l, respectively, and $p(x_i^k, x_j^l)$ is the joint probability for signal x_i^k and signal x_j^l.

In the template-based alignment point selection, the mutual information of the two-signal template is calculated against the data segment in the first signal found by DTW and a sliding window from the second sensor's data stream. The peak of mutual information is used to determine the matching data segment on the second data stream. In the entropy-based alignment point selection, the mutual information is calculated between the first data stream segment, o_i^k, and a sliding window on the second data stream. In both cases, the search range on the second sensor data stream is determined as

$$range = \left(t_i^k - t_0^l\right) \times \frac{ppm_l}{1{,}000{,}000}, \tag{9}$$

where t_l^k is the time of the observation in the first data stream, t_0^l is the start time of second sensor, and ppm_l is the stability of the second sensor. Using the time of the observation from the first sensor, the $window = t_l^k \pm range$. Longer runtimes generate larger search windows because the estimated drift grows over time. As with the template-based method, the peak of the mutual information calculation is selected as the corresponding observation, o_j^l, for the second data stream as shown in Figure 3. The two observations together define the alignment point (o_i^k, o_j^l).

A limitation of this method is that the alignment points may have some error due to inaccuracies in the mutual information-based pattern matching algorithm. To account for this possible error, we assign an error estimate, ϵ, to each alignment point based on the expected quality and type of the alignment point. For example, wireless messages shared between sensors (i.e., cyber couplings) have alignment errors measured in tens

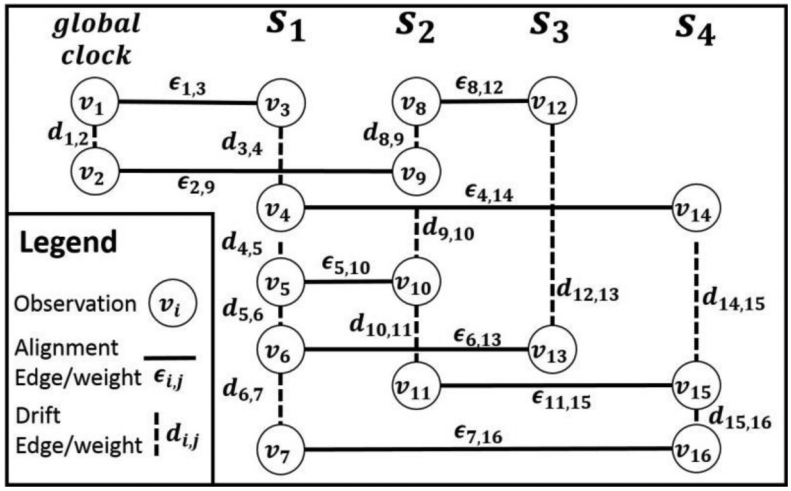

Fig. 4. Graph model for multi-sensor system.

of milliseconds, while alignments found through physical coupling can have drift errors measured in seconds.

4. MULTI-SENSOR MAPPING

We presented a graph-based model of a multi-sensor system in our prior work [Bennett et al. 2015b]. Figure 4 shows an example of the graph model. In this graph, the vertices represent observations from the sensors, the horizontal edges represent the alignments between two observations, and the vertical edges represent the drift between observations on each sensor due to clock instability. The edges have associated weights based on expected drift ($d_{i,j}$ for vertical edges) or expected alignment error ($\epsilon_{i,j}$ for horizontal edges). This model is used, along with a shortest path algorithm (e.g., Dijkstra's algorithm [Dijkstra 1959]), to estimate the best subset of possible alignment points to reduce drift in the system.

As we continue to test with more sensors and more possible interactions, there may be a possible alignment point with ambiguity around the sensors involved. For example, if there are three people in a conference room and all of them get up and leave at approximately the same time with only one person opening the door, then there are three possible couplings to consider when determining which person opened the door. In this section, we discuss the techniques used to determine the most likely couplings in ambiguous scenarios and methods to further improve the synchronization results.

The alignment point selection and matching approaches discussed in Section 3 are the basis for our synchronization algorithm. Consider humans with wrist-worn sensors encountering environmental sensors (e.g., sensors on doors, cups, etc.); we need to determine the alignment points between candidate observation pairs. If we consider every coupling that occurs on an environmental sensor to be equally likely to have been caused by all people in the environment with wearable sensors, then we will have ambiguity and a graph with many couplings that do not actually exist. Starting with this graph of all possible couplings, we must "prune" the graph and remove ambiguous and erroneous couplings. This includes finding the best alignments for known sensor couplings and resolving ambiguity in couplings with uncertainty. With these factors in mind, we use the delay model, the match quality, the concept of logical outliers, and timing error-based outliers to determine the most likely alignment points for

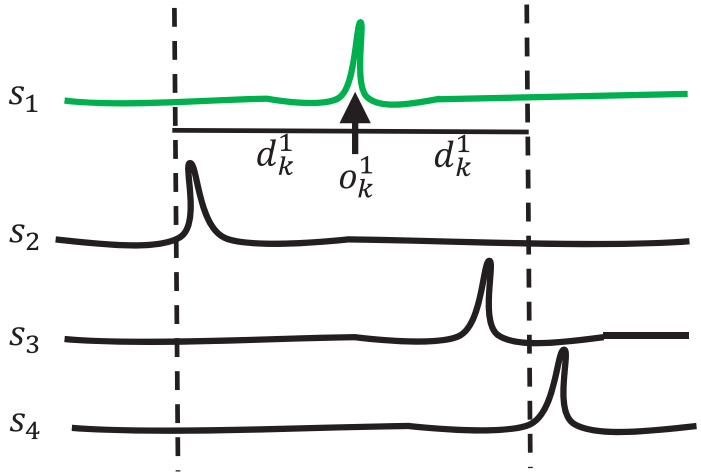

Fig. 5. Example of using the delay model to determine which sensors might have coupling.

synchronization of the stored data and to prune the graph of all possible couplings into an ideal graph of the most likely and least ambiguous couplings as such as the one shown in Figure 4.

4.1. Delay Model Mapping

Based on the stability measure (*ppm*) for a clock and the amount of time that the sensor has been operating, there is an expected drift in the sensor timing. The estimation of this drift for a particular observation, o_i^n, is found by rearranging Equation (1) and is defined as

$$d_i^n = \Delta t = \frac{ppm_n \times T_i^n}{1{,}000{,}000}, \tag{10}$$

where ppm_n is the stability for sensor s_n and T_i^n is how long the sensor has been operating until the observation in question. We are using the delay model based on the possible drift between sensors to determine which sensors are possible candidates for an alignment point with the sensor in question. Based on the expected drift in a sensor, we determine the interval, t_{start}^n to t_{stop}^n, of possible times on the sensor, s_n, as

$$t_{start}^n \leq t_i^n \leq t_{stop}^n, \tag{11}$$

where $t_{start}^n = t_1^n - d_1^n$ and $t_{stop}^n = t_{end}^n + d_{end}^n$. It should be noted that t_{start}^n will be equal to t_1^n because there is assumed to be no drift when the sensor first starts because no time has passed (i.e., $T_1^n = 0$).

Figure 5 illustrates an example of how the drift is used to determine sensors that might have a coupling. In the figure, the data labeled s_1 represent the data from the sensor that will be synchronized. In this example, s_2, s_3, and s_4 have ideal clocks. Using the entropy-based alignment point selection method described in Section 3.4, the event (i.e., observation o_k^1) in the s_1 data can be identified. The drift, d_k^1, is based on the delay model and gives a range on either side of the measured event that could correspond to when the measured event actually occurred. Based on this range, we can see that s_4 does not have a measured event within the range in question. Therefore, we remove this sensor from consideration when looking for couplings with s_1. The figure also shows that there are events measured on s_2 and s_3 that are possibly due to the same event

that is observed on s_1. We must determine which of these events is the same as the event measured on s_1.

4.2. Logical Outliers and Match Quality

There are other considerations that should be taken into account before assuming the sensors with data can be used for coupling. We use the idea of logical outliers to help further reduce ambiguity and reduce the risk of misinterpreting couplings.

Definition 4.1 (*Logical Outlier*). A logical outlier is a coupling/alignment point that, in conjunction with other known information about the system, creates a contradiction based on time, data, or physical proximity.

Logical outliers can be determined based on several different criteria. The first consideration for logical outliers is physical proximity. If information from the system shows that two sensors are in different locations, then there cannot be a physical coupling between these sensors. For example, if a person with a wrist-worn sensor is known to be in a car, then his or her data cannot be coupled with sensors in an office. This information could be determined through received signal strength indicator (RSSI) signals, radio frequency identification (RFID), or a camera that watches the environment. When considering the physical location for a coupling, whether a sensor is in the location where the coupling occurred, r_i^n, can be considered binary and is defined as

$$r_i^n = \begin{cases} 1, & iff\ s_n\ in\ location\ at\ t_i^n \pm d_i^n \\ 0, & otherwise \end{cases}. \tag{12}$$

Another consideration for logical outliers is based on time. The timestamps on the sensors must increase monotonically. If a proposed coupling creates a scenario in which two or more segments on a sensor occur out of chronological order, then a time-based logical outlier has occurred. This is possible if many possible couplings occur within a short time frame and as the search windows increase in size due to drift. These both increase the number of options within a given drift range, which can increase the number of possible errors based on the matching algorithm. All of the observations on a single sensor due to the matching algorithm must remain in chronological order,

$$\forall o_j^n, o_i^n; t_j^n < t_i^n\ where\ i > j. \tag{13}$$

When multiple sensors (i.e., wearable sensors) could have caused a coupling with a single sensor (i.e., environmental sensor), the most likely coupling must be determined to create the alignment point. As described in Section 3.5, we use a matching algorithm based on mutual information to determine alignment points. The maximum peak of the mutual information calculations is used to determine the most likely match between the two segments of sensor data under test. Mutual information increases as information about one random variable tells us more about another random variable. If we expect that two segments are similar, then knowledge about the first segment can give us information about the second segment. The two segments that tell the most about each other are likely to be the most similar and therefore have the highest mutual information.

These peaks represent the quality of a given match between data segments on a pair of sensors (e.g., an environmental sensor versus a wearable sensor). The magnitude of these values can vary based on the data being compared. This means that the mutual information values of correct matches can have a large range. Therefore, we compare the peak mutual information values from each sensor pair to the peak value from the other sensor pairs to find out which wearable sensor was most likely a part of a specific coupling. The maximum peak mutual information value of the possible couplings is

chosen as the most likely coupling. Given the set of mutual information peaks, $\boldsymbol{M_{c_i}}$, for a coupling, c_i, the most likely sensor is determined to be the sensor with the mutual information peak value equal to the maximum mutual information peak. We denote this sensor as s_{c_i}, where

$$s_{c_i} = s_n \; \textit{iff} \; s_n \, \textit{mutual information peak} = \max(\boldsymbol{M_{c_i}}). \tag{14}$$

Because the maximum peak value of mutual information may be very close to other peak values, we also provide a parameter, λ, which allows for a further reduction in the amount of ambiguity in the matches. If other peaks are within λ% of the maximum value, then the peak can be considered ambiguous. It is important to note that the ultimate goal of the algorithms is to synchronize the system and not necessarily to find a match for every coupling. If some possible alignment points are ambiguous, then removing them from consideration may be more beneficial than matching them incorrectly.

4.3. Timing-Based Outlier Detection

If all of the ambiguous couplings have been properly resolved, then we would expect that the alignment points determined are useful for synchronization of the system. Unfortunately, as was mentioned in Section 3, there can be some error in the matching algorithms. These errors can occur due to large movement speed differences between the test data and the templates when using DTW or due to an erroneous data segment being selected in the mutual information matching. Because an incorrect match can greatly reduce the quality of the overall synchronization, it is important to determine which matches are likely to be incorrect and remove them from the synchronization calculations. Because we are looking for outliers, we make two assumptions. The first assumption is that we have at least three alignment points between two sensors, because any two points define a linear model and there would be no possible outliers with two or fewer points. Our second assumption is that our matching algorithms find the correct matches more than 50% of the time. Again, if a majority of the matches found are erroneous, finding outliers will not provide much benefit.

If there are sufficient couplings between two sensors, then we look for outliers in the alignment points. As each alignment point is found, we get the timing information from the observations. Based on the times of the matching segments, we calculate the following parameter for each alignment point:

$$\delta_z^{k,l} = t_i^k - t_j^l, z \in \mathbb{N} \tag{15}$$

where $\delta_z^{k,l}$ is the timing adjustment/correction for each alignment point between sensors s_k and s_l. Because drift increases based on time, the time correction is not enough to determine outliers. Without any synchronization applied, this number would increase monotonically for a clock with poor stability versus an ideal clock. Considering our delay model, we recognize that the drift is approximately linear over time. We also note that the sensors are providing data at fixed rate. Therefore, we normalize the drift correction as

$$\tau_z^{k,l} = \frac{\delta_z^{k,l}}{i}, \tag{16}$$

where $\tau_z^{k,l}$ is the error per sample, and i is the sample number of the observation.

Essentially, $\tau_z^{k,l}$ represents the slope of the delay in the system based on the alignment points. Considering the need to remove possible matching errors (outliers), and the relative linearity of the delay over time, a linear model can be used to detect and reject

outlier alignment points. By considering the per-sample error of the alignment points from the beginning of the dataset, alignment points that do not fit the current model should be rejected as outliers.

The modified z-score is used for timing-based outlier detection [Iglewicz and Hoaglin 1993] and is defined as

$$H_z^{k,l} = \frac{0.6745(\tau_z^{k,l} - \widetilde{\tau^{k,l}})}{MAD} \tag{17}$$

with *MAD* denoting the median absolute difference and $\widetilde{\tau^{k,l}}$ representing the median. As is recommended by Iglewicz and Hoaglin, we use a threshold $\eta = 3.5$ and remove alignment points as outliers when the absolute value of the modified z-score is higher than this threshold. The modified z-score was designed to find outliers in datasets that have too few samples for other outlier detection methods.

4.4. Alignment Point Discovery and Pruning Algorithm

Algorithm 1 shows how all of these tools come together to determine the final set of alignment points that make up the graph that is used for synchronization. We start with the set of observations from an environmental sensor, s_j, selected by the alignment point selection algorithm described in Section 3.4, which we call $\boldsymbol{\Omega^j}$, and the sets of all observations from each of the m wearable sensors, $\boldsymbol{W^1} \ldots \boldsymbol{W^m}$. We then eliminate possible couplings based on the delay model and physical location as described in Sections 4.1 and 4.2, respectively (Algorithm 1: line 4). Possible matches are determined as presented in Section 3.5 (Algorithm 1: line 5), which are filtered based on the match quality as presented in 4.2 (Algorithm 1: lines 9 and 10). These are further reduced by the timing-based outliers discussed in Section 4.3 and the timing logical outliers covered in 4.2 to produce the final set of couplings (Algorithm 1: lines 16 and 17).

ALGORITHM 1: Alignment Point Discovery and Pruning

```
1: for all o_i^j ∈ Ω^j do
2:    for all W do
3:       W_k ← W
4:       if ( t_start^k ≤ t_i^j ≤ t_end^k AND r_i^k = 1) then      // sensor has data at time and in lab
5:          mi_k ← Match_Algorithm (o_i^j, W_k)                   // look for match on wearable k
6:          M_ci + mi_k                                           // add mutual information to set
7:       end
8:    end
9:    s_ci ← s(max(M_ci))                                         // select the largest value from set
10:   if (s_ci − λs_ci still max) then                            // compare max after removing λ
11:      A_k ← (o_i^j, o_l^k)                                     // add alignment point to the set
12:   end
13: for all A_k do
14:   Calculate H_z^k,j                                           // calculate the modified z-score
15:   for all (o_i^j, o_l^k) do
16:      if (t_j^k<t_i^k where i>j OR H_z^k,j > η )                // check for timing based outliers
17:         remove (o_i^j o_l^k)                                  // remove timing based outliers
18:      end
19:   end
20: end
```

Table I. Sensor Types and Capabilities

Sensor Type	Wearable	Fixed Environmental	Mobile Environmental
Collects IMU Data (40Hz)	Yes	No	Yes
Broadcasts Beacons (1Hz)	Yes	Yes	No
Receives Beacons (i.e., RSSI)	Yes	No	Yes
Data Collection/Storage	MicroSD	None	MicroSD
Battery/Wall Plug	Battery	Wall Plug	Battery

5. EXPERIMENTS AND PERFORMANCE EVALUATION

Our experiments take place in a lab environment. Four lab members put on a wrist-worn sensor on their dominant hand when they first arrive in the lab on a given day, and they do not remove it until the end of the day. The lab members carry out their normal daily tasks (e.g., going to class, typing, having meetings, etc.). In addition, there are other sensors throughout the lab. The data collected by these sensors are used for our experiments. The next sections will explain the sensors and how the algorithms discussed work with the data.

5.1. Sensors

The primary sensor used in our experiments is a custom IMU-based sensor. Each sensor has a Texas Instruments CC2538 microprocessor, which handles processing of incoming data, and an InvenSense MPU9150 IMU that measures 3-axis accelerometer and 3-axis gyroscope data. The CC2538 allows the sensors to send and receive beacons over the Zigbee protocol. These beacons allow sensors to calculate an RSSI that we use to approximate location [Kajioka et al. 2014].

The sensors provide a local software-based RTC that is used to timestamp observations made by the sensor (i.e., IMU measurements, RSSI beacons received, UTC messages). These data include IMU readings from the MPU9150, RSSI readings from other sensors, and UTC, which is received from a server every 20s. As each of these observations are made by the sensor, the data are timestamped with the local clock and stored locally on a micro SD card.

Though all of the sensors are based on the same hardware, they are running different firmware based on their purpose in the system. Table I covers the capabilities for the sensors that are set up as wearable sensors, fixed environmental sensors, and mobile environmental sensors. Wearable sensors are affixed to the body of one of the people in the lab and provide information about their activities. The fixed environmental sensors are attached to walls, shelves, and other locations in the lab. These sensors primarily serve as consistent beacons within the lab environment. The mobile environmental sensors are attached to movable items in the lab (e.g., water bottles, chairs, erasers). These sensors can provide context based on the interactions people in the lab have with them.

5.2. Experimental Setup

There are a few other components to the experimental setup beyond the sensors. As mentioned, there is a server that sends out timing information. The server also commands each sensor to send out beacons sequentially to avoid collisions.

Finally, the experiments are recorded by a ceiling mounted camera with a fish-eye lens. This allows the camera to capture a 360×180 degree view of the lab space. The camera data are used to validate interactions observed in the data. Figure 6 shows an example of the lab setup used for these experiments. The camera is centrally located to capture the entire room. The fixed environmental sensors are around the room to

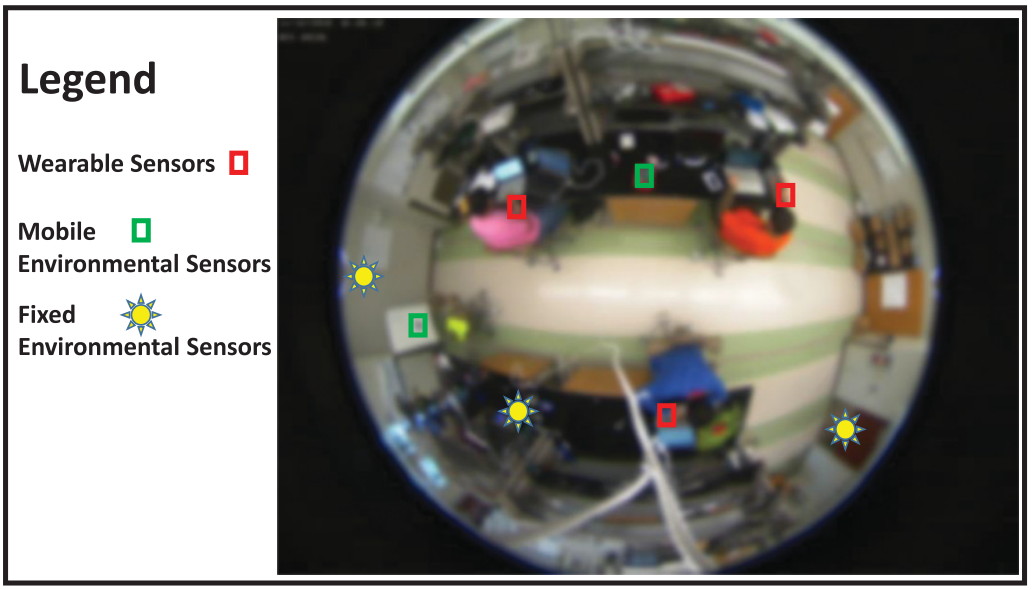

Fig. 6. Example of the lab setup for the experiments.

provide the beacons, and the mobile environmental sensors and people are free to move around the rest of the room.

Typically, there are four people with wearable sensors, five fixed environmental sensors, and two mobile environmental sensors during the experiments. Data are collected from the time the first person arrives in the lab until the last person leaves the lab. The daily data collections generally last between 9 and 10h. All data are collected on microSD cards and processed offline using MATLAB.

5.3. Analysis

In this section, we will present examples from our data collections of how the algorithms are applied and discuss the effects of the updates on overall synchronization. In this dynamic system, sensors enter and exit the environment at different times as the subjects with wearable sensors leave or enter the lab based on their independent schedules (e.g., going to class, going to lunch, etc.). Sensors also start and stop at different times due to the subjects starting and ending their days at different times. The need to recharge or exchange batteries for the mobile environmental and wearable sensors also causes variation and gaps in the collection of sensor data.

Delay model mapping is our first step in selecting possible alignment points and reducing ambiguity. The delay model is estimated based on an expected clock accuracy and knowledge of approximately how long the clock has been running as described in Equation (1). The local clock on the sensor has an estimated stability from 20,000ppm up to 60,000ppm and can vary in this range over time. This error is due in part to the non-preemptive handling of RTC updates on the sensor. Understanding this error and how long a sensor has been operating, we can eliminate some sensors as possible alignment points. Figure 7 shows an example of this. The red dashed line is the acceleration magnitude from mobile environmental sensor 2, which is attached to an eraser. The blue signal is from a wearable sensor, wrist-sensor 2. The vertical green lines show the worst-case expected error based on the delay model. The wrist sensor in question does not have any movement data during the time period in question.

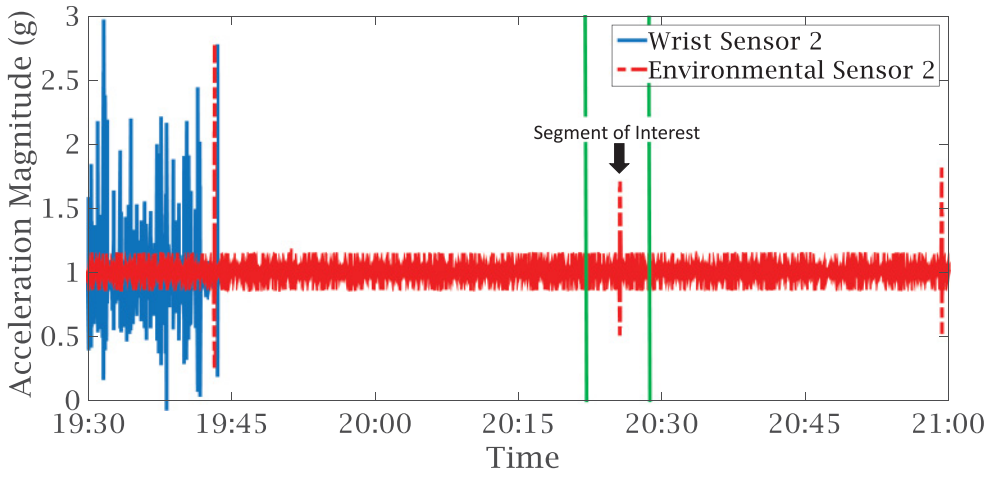

Fig. 7. Example of using the delay model to determine which sensors might have coupling.

Table II. Peak Mutual Information Values for Wrist-Worn Sensors versus a Movable Environmental Sensor

	Wrist Sensor 1 MI Peak	Wrist Sensor 2 MI Peak	Wrist Sensor 3 MI Peak	Wrist Sensor 4 MI Peak
Coupling 1	0.902	0.857	0.559	0.695
Coupling 2	0.812	0.536	0.533	0.593
Coupling 3	0.670	0.611	0.591	0.599
Coupling 4	0.677	0.768	0.739	0.793

In fact, the sensor was powered off during this time and has no data at all. Based on this information, we are able to eliminate (i.e., prune) this wearable sensor from consideration as the sensor that coupled with the environmental sensor. This reduces the amount of possible couplings that must be investigated.

As was discussed in Section 4.2, the match quality is another factor used to remove ambiguity in possible alignment points. The match quality is a comparison of the peak mutual information values for all sensors that may be involved with a coupling. Our movable environmental sensors are static the majority of the time, so we expect the IMU data to have a low magnitude usually. Because these sensors are affixed to inanimate objects, if there is motion on the sensor, we expect that it occurred due to one of the subjects with a wearable sensor. To determine the coupling, we must determine which of the subjects most likely moved the object.

Table II lists four possible interactions/couplings for a moveable environmental sensor. Each coupling could have occurred based on one of the wrist-worn sensors in the environment. The peak mutual information value is calculated for each possible coupling between the environmental sensor and one of the wearable sensors. Each possible coupling is considered independently. In this situation, we can see that the first three couplings likely occurred between the person wearing wrist sensor 1 and the environmental sensor. Coupling 4 appears to have been caused by the person wearing wrist sensor 4. When looking at the match quality measure (i.e., MI peak), we consider how large the maximum mutual information value is relative to the others. There is less certainty about couplings with mutual information values from all wearable sensors that are not distinct. Therefore, we only consider couplings that have mutual information peaks based on $\lambda = 5$. This means the maximum peaks have a margin of at least 5% over other mutual information peaks.

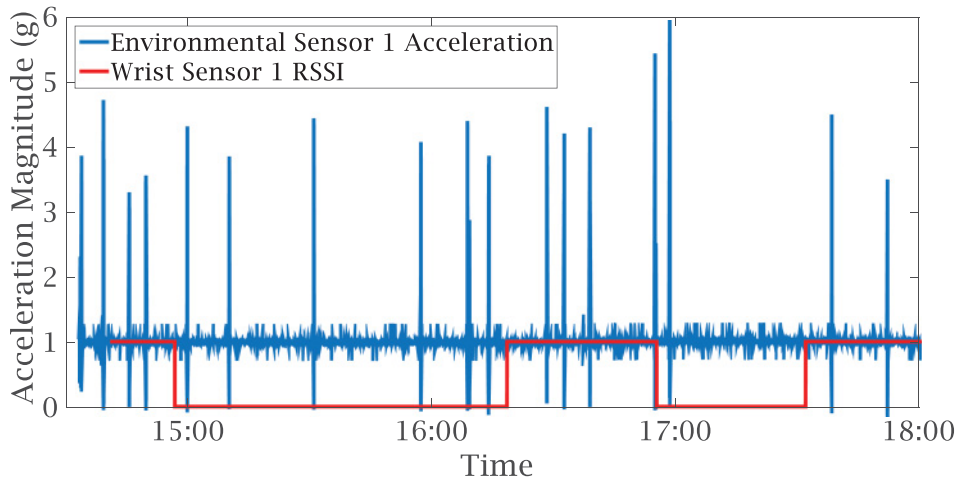

Fig. 8. Example of a logical outlier based on RSSI measurements.

Table III. Sample Error, Modified Z-Score and Outlier Information

$\tau_z^{k,l}$	-0.607×10^{-3}	-0.602×10^{-3}	-0.589×10^{-3}	0.615×10^{-3}
$H_z^{k,l}$	0.883	0.466	0.466	86.09
Outlier?	No	No	No	Yes

Logical outliers can be determined in a number of ways. Physical proximity relative to another sensor can be used to determine if a coupling is logically possible. When considering the peak mutual information scores, we may find false positives. When sensors are in the lab, they receive beacons from other sensors and store this RSSI data. Each time a beacon is received, this is timestamped on the sensor with the local clock. When the sensors are outside of the lab, they do not receive RSSI data. Figure 8 shows IMU data from a mobile environmental sensor as well as r_i^n, defined in Equation (11), which is generated from RSSI data from a wearable sensor. It can be seen that the wearable sensor was not in the lab space between 15:00 and 16:00. Therefore, the couplings that occur during this time cannot be attributed to this wearable sensor. Regardless of any other information (e.g., mutual information) that may suggest that these sensors are coupled, the RSSI information eliminates this option as a logical outlier. This information is used to remove all possible couplings between these two sensors during this time period from consideration for our graph.

It is worth noting that the available information may leave some ambiguity as to which wearable sensor coupled with the environmental sensor. As was previously stated, the ultimate goal is not to determine every possible alignment point but to reduce ambiguity where possible and synchronize the system with the available information.

In Section 4.3, we explained that detecting timing-based outliers requires multiple couplings between two sensors. It is also assumed that the algorithm finds alignment points accurately. If the alignment points all have a variety of error, then timing-based outliers may not be found or quality alignments may be considered outliers. The first row of Table III shows the error-per-sample values calculated for four couplings between two sensors. The second row shows the modified z-score as calculated in Equation (16). The final row labels the outliers. Because the modified z-score for the sample error in the fourth column is greater than the threshold, $\eta = 3.5$, it is considered an outlier.

Table IV. Maximum and Average Drift for Synchronized Data for Day 1

Day 1 ~9.6h collection time. 3 people with wearable sensors				
		Wrist 1	Wrist 2	Wrist 3
Original Data Drift	Median	366s	303s	53s
	Max	694s	1269s	232s
Data Driven Synchronization	Median	5s	54s	18s
	Max	41s	490s	197s

This information does not say that the coupling did not occur but that the correction made does not fit the model and therefore may be a detriment to synchronization.

5.4. Performance

As mentioned, the sensors have a local RTC. This clock is synchronized to the UTC when the sensor is first powered on. After the initial synchronization, the local clock on the sensor runs independently. We use the local clock as our default sensor timing. Our gold standard for synchronization calculations will be based on the UTC clock, as this is the most accurate clock in the system.

Because the environmental sensors are always in the lab, these sensors always had access to the UTC updates from the server when operating. For this reason, we synchronize the environmental sensors through cyber couplings with the UTC. The environmental sensors are then used to synchronize the wearable sensors through physical couplings.

We show the results for the algorithms run on two different days of data. The data collections are generally between 9 and 10h in length. The number of people with wearables sensors varied between 3 and 4. We measure the median and maximum drift for the data. The median drift is calculated based on all samples of IMU data on each sensor

In the following tables, we present the original data drift versus the data driven synchronization algorithm for the wearable for each day. We will present further analysis on the scenarios that generated the synchronization results.

The data collection on Day 1 was over a period of 9.5h. The results for this data collection are shown in Table IV. There was a total of 41 couplings between the three wearable sensors and the two environmental sensors. Wrist 1 had a reduction in median drift of greater than 98%, while Wrist 2 had a reduction of around 82%, and Wrist 3 had a 66% reduction. Of the 41 couplings, 5 couplings were attributed to the incorrect person based on the mutual information peaks. However, all of these errors were discarded through later stages of the algorithm. For example, Figure 9 shows the original clock drift (red line), the synchronized clock drift without outlier rejection (blue line), and the synchronized clock drift with outlier detection (green line) for the Wrist 1 sensor. There were three erroneous alignment points found for this wrist sensor based on mutual information. Using timing-based outliers, they are removed from the calculations, and the drift is greatly reduced.

Wrist 2 did not perform as well as Wrist 1 due to having more alignment points that were inaccurate. The maximum error in the synchronized data was due to a single alignment point. Because this point occurred in the middle of the data, the subsequent alignment points were able to correct this error so it did not carry forward in the data. Wrist 3 saw the smallest reduction in drift for a few reasons. The primary reasons are the amount of couplings and the timing of the couplings. This sensor had the least couplings of the group, and they were relatively early (approximately 3h into the collection). There was no drift at the last coupling, but the algorithm does not correct for

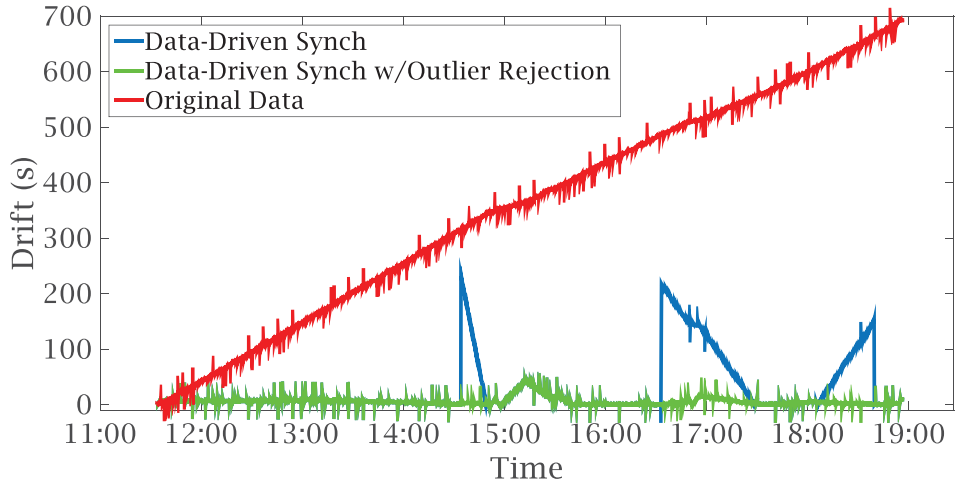

Fig. 9. Original data versus Synchronized data with and without outlier rejection.

Table V. Maximum and Median Drift for Synchronized Data for Day 2

Day 2, ~9.5h collection time. 4 people with wearable sensors					
		Wrist 1	Wrist 2	Wrist 3	Wrist 4
Original Drift	Median	551s	142s	505s	247s
	Max	3788s	317s	1999s	573s
Data Driven Synchronization	Median	24s	6s	75s	36s
	Max	1910s	44s	913s	168s

any drift after the last coupling measured. Therefore, the drift on this sensor continued to grow for the majority of the collection time.

The results for the second day are shown in Table V. This collection has four subjects with wearable sensors, two environmental sensors, and 70 couplings based on alignment point selection between them before the algorithm prunes the alignment points. Wrist sensors 1, 2, 3, and 4 see median drift improvements of 96%, 96%, 84%, and 84%, respectively. Wrist 1 and Wrist 2 both have a few erroneous alignment points that lead to their max errors. These errors can occur in the matching algorithm due to the size of the search window and the similarity of some movements. When there are multiple errors, the outlier rejection will only remove more extreme outliers but not every incorrect coupling. In this dataset, no timing based outliers were removed from the Wrist 1 sensor data, but two were removed from the Wrist 2 sensor data, which is why Wrist 2 sensor has a smaller maximum error.

Wrist 3 has many alignment points but has a number of erroneous points throughout the day. This, combined with the drift of the sensor, increases the median drift error on this sensor. There are also no alignment points removed through outlier rejection. The circumstances for Wrist 4 are similar to the circumstances for Wrist 3 on day 1. There were two couplings in this case that occur approximately 1.5h before the end of the data collection. Both of these couplings match well but cannot remove any drift for the remaining time of the collection. Because of this, the maximum error in this sensor's data occurs at the end of the collection.

5.5. Graph Model

As was discussed in Section 4, we use a graph model to determine which alignment points should be used for synchronization. These decisions are made based on some

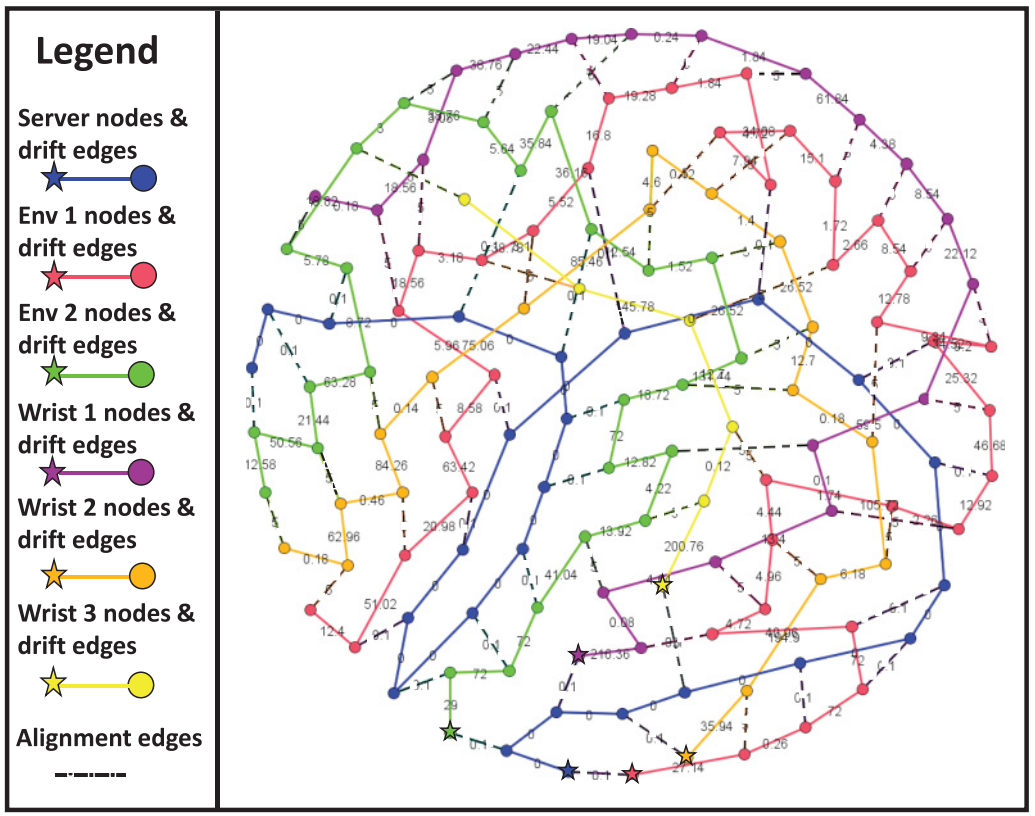

Fig. 10. Day 1 data alignment graph with 1h UTC intervals. The star nodes near the bottom represent the first observation on each sensor.

estimates in our model. Because our sensors have software (SW) based RTCs with 1s resolution and wireless communication times are in the tens of milliseconds for our packets, all interactions with the server can be considered ideal. These alignments would have a very small edge weight, and because the server is our "perfect" global clock in this system, these edges will always be used.

The other interactions that must be considered are the physical interactions between the moveable environmental sensors and the wearable sensors. We must determine which of these alignments are useful for synchronization. Because of the high rate of couplings between the environmental sensors and the server (every 20s) and the high rate of drift on the wearable sensors, our model would use every edge to synchronize. We consider slower servers/environmental sensor coupling rates to determine when there might be a benefit in reducing the number of alignments used in the day 1 graph assuming drift edge weights based on a fixed stability (20,000ppm), alignment edge weight of 5s, and the actual topology from the collection.

Figure 10 is a graph for the collection from day 1 with server/environmental sensor UTC time updates happening at the top of each hour (note that this graph is formatted in this manner for space considerations). Each sensor is color coded, and the stars represent the first observation on each sensor. There are 66 total couplings in the graph. In this scenario, there are six alignments that are not used based on the shortest path algorithm. This is due in large part to the stability of the sensors used in this estimate. With 20,000ppm, there are about 72s of expected drift every hour of operation.

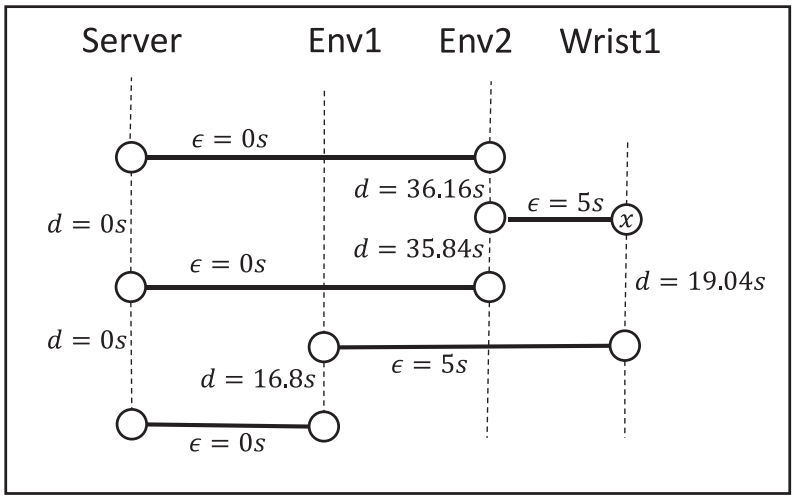

Fig. 11. Subset of a graph for UTC once an hour.

With such large drift, updates from the server are critical for the environmental sensors to have a clock that is reasonably accurate. When increasing the rate of server couplings to 30min intervals, the model again uses all alignments points available for synchronization.

To further illustrate this point, we can look more closely at one of the alignments that would not be used based on the shortest path algorithm when the server couplings occur at 1h intervals but is used when the interval is shorter. Figure 11 is a subset of the previous graph rearranged to more clearly show the selection of alignments. There are two alignments shown for the Wrist 1 sensor. One with Environmental Sensor 1 (Env1) and the other with Environmental Sensor 2 (Env2). There are three paths (two through Env2 and another through Env1) shown in the figure to the observation marked with the χ. The algorithm will find two paths to that observation, one with weight of 40.84s and the other with a weight of 41.16s. Only one of the lower weight paths will be chosen to reach this observation, and because the path through Env1 includes a required alignment, the alignment between Env2 and Wrist1 will not be used in this case. The observation on Env2 that is coupled to the χ observation happened almost halfway between two server couplings. This is why the drift on the two Env2 edges are so similar. If the server couplings occurred more frequently, then the drifts on Env1 and Env2 would be smaller, but the nearly worst-case drift edges seen on Env2 would decrease by more than the drift of the edge on Env1. Therefore, all alignments in this subgraph would be used. As we stated earlier, due to the drift estimate for the sensors, these scenarios only occur when there is an interval of 1h or more for server/environmental sensor couplings in our system.

5.6. Performance Analysis and Future Directions

In general, our alignment algorithm and pruning techniques worked well to reduce the overall drift in the system. However, there were still some cases where we had matching errors that hurt the overall drift reduction. One of the primary factors that generated errors was the window size for the matching algorithm. When the amount of drift in the system can vary, the algorithm needs to be able to handle the worst-case drift. This means that the search windows in the sensor data streams may be larger than necessary for some data. When considering larger windows of data, there

are more opportunities for false positives. For example, if considering a sensor with approximately 20,000ppm of error, then there can be 72s of drift after 1h of runtime. If we do not know whether a sensor is faster or slower than the sensor data that we are trying to align with, then there would be 144s of data to search through.

When considering over 2min of data, it is possible that multiple events similar to the interaction in question could be found. This risk only increases as time passes and the amount of possible drift increases. There are a few concepts that could mitigate this risk. If the sensor stability is consistent, then the search area used in the matching algorithm could be reduced by centering the search on the regions of the data stream based on the expected drift. If the speed of one sensor relative to the other is unknown, then this method would generate two smaller search windows centered around the time of the interaction based on sensor S_n and the drift estimate of the other sensor (i.e., $t_{\vec{i}}^{n} + d^1$). Another concept would be to weight the matching algorithm score (e.g., mutual information) based on the height of the peak relative to other peaks from the same sensor's signals or the proximity of the peak to the expected drift locations. This would essentially be increasing the value of matching scores that fit expectations of the system. Reducing the search area without excluding valid parts of the segment may also be possible by estimating the sensor stability for segments of interest. When the sensor stability varies (as in our case), a method to determine the stability within a specific range would allow for the window size to adjust to fit the needs of each coupling further increasing the opportunity for correct matches. Considering multiple alignment points at a time may also help in reducing ambiguity. Analyzing a collection of points at the same time could reduce search window sizes and allow for reduced processing. Additionally, this method may reduce the possibility of erroneous points by highlighting more logical outliers.

Having more alignment points gives better synchronization results for a few reasons. First, if the drift is not linear, more points over time will help to better track the changing drift. As we saw with Wrist 3 in the first day's data, having a few points early improved synchronization, but the lack of points throughout the collection reduced the potential improvement. Additionally, outlier detection performs better as the dataset gets larger and more consistent. Therefore, looking at more sensors and things in the environment (e.g., lights, laptops, mobile devices, other people) to generate couplings could increase the number of alignment points available and thereby increase the synchronization improvements in the system.

As the number of sensors in the system grows, the amount of ambiguity in couplings grows. Because the algorithms are executed offline or on cloud-based servers, we do not expect computation to be a major concern. The algorithms can also be executed on the data from multiple sensors simultaneously if necessary. The greater difficulty comes as the types of sensors in the system increases. This means that the types of couplings between sensors will increase, and this must be understood before couplings can be found in the data streams. If sensors collect a similar type of data, then the entropy-based alignment point selection could be used to find the couplings. If they collect different data modalities, then the template based alignment point selection must be used. The obstacle faced in both of these scenarios is that an expert will be needed to determine if couplings can occur between sensor types and what those couplings would be. Once the couplings are characterized, they can be leveraged for any dataset. Beyond possible error in the matching algorithms, this is one of the key limitations of this technique. However, once the types of couplings are determined, the algorithms should be able to run without constraint. Moreover, the additional sensors will likely increase the amount of couplings that each sensor is involved in and improve the outlier rejection techniques and improve the overall synchronization.

We generally expect that wireless messaging based synchronization techniques will be more accurate, if available. To compare our technique with these systems would require an on-sensor clock that is more accurate than both. Our current sensors do not have this type of clock. Therefore, we are unable compare our technique with wireless synchronization techniques. It is also important to note that our proposed technique is useful *after* the data collected and hence is orthogonal to wireless-based synchronization techniques where synchronization must occur *as* the data are being collected.

6. CONCLUSIONS

We presented a data-driven synchronization method for sensors along with algorithms to resolve ambiguity in multi-sensor environments. The updates to the algorithm worked well to identify and remove poor alignment points when the sensors had multiple couplings. Additionally, many possible couplings were rejected before being processed. This technique improved median sensor drift error by 66% to 98% in our experiments. This was all accomplished with no additional hardware or software modification on the sensors. Without this technique, the data that had been collected would have remained poorly synchronized.

There were erroneous alignment points that were not removed, but the synchronization was still improved for all sensors. In situations where there is a single coupling on a sensor that meets all criteria (e.g., highest mutual information value, RSSI values, valid data within the delay range of the targeted segment), it is difficult to determine whether this alignment is erroneous, but a larger number of couplings helps to reduce the impact of a single erroneous alignment point on synchronization quality.

We are currently looking into the effects of variations in data sampling frequency, which may require resampling or time scaling, on the accuracy of the synchronization technique. For future work, we will look into a larger variety of environmental sensors that will lead to increased couplings. In this work, the RSSI data were used in a binary fashion. The sensor was considered to be in the lab or not. Using the RSSI to gain more fine-grained information on location may help to further reduce the risk of using incorrect couplings for synchronization and determine if none of the sensors in the environment are responsible for a particular coupling (i.e., a person without a wearable sensor causes the event). Additionally, we consider a fixed graph of alignments and drifts. Each update to the system could affect the other observations. If there are two very reliable couplings with a high-quality clock for a sensor, then other couplings on that clock may become less valuable as we correct the drift through the higher-quality couplings. We will explore how to efficiently solve this problem with a graph that is dynamic.

REFERENCES

T. R. Bennett, N. Gans, and R. Jafari. 2015a. A data-driven synchronization technique for cyber-physical systems. In *Workshop on the Swarm at the Edge of the Cloud (SWEC).*

T. R. Bennett, N. Gans, and R. Jafari. 2015b. Multi-sensor data-driven: Synchronization using wearable sensors. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers.* ACM, 113–116.

D. J. Berndt and J. Clifford. 1994. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, Vol. 10. 359–370.

D. Broman, P. Derler, and J. Eidson. 2013. Temporal issues in cyber-physical systems. *J. Ind. Inst. Sci.* 93, 3 (2013), 389–402.

T. M. Cover and J. A. Thomas. 1991. *Elements of Information Theory.* Wiley-Interscience, New York, NY, USA. 12–49.

E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Num. Math.* 1, 1 (1959), 269–271.

J. Elson, L. Girod, and D. Estrin. 2002. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operat. Syst. Rev.* 36, SI (2002), 147–163.

S. Ganeriwal, R. Kumar, and M. B. Srivastava. 2003. Timing-sync protocol for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. ACM, 138–149

D. Guidoni, A. Boukerche, H. Oliveira, R. Mini, and A. Loureiro. 2010. A small world model to improve synchronization algorithms for wireless sensor networks. In *Proceedings of the 2010 IEEE Symposium on Computers and Communications (ISCC)*. 229–234.

M. Harashima, H. Yasuda, and M. Hasegawa. 2012. Synchronization of wireless sensor networks using natural environmental signals based on noise-induced phase synchronization phenomenon. In *Proceedings of the 2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*. 1–5.

Y. H. Huang and S. H. Wu. 2010. Time synchronization protocol for small-scale wireless sensor networks. In *2010 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–5.

B. Iglewicz and D. Hoaglin. 1993. *How to Detect and Handle Outliers*. Vol. 16. ASQ Press.

S. Jain and Y. Sharma. 2011. Optimal performance reference broadcast synchronization (oprbs) for time synchronization in wireless sensor networks. In *Proceedings of the 2011 International Conference on Computer, Communication and Electrical Technology (ICCCET)*. 171–175.

S. Kajioka, T. Mori, T. Uchiya, I. Takumi, and H. Matsuo. 2014. Experiment of indoor position presumption based on rssi of bluetooth le beacon. In *Proceedings of the 2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. IEEE, 337–339.

K. lae Noh and E. Serpedin. 2007. Adaptive multi-hop timings synchronization for wireless sensor networks. In *Proceedings of the 9th International Symposium on Signal Processing and Its Applications, 2007 (ISSPA 2007)*. 1–6.

F. Lamonaca, A. Gasparri, E. Garone, and D. Grimaldi. 2014. 'Clock synchronization in wireless sensor network with selective convergence rate for event driven measurement applications. *IEEE Trans. Instrum. Meas.* 63, 9 (2014), 2279–2287.

E. Lee, J. Rabaey, B. Hartmann, J. Kubiatowicz, K. Pister, A. Sangiovanni-Vincentelli, S. Seshia, J. Wawrzynek, D. Wessel, T. Rosing, D. Blaauw, P. Dutta, K. Fu, C. Guestrin, B. Taskar, R. Jafari, D. Jones, V. Kumar, R. Mangharam, G. Pappas, R. Murray, and A. Rowe. 2014. The swarm at the edge of the cloud. *IEEE Design Test* 31, 3 (2014), 8–20.

H. Ling and K. Okada. 2007. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 5 (2007), 840–853.

M. Lukac, P. Davis, R. Clayton, and D. Estrin. 2009. Recovering temporal integrity with data driven time synchronization. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. IEEE Computer Society, 61–72.

M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. 2004. The flooding time synchronization protocol. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. ACM, 39–49.

A. Pinho, D. R. Figueiredo, and F. Franca. 2012. A robust gradient clock synchronization algorithm for wireless sensor networks. In *Proceedings of the 2012 4th International Conference on Communication Systems and Networks (COMSNETS)*. 1–10.

X. Qian, X. Shen, G. Dai, J. Zhang, and C. Lv. 2010. Clapping and broadcasting synchronization in wireless sensor network. In *Proceedings of the 2010 6th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. IEEE, 140–145.

P. Sommer and R. Wattenhofer. 2009. Gradient clock synchronization in wireless sensor networks. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. IEEE Computer Society, 37–48.

W. Su and I. F. Akyildiz. 2005. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Trans. Netw.* 13, 2 (2005), 384–397.

A. Swain and R. Hansdah. 2011. A weighted average based external clock synchronization protocol for wireless sensor networks. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops (ICDCSW)*. 218–229.

K. Yildirim and A. Kantarci. 2014a. External gradient time synchronization in wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* 25, 3 (2011), 633–641.

K. Yildirim and A. Kantarci. 2014. Time synchronization based on slow-flooding in wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* 25, 1 (2014), 244–253.

Received December 2015; revised April 2016; accepted August 2016