Proximity Interactions between Wireless Sensors and their Application

Waylon Brunette, Carl Hartung, Ben Nordstrom, Gaetano Borriello
Department of Computer Science and Engineering
University of Washington
Box 352350 Seattle, WA 98195
{*wrb, chartung, bennord, gaetano*}@cs.washington.edu

ABSTRACT

Many applications in ubiquitous computing rely on knowing where people and objects are relative to each other. By placing small wireless sensors on people, at specific locations, and on or in a wide variety of everyday objects we can collect these proximate relationships and deduce much about a person's or an object's context. This paper investigates the practical issues of recording these proximity interactions using RF wireless sensors and explores the benefits of collecting/mining proximity data and how user context and usage habits can be inferred for use by proactive applications. We describe some of the issues we faced in collecting usable proximity data from RF wireless sensors. Specifically, we discuss some of the ranging experiments we conducted, our approach to utilizing the limited local data store, and how we implemented a low-overhead time synchronization scheme. We present initial results from one of the applications we are targeting: a proactive reminding system that informs users when they leave important items behind.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *wireless communication.*

General Terms

Experimentation

Keywords

wireless sensors, RF proximity, distributed clock synchronization, embedded systems, ubiquitous computing.

1. INTRODUCTION

Wireless sensors are now small enough that they can be embedded in many everyday objects. They can be used to gather contextual information through a variety of sensors including light, temperature, acceleration (force), orientation, humidity, etc.. Our interest is in using the sensor's radio itself as a sensor to detect when other devices are nearby. Proximity interactions are likely to prove invaluable for a wide variety of applications that focus on minimizing user distraction by gathering information about the user's context and exploiting that context to better tailor human-computer interactions. Recent work by Lamming [1] with infrared proximity sensors further inspired us to better understand the technical issues involved in collecting and communicating proximity data. With shrinking size and power requirements, wireless sensors are likely to be placed in more and more objects further motivating better understanding of the issues of scale in proximity readings.

Many meaningful proactive applications can be created by mining proximity data gathered by placing nodes on people, in various locations, and on important everyday items. We are exploring several applications including: placing wireless sensors at known fixed locations so that portable sensors can use them as landmarks for determining their dynamic position; attaching sensors to or placing them inside of every object that a person wants to make sure they take with them when they leave a place (e.g., home or office) so that the objects themselves can remind the user when one is left behind; and placing sensors on people and equipment in an operating room to track workflow and equipment usage. This range of uses motivates some of the technical issues discussed in the following sections: collecting proximity data in environments that interfere with perfect radio-frequency (RF) propagation, storing the data efficiently in small memory-limited sensors, synchronizing clocks across sensors to make the data easier to aggregate and process in a peer-to-peer architecture, and optimizing power consumption.

Many different solutions have been proposed to characterize the interactions between people and objects; however, no generally accepted method has emerged for recording proximity data. Approaches using computer vision [2] require fixed infrastructure and video cameras are generally considered intrusive. Solutions such as infrared transceivers suffer from being obstructed easily and not working at all when inside of objects [1, 3]. RF identification (RFID) tags require large and bulky readers that have a high infrastructure cost where the size of the reader correlates to the range of detection. There has also been a lot of work done in the area of precise localization in order to find the exact location or the distance between a person and object. However, these approaches tend to be costly and infrastructure heavy [4].

WSNA'03, September 19, 2003, San Diego, California, USA.

Our goal is to further our research into proactive applications by creating a simple, easily deployable solution that can be used as a test bed to record the interactions between people, objects, and locations. We assert that enough information can be mined from simple proximity data to provide some useful context without requiring a large infrastructure.

To these ends, we settled upon using RF wireless sensors as a simple method of gathering contextual information through RF proximity detection because they are a peer-based, ad hoc solution that requires little infrastructure. They are also an off-the-shelf solution that continues to shrink in size and power requirements. In the near future, wireless sensors' volume is likely to shrink to a few cubic millimeters [5]. This tiny form factor will allow sensors to be deployed into many environments without adding appreciably to usage constraints. Power harvesting research [6] is an emerging area for sensor networks that is likely to produce a sustainable energy model for such sensors.

We present results obtained from our initial implementation and experiments. The work to date only focuses on our experiences collecting and storing the data, leaving the analysis and actuation as part of planned future work. In Section 2, we provide more detail on some possible application scenarios to set a context for our work. Section 3 describes the most relevant parts of a growing body of related work, and Section 4 provides details of our test bed implementation. In Section 5, we discuss some of the challenges we encountered during our implementation such as time synchronization between sensors and optimizing the range of the sensor radios. Section 6 shows some of our initial experimental results. Finally, in Section 7, we present our conclusions and outline plans for future work.

2. APPLICATIONS

Proactive applications try to anticipate users' needs and minimize distraction [7]. To accomplish this, they need to have some model of the users' current context to help them make decisions about what actions to take automatically. When designing our RF proximity system we decided to target applications where our approach could be used to gain insight about context by examining the proximal pairings of people, objects, and locations.

One of our target applications is borrowed from the seminal work by Lamming and Bohm [1] and seeks to create a proactive reminder system. For example, an application could generate an alert on your wristwatch if it sensed that you got into your car to go to work without some important papers. The code for the application might be running on a personal server [8] augmented with a proximity sensor, with another sensor in the car, and others on the papers you tagged (by attaching a proximity sensor to them) the previous afternoon when you decided to bring them home from work. In this case, the important information is that your papers are within your proximity (i.e., in the car) not their precise location in the car. Another example might be to help you find your misplaced PDA. In this case, a nearby display could show the 'bedroom' as the general location for the PDA, since that is where it was last seen by the other sensors. It chose to display the PDA's location because the application has observed that this is one of the items you consistently spend time near and it is not currently present. The location was obtained from wireless sensors placed in each room in your house. This simple hint as to where the PDA may be significantly narrows the search. Another reminding example might entail an application reminding you that you don't have your house keys as you are leaving work. In this example, data mining techniques are used to infer that you are deviating from your past habits by examining the location, time of day, and the items that were in proximity such as purse (briefcase or backpack), laptop, and jacket. The system knows from previous experience that the vast majority of the time this contextual situation occurs your house keys are also normally within your proximity. In these examples, the key piece of information is only the presence of the item, not its precise location (i.e. distance coordinates).

Another possible application is to use proximity data to track and improve asset management and logistics. To improve logistics, we study where people are when they do their work, what tools they are using, and who else interacts with them. By placing a node on every person, object, and location of interest, massive amounts of data could be obtained that include information about how long people were near that location, who else was there, and/or what object(s) were in use. Specifically, we are looking at a scenario set in a hospital operating room where equipment is shared between busy people who are constantly moving. The proximity data allows doctors and hospital administrators to track and organize equipment more efficiently. It could also show patterns in workflow and identify ways to simplify or expedite procedures by identifying repetitive movement patterns.

3. RELATED WORK

Using proximity to gain contextual information is not a new idea. Lamming's Specs project [1] uses low-power infrared to gauge proximity between two transceivers. This work inspired us to look at the issues in using RF-based proximity. RF has benefits over infrared because it does not require a line-of-sight between sensors and allows sensors to be inside of objects (e.g., a purse or teddy-bear). By using RF, a PDA in your backpack can still be seen as being with you, whereas the backpack would certainly block infrared emissions from an object within it. The ParcTab system [3] also used infrared beacons to create location-based context-aware applications and suffered from the same problems. However, the ParcTab was a PDA that was likely to be visible while in use in a user's hand.

Smart-Its Friends [9] use a combination of both RF proximity sensors and accelerometers to create associations between devices based on proximity and similar movement (as in shaking two objects together). The Smart-Its project focuses on helping users make deliberate associations between trusted devices. Our approach differs in that we are concerned with recording all proximity relations including those with objects in the environment that will let us gather as much context as possible for data mining. An early version of Smart-Its was used in a project called MediaCup that sought to enhance everyday objects with sensing capability [10]. Although, this work did use some proximity data to detect meetings in progress, its focus was on sensing different properties of the objects such as the temperature of the coffee cup to infer how long a user may have left it unattended.

Other RF research such as SpotON has focused on using RF signal strength to determine locations of objects [11]. However, RF signal strength is unreliable in dynamic environments and is not necessarily needed to gain the basic level of context that can

be ascertained by proximity. Another focus has been on deploying ad hoc RF beacons to determine location [12] by using them as landmarks. An RF transmitter can be used to broadcast its coordinates (or more semantically meaningful location, e.g., 'kitchen' or 'northwest corner of the building') to any other receivers that happen to come within range.

In support of our approach, Guibas' work focuses on applying mathematical techniques to track relations and reason about them. He points out that it is less expensive for a sensor to sense a relation (such as proximity) than to do expensive localization computations to yield a precise 3-D position [13]. Moreover, simpler relations are likely to be more numerous and more accurate than detailed measurements.

4. IMPLEMENTATION

We chose to implement our proximity system using a Crossbow sensor platform derived from the UC Berkeley sensor 'motes' [14]. The Mica2dots, pictured in Figure 1, are the smallest form-factor platform available and can be easily attached to or inserted into objects. Motes contain an 8-bit microcontroller, radio transceiver, and flash memory for storing data. The code for the motes was written within the TinyOS run-time environment. Once deployed, the motes continually track the presence of other motes within proximity range through radio messages. Our motes are programmed to continually send out identification messages at spaced time intervals to inform the other motes in the area of their presence. There are no false positives with this approach; no mote will hear another that is not actually present. However, because RF reception changes with the details of the environment, two nodes that are near each other may not always receive each other's messages or receive them asymmetrically. An important issue with RF proximity is how much power to put into the radio transmission. Lower power requires nodes to be closer to hear each other, and their signals may be easily attenuated by obstructions (including people) in the environment. Higher power may make the signal propagate too far, thus, making for too large a proximity range and decrease battery life.

We deployed motes as RF proximity detectors to "see" and "remember" who or what was nearby and for how long. Each mote is responsible for keeping a log of all other motes it has 'heard'. The log includes the mote ID, the time the other mote is first heard, and how long it stays within range. Motes log the event to their non-volatile flash storage once the other mote leaves. Each mote sends out a message approximately every 300ms. We chose such a fast update rate to ensure we caught all proximity interactions on a human scale. Several tests were preformed to ensure that a person who came within 3-5 feet of a location would be detected. The motes showed an acceptable rate of detection with a 300ms refresh rate. Because of network interference and lost messages, a timeout was used to make the mote wait before assuming a signal was lost. This helps to smooth the data and reduce the amount of storage space needed for the log. Since every mote broadcasts regular identification messages, received messages do not elicit a response.

With each mote keeping its own log of time intervals and the possibility of asymmetric reception, some form of time synchronization was necessary to make the logs easy to align and compare. This is especially the case if we want to do so on the motes themselves rather than a centralized infrastructure that can keep track of the drift of each node. However, due to the nature of human-scale proximity interactions, we do not need a high-precision method that would only consume valuable resources (communication bandwidth and power). Instead, we decided that for most data mining applications, knowing if an object was present any time during a particular one second time window is sufficient. Therefore, our synchronization scheme can be loose – to within one second resolution. This allows us to keep power consumption at a minimum by simply sending a local timestamp in each radio message. The receiving mote adjusts its clock ticks to bring its own clock in line with the time it receives. If it is behind, then it runs its update increments slightly faster until it catches up. If the received time is less than the local time, then the device does not set its clock backwards to ensure that the internal logs stay in temporal order. Instead, it temporarily slows its update increments. This leads to an averaging affect among the communicating motes and keeps their clocks synchronized to within the required resolution. The loose time synchronization method is discussed in further detail in section 5.3.

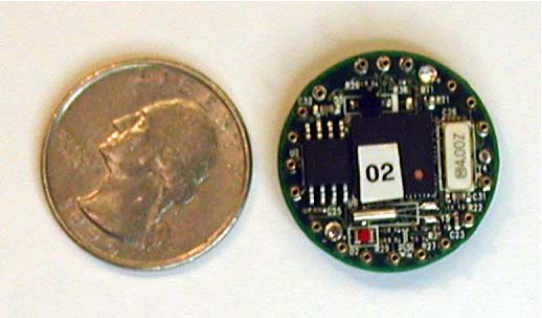

FIGURE 1: Picture of Mica2dot

Initial mote setup and data downloading was done using serial communication with a PC application. We did not assume that all of the devices start with the correct time. All devices start with time zero until they receive their first proximity message. This way a mote can have its timer explicitly set by a PC, which in turn will cause the time to be propagated out to other active motes via the radio. This causes a cascade as the other motes come into contact with any mote that has a local time.

5. CHALLENGES

In this section, we discuss some of the issues that must be addressed in building RF proximity sensors. Specifically, we discuss some of the ranging experiments we conducted, how we implemented a low-overhead time synchronization scheme, and our approach to utilizing the limited local data store on board the sensors.

5.1 Radio Range

For radio proximity detection to be interpreted in a meaningful way, the radio transmission range must be limited to a known distance. Unfortunately, the wireless sensors we use are designed to have a large radio range so that they can easily form ad hoc networks. In our tests, the Mica2dots using 433 MHz Chipcon radios showed very large radio range even at the lowest transmit power settings. Using the quarter wave whip antenna that is

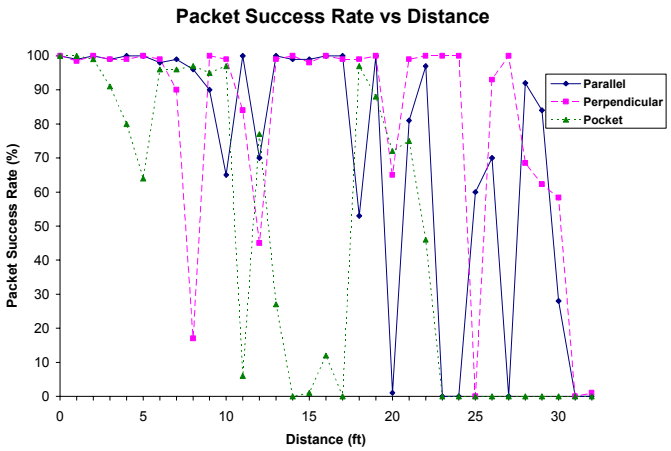

FIGURE 2: Distance of Radio Packet Reception

supplied with the motes and the transmit power set to the lowest we still obtained reasonable reception 80 ft away. To compensate for this very large radio range, we cut the antenna length to 1/6 of the wavelength to reduce the transmission distance. Results from experiments with the short antenna is shown in Figure 2 – the shorter antenna yielded an unobstructed transmission distances of about 18ft for good packet reception continuing out to about 30ft before tailing off. Figure 2 shows the percentage of successful packets for three configurations. In the first case, the antennas between the two motes are arranged to be parallel to each other. The second case has the mote antennas perpendicular. The final case shows the effects of a mote being in a front pocket while the person is facing the other mote. For this last case, the reliable range dropped to 10ft. A large number of people in the vicinity of the experiment disrupt the transmission and make the effective transmission radius much smaller. As Figure 2 illustrates RF is noisy and is affected by many conditions. It is hard to obtain precise models of distance measurements as people, objects, and locations have effects on the distance a radio can transmit. To compensate for this, the power levels can be raised on the mote's radios. However, too much power will result in inaccurate results as motes can transmit much further with nothing blocking their line of sight. We also plan to experiment with attaching attenuators to the antenna to limit the power while maintaining good reception at the lower range. For each RF technology one must find the right balance between antenna length, radio power setting, and the desired proximity range. One approach we plan to investigate is the dynamic variance of RF power on a platform that has a wider range of power settings. Proximity data will then need to be tagged with power setting as well as local time. However, this may generate enough data for an algorithm to be able to detect variance in reception success to different motes and thus compensate for some of the obstructions in the environment.

5.2 Time Synchronization

For partial proximity data to be useful there needs to be a global time that will allow temporal synchronization of data from different devices. However, in our proximity applications, nodes can be separated from the network for extended periods of time making it difficult to keep a single synchronized global time. According to Elson and Römer [15] wireless sensor networks do not fit traditional network assumptions and therefore neither have the resources nor infrastructure required for most traditional synchronization algorithms. They claim the best solution is for each node to keep its own local time rather than have a global clock. One method they suggest for synchronization is to have each node build up a conversion table of parameters that relate the local clock to other nodes' clocks.

Our implementation uses a loose synchronization scheme to establish a consensus time for data analysis. Each node keeps its own internal time that is continually being updated by its neighbors towards an overall network wide consensus time. When a new proximity message arrives from another device with a different global time each device assumes they are only partially right. The devices then average their local time with the other device's time to produce a revised local time. Note that individual devices do not need to be aware of the actual world time as long as they all use identical and uniform time divisions and can agree on a global consensus time. A typical user moves slowly enough that a coarse time scale with resolution of one second is adequate to record interactions with a person's surroundings. Therefore, it did not seem necessary to use a tight synchronization technique to keep the local times of the devices within milliseconds of each other. These more typical synchronization methods would have only shortened the life of the sensors by unnecessarily wasting valuable communication and computation resources to implement the synchronization algorithms with much higher resolution than required.

There are two main concerns about devices that are only loosely synchronized. First, radio traffic could vary the send-receive delay of time-stamped messages. This shouldn't be too large of a factor as the motes are located only a short distance away from each other and the time of flight should be negligible. In addition, messages are sent and processed in milliseconds. Since we are only concerned with synchronization at the level of a second, we assume this delay is not significant. Second, the devices' drift rates might not permit convergence to a synchronized time. If devices do not converge within a few seconds then reconstructing the temporal ordering of events across all the devices will be made much more difficult. We decided to conduct a simple test using three motes to see whether the devices drift significantly. We found that for a given 12 hour period, the devices did not drift more than 1 second. According to Elson and Römer, typical sensors will drift about 0.6 ms after 60 seconds [15]. Since the devices do not have a significant drift rate they should be able to stay synchronized with an occasional correction via the radio.

5.2.1 Evaluation

We ran several simulations to verify that our averaging method would keep our proximity devices synchronized. The simulator was a discrete event driven simulation where units of time were simulated at 0.1ms. Three separate simulations were run at "extreme", "high", and "normal" (or typical) drift rates. Each sensor was assigned a randomly chosen drift rate within the bounds specified in Table 1. Our simulation results showed that global time averaging yielded tighter synchronization than was required, namely, better than one second in all cases.

5.2.1.1 *Simple Averaging*

To see the effect of drift rates on the device's local time, we simulated 100 devices that sent messages to each other over a one-week period. These 100 devices were mostly isolated and could only occasionally communicate. The goal of the simulation was to see if separated devices that only occasionally saw another node could still all converge to a single global time. Each device was set with a random drift rate (either faster or slower) bounded by a maximum and minimum. In this simulation, the sensor sends a proximity message every 0.5 seconds. However, the sensor is only able to successfully communicate with another node 1/120 of the time. Overall, this means that a sensor will actually communicate with another node approximately once per minute. Three ranges of drift rates were simulated and are listed in Table 1. The drift rate indicates how much extra time a device thinks has passed per minute. Table 1 also shows the results of the experiment including the largest difference in internal time between any two devices that occurred during the week. Note that the maximum difference is approximately an order of magnitude greater than the maximum drift per minute. This is as expected with an average of one communication per minute. There is likely to be at least one node that is unfortunate enough to not communicate for 10 consecutive minutes.

TABLE 1: Time difference between any two nodes (simple averaging)

	Drift Rates (ms/min)	Max. Diff (ms)	Mean Diff (ms)	Std. Dev.
Extreme	20 – 80	886.5	100.0	76.0
High	2 – 8	93.5	10.1	7.7
Normal	0.2 – 0.8	8.3	0.9	0.7

We also used the simulator to see the effects of nodes trying to converge while having very different internal times. To test this we gave each node a random start time of less than a minute. Figure 3 shows the rate of convergence under the different drift rates. The simulation indicates that if each device adjusts its idea of the global time by averaging its local time with that of other devices in the network, there is a trend for the devices in the network to agree upon a global time within a few minutes, even with different local starting times and extreme drift rates.

5.2.1.2 *Averaging with Groups*

Applications of proximity devices will likely involve several devices that are in constant contact with each other forming a well-connected cluster. For example, this might be the case for the devices a person carries in their pockets or bag. These groups of devices will generally have agreement on a global time but will occasionally be exposed to other devices that may or may not agree as they have been separated for a long period of time. For example, this might be the case with sensors that were left at work over the weekend. These network partitions allow groups of sensors to be isolated and to drift away from the consensus time. We simulated the effects of this situation by clustering devices together into several groups to see if our loose synchronization scheme would work with isolated clusters. For the loose synchronization method to work the network of isolated clusters would eventually have to agree upon a global time. To test this we simulated 100 sensors that were randomly assigned to 20 different clusters of sensors where the number of sensors varied per cluster. In the simulation 2/3's of the clusters were stationary and never

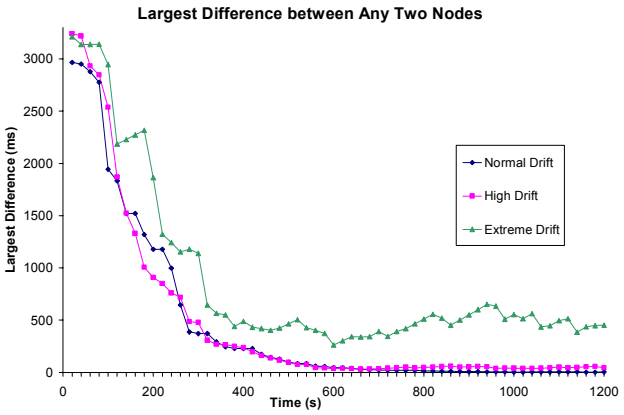

FIGURE 3: Convergence of time between nodes

communicate with each other, while the remaining 1/3 were mobile clusters that could move within range of any cluster. Mobile clusters moved to another cluster after some random interval between 5-20 minutes. The mobile cluster was then isolated for 5 minutes before moving to another cluster. The idea was to mimic typical behavior of a user moving from room to room at home or work. The results in Table 2 show that group averaging will only work with reasonable drift rates. The maximum difference between any two nodes only remained below 1 sec in the case of normal drift rates. Figure 4 shows how the isolated groups cause local time to drift and then quickly move closer to global consensus time when a mobile group arrives.

TABLE 2: Time difference between any two nodes (group averaging)

	Drift Rates (ms/min)	Max. Diff (ms)	Mean Diff (ms)	Std. Dev.
Extreme	20 – 80	8615.4	1214.2	1197.4
High	2 – 8	1593.0	180.4	183.0
Normal	0.2 – 0.8	105.9	14.6	13.7

A consensus time can only be reached with nodes having small enough drift rates. Of course, if we increase mobility and make network partitions more short-lived, then consensus can be reached more easily. These are the two key parameters for loose synchronization to work well: the mobility of the moving clusters needs to be high enough to distribute the consensus time frequently enough and the drift rate needs to be minimized.

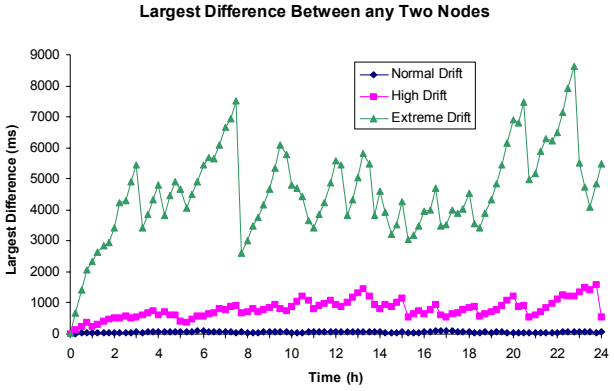

FIGURE 4: Largest difference in local time between nodes

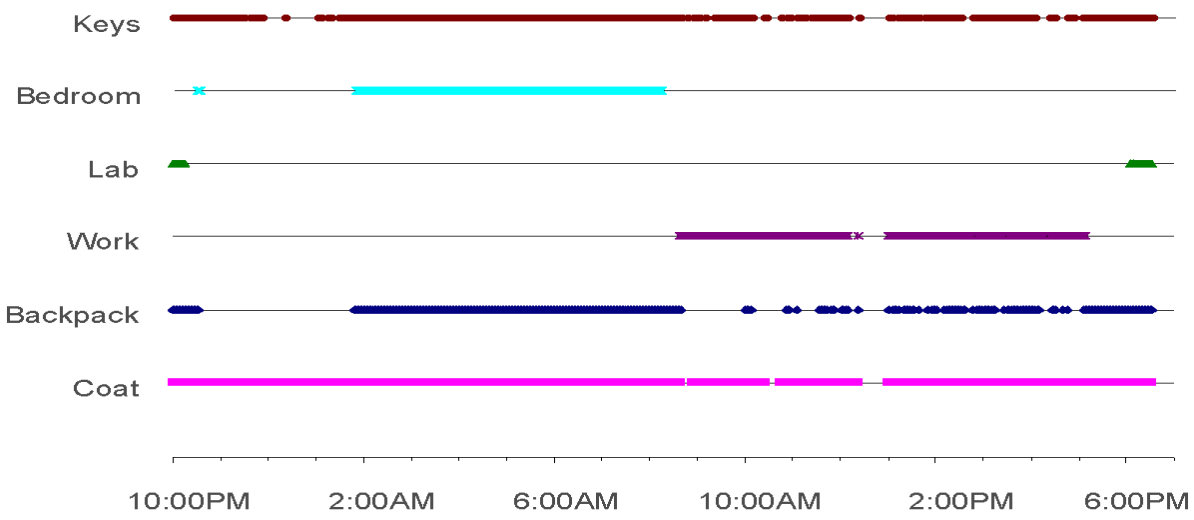

FIGURE 5: Day-in-the-life interactions of a subject and his belongings

5.3 Limited Storage & Power

The major factors that determine the useful lifetime of a node in our system are storage size and battery life. These are two of the current limitations that plague sensor network research in general. We believe both issues can only be solved with new technology. Storage density is continuing to rapidly increase making it soon possible to have megabytes of storage on even the smallest, lowest-power sensors. However, the same rate of improvement is not true for power. Power harvesting methods are in the earliest stages of research and battery life is only slowly increasing (approximately doubling every 10-20 years for a given technology).

Today, however, storage size does limit the length of time a mote can gather data before needing to offload it. The motes we use have limited storage capabilities. In our implementation, several methods were used to decrease the amount of data that needed to be stored thereby increasing application lifetime. To save log space only the minimum amount of information was stored in each log entry: mote ID, the time when the mote was first heard, and how long the mote remained in proximity range. We also implemented a smoothing operation that waited four consecutive updates before considering the mote gone to compensate for radio unreliability. The smoothing operation significantly decreased the number of log entries and created cleaner data.

Currently motes only have 4Mbits of space for non-volatile data storage which is not enough for large deployments. The number of devices present and the amount of movement increases the number of log entries needed dramatically. As the devices that are in proximity increases, the number of possible log entries grows quickly with an upper bound of $O(N^2)$ where N is the number of devices. As the movement in the system increases the number of times log entries must be made for items entering and leaving range also increases. If this system is to be used on a large scale with hundreds of devices over long periods of time more storage space is required. Compounding the problem is the fact that the TinyOS log component moves data in and out of permanent storage in 16 byte blocks causing bytes to be wasted within every log entry. We plan to rewrite this portion of TinyOS in the future to solve this underutilization problem.

We have only scratched the surface of the power issue in that we believe it can only fundamentally be solved with new technology. Some methods we use to reduce power consumption are to increase the amount of time the mote is in sleep mode, limit the number of log writes, turn off as many components as possible, and implement a time synchronization system that added no extra radio messages.

Another benefit of our time synchronization is it allows us to create listening windows to reduce the duty cycle even further. However, with only loose synchronization the listening windows need to be larger than what would be possible with perfect synchronization but they still reduce the radio listening time significantly.

One major problem we encountered is the very limited battery life of the new Mica2dots. The Mica2dots are powered by coin cells that have approximately 220mAh (yielding 4-5 hours of operation) compared to the AA's of the larger and more flexible Mica motes which have approximately 2600mAh (for 2-3 days of operation). To compensate for the smaller coin cell batteries of the Mica2dots, we are looking into ways to decrease the duty cycle even further after we have gained more experience in mining the proximity data.

6. EXPERIMENTS

In our experiments we were only concerned with collecting data from people who were interacting with other tagged items and then being able to transfer the data to a central location for further data analysis. We used both Micas and Mica2dots as platforms.

The first experiment we report lasted a total of 21 hours using Micas. For this test we deployed approximately 6 motes to be 'senders', and 1 mote to be a 'receiver'. Recording proximity data asymmetrically is appropriate for our reminder application. The receiver was placed in the subject's pocket so it could be carried around throughout the day. The senders were either

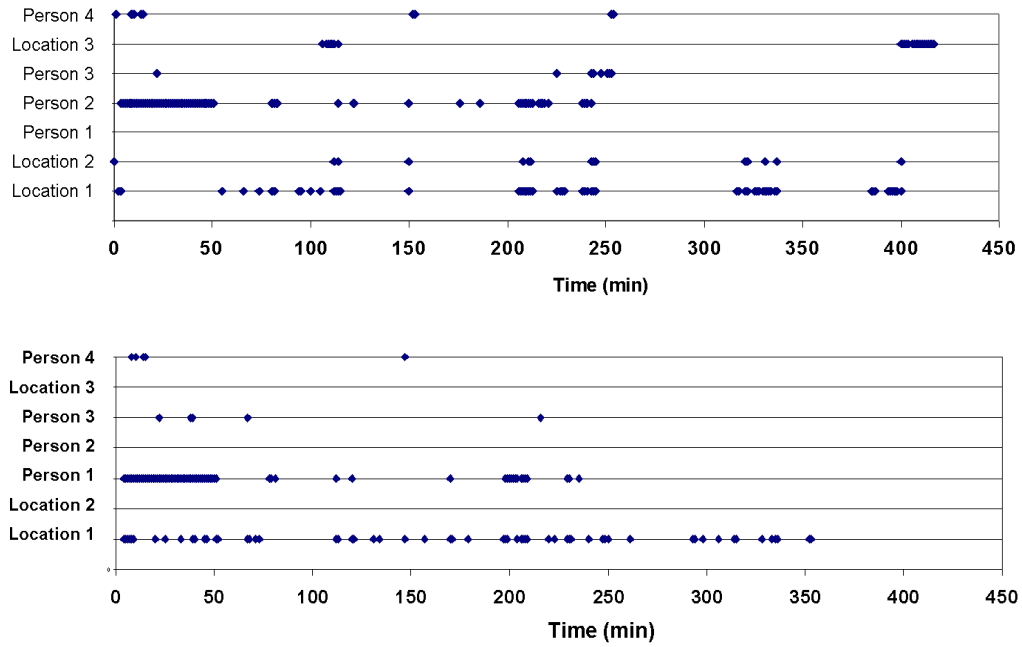

FIGURE 6: Interactions of Person1(top), Person2 (bottom), and the surrounding environment

attached to specific items, or placed strategically in different locations. Figure 5 represents the data that the receiving mote recorded throughout our day test. The experiment began at 10pm and finished at approximately 7pm the next day.

The experiment started with the subject in the research lab at 10pm. Shortly afterwards, the subject went home with his tagged objects (i.e. backpack, coat, and keys). After arriving at home the subject ventured out again to meet some friends. When the subject returned home, he slept for the next 7 hours as corresponds to long solid period 1:30am to 8:30am in Figure 5. In the morning the subject took a short bus ride to work and arrived just before 9am. The figure shows when the subject took a lunch break and was separated from all the motes for a short time. At about 5pm, the subject left work and took a longer bus ride back to the lab to conclude the experiment.

A few days after the experiment was complete the subject was given Figure 5 to verify the data. The subject was able to recreate most of his day and explain why sensors would not have been detected by each other at certain times. For example when he went out at night to meet his friends he left his backpack at home. Another example is he told us that he put his keys down around midnight when he arrived at his friend's house explaining the breaks in the graph of him being by his keys only sporadically.

We set up a second experiment to test our system by tagging 3 locations and 4 people working within the lab. This experiment ran for 7 hours. The motes gathered proximity data symmetrically by logging each time a person came in contact with another person as well as the time they lost contact. The graphs of two of the people tagged show times and duration of when each was within proximity of other tagged objects. The data gathered showed some interesting trends in terms of proximity, without having the need to calculate exact locations. For instance, the graphs show that Person2 frequented Location1, indicating that that is where most of their work was done. The data also indicates that there was a zone in between Location1 and Location2 in which a person could stand and be within proximity of both sensors. Person1 and Person2 were together for extended lengths of time between times 5-60 and 200-230, indicating a meeting or conversation between the two.

This experiment also showed that Person2's mote missed recording Person1 a few times. This most likely occurred because someone was probably walking at a distance that was on the fringe of the radio range and only remained in range for a short time. Verification of data is difficult since we did not have a reference to compare to and sensors occasionally miss messages. It is difficult to track loss in an actual application deployment since the motes are always beaconing and you don't know exactly what is within range at certain times.

These initial experiments show that our method of collecting proximity data seems to be feasible. Our initial experiments show that useful data can be gathered through RF proximity interactions. Future experiments are needed in other real-life environments and for longer durations. We plan to expand our work to apply machine learning techniques to larger volumes of data so that we can experiment with detecting anomalous patterns that may lead an application to ask for user intervention. We are also searching for methods to verify mote interactions while they are in constantly changing environments – typical of applications such as the personal reminder system where the environment can be changing between a house, bus, mall, or a car.

7. CONCLUSION & FUTURE WORK

Collecting proximity data with wireless sensors is a simple method of obtaining large amounts of context information. However, further work needs to be done on exploring how this data can be used in real time, how difficult it is to extract meaningful data, and how well patterns can be learned from proximity interactions. We plan to continue our work evaluating

the collection of proximity data as a method of obtaining context information by doing much more extensive testing that will include multiple day experiments of common situations using between 50 to 100 motes. This will allow us to further verify our loose time synchronization scheme and start to examine ways to deal with noise in our data. Our initial implementation will continue to be refined and used in the design of future experiments and in the development of applications such as an object reminder system and workflow analysis. Further work also remains in the area of power and storage management. Power management is one of the critical areas needing work as it is one of the largest hurdles to building a deployable system that will last several days.

The next major step will be to design and implement an application based on the analysis of the proximity data. The application we implement may be a reminder system that tracks various objects and generates a reminder when an expected object is not present. This will present many challenges such as finding methods to dynamically offload the data from logs as they fill up. Once offloaded the logs need to have some distributed algorithm that allows all the data to be reassembled for analysis. Once the reminding pattern is learned it needs to be stored in some way that is easily accessible by the sensors to determine if a reminder is needed.

Another possible application may be to track the people and equipment in an operating room. In this scenario, motes would be attached to items that are continually being moved between operating rooms as well as on the doctors or nurses that use them. This would allow for the study and organization of operating room equipment and workflow.

Finally, we believe that a wide range of sensors can be used to gather proximity data. Infrared and RF sensors have gained much attention for now, but both these technologies require relatively large devices with high power needs. A more promising approach may be to utilize passive RFID tags. With a combination of mobile and fixed readers, it may be possible to gather very similar proximity data but with completely passive tags in the objects [16]. Issues that arise in this case are a complete loss of symmetry in the proximity data collected (making analysis potentially more difficult and only from certain devices rather than from any), a potentially more fragile system with the fewer readers being points of failure, and greater power needs for the mobile readers. In the end, it is likely that a combination of sensor technologies will be tailored to particular application domains.

8. ACKNOWLEDGMENTS

We would like to thank Corey Ojima and Tim Underwood for their help in developing and testing the loose time synchronization of the sensors. Michael Perkowitz provided valuable suggestions on the presentation of the material.

9. REFERENCES

[1] M. Lamming and D. Bohm "SPECs: Personal Pervasive Systems". IEEE Computer, Vol 36, No 6, pp 109-111, June 2003

[2] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," Proceedings of the IEEE, Vol. 89, No. 10, pp. 1456-1477, 2001

[3] B. Schilit, N. Adams, and R. Want. "Context-aware computing applications." In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp 85- 90, 1994.

[4] J. Hightower and G. Borriello. "A survey and Taxonomy of Location Systems for Ubiquitious Computing" UW-CSE Tech Report #01-08-03, August 2001

[5] J. Kahn, R. Katz, and K. Pister. "Next century challenges: mobile networking for Smart Dust." In Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking, pp 271-278, 1999

[6] M. Rahimi, H. Shah, G. Sukhatme, J.n Heidemann and D. Estrin. Project available at http://www.cens.ucla.edu/Project-Descriptions/Energy_Harvesting/index.html

[7] Tennenhouse, David. "Proactive Computing." Communications of the ACM, Vol. 43, No. 5, pages 43-50, January 2000.

[8] R. Want, T. Pering, G. Danneels, M. Kumar, M. Sundar and J. Light. "The Personal Server: changing the way we think about ubiquitous computing." Proceedings of UbiComp 2002, pp 194-209.

[9] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl and H.W. Gellersen, "Smart_Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts". Proceedings of UbiComp 2001, pp 116-122.

[10] Gellersen, Hans-Werner and Michael Beigl and Holger Krull. "The Media Cup: Awareness Technology Embedded in an Everyday Object." Proceedings of the First International Symposium on Handheld and Ubiquitous Computing, No. 1707, pp. 308-310, Springer, September 1999.

[11] J. Hightower, R.Want, and G. Borriello, "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength," UW CSE 00-02-02, Feb. 2000.

[12] N. Bulusu, V. Bychkovskiy, D. Estrin, and J. Heidemann. "Scalable, Ad Hoc Deployable, RF-Based Localization." Proceedings of the Grace Hopper Celebration of Women in Computing, October, 2002

[13] L. Guibas, "Sensing, tracking, and reasoning with relations." IEEE Signal Processing Magazine, Vol 19 No 2: pp 73-85, March, 2002.

[14] http://www.xbow.com/Products/ Wireless_Sensor_Networks.htm

[15] Elson, Jeremy and Kay Römer. "Wireless Sensor Networks: A New Regime for Time Synchronization." Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I), 28-29 October 2002.

[16] Guide Project Web Page, http://seattleweb.intel-research.net/projects/guide/index.htm.