Towards Continuous Asset Tracking: Low-Power Communication and Fail-Safe Presence Assurance

Bernhard Firner, Prashant Jadhav, Yanyong Zhang, Richard Howard, Wade Trappe
WINLAB, Rutgers University
North Brunswick, NJ 08902, USA
{bfirner,pjadhav,yyzhang,reh,trappe}@winlab.rutgers.edu

Eitan Fenson
Inpoint
770 Lexington Ave, Sixth Floor
New York, NY 10021
e-fenson@inpointsys.com

***Abstract*—Asset tracking is an important application domain for wireless sensor networks. However, continuous tracking of a large number of items at the individual item level over a significant period of time is still not feasible. There are two main obstacles. The first is the need for efficient, low-power communication protocols. Many current protocols employ energy-expensive methods to achieve reliable communication for arbitrary traffic situations. Such protocols are not suitable for continuous asset tracking applications. The second challenge is the lack of a robust presence detection algorithm that can differentiate packet losses caused by a missing item from packet losses caused by the ambient radio environment.**

In this paper, we designed a simple communication protocol, Uni-HB, and demonstrated it can lead to longer system lifetime and higher communication reliability than several popular protocols. We also devised two robust detection algorithms that can yield low false alarm rates while achieving timely loss notification. We took an experimental approach, and evaluated protocols on a generic embedded hardware platform that has an similar architecture to motes. We also derived analytical models to validate our experimental measurements.

I. Introduction

Wireless sensor networks have enabled many applications that were impossible before, some of which are continuous, item-level asset tracking applications. A system that provides continuous item-level tracking capability can be useful in many scenarios. Imagine, such a system will allow the customers at a jewelry store to try on the merchandise and appreciate them from many angles while *freely* strolling around the store; such a system can help keep track of the whereabouts of patient charts so that they are *always* available when needed; such a system can also ensure a soldier's backpack is *constantly* equipped with all the required necessities. Compared to traditional asset tracking applications which can only report missing items at a coarse temporal (i.e. once a day) and spatial (i.e. not every item) granularity, continuous, item-level asset tracking applications demand much finer tracking granularity both in time and space. These applications must also be fail-safe, unlike many current inventory systems. Despite the much finer tracking granularity, these applications are not willing to compromise on the system lifetime – they usually require more than one year's lifetime using coin cell batteries. The finer granularity of this system lends itself to a new kind of asset tracking system - a *presence assurance system* (PAS). Whereas the status of an item in current asset tracking systems is expressed in the sentence "item X was last seen at location Y" the status of an item in a PAS is "item X is currently at location Y." If a PAS is robust against sensor failure, tampering, and theft then we call it a fail-safe PAS.

Building a fail-safe PAS poses new challenges for the underlying system design, in both hardware and software. Pre-existing solutions do not adequately address these challenges[1]. For example, passive RFID tags, which are popular for traditional inventory tracking applications, suffer from poor range, difficulty avoiding transmission collisions in dense environments, and poor performance when attached to certain items. Sensor networks have also been deployed for asset tracking purposes [2], [3]. While these platforms provide good range and sophisticated communication protocols, they are designed for more general-purpose usage, and their protocols and algorithms may not suit the specific requirements of these applications, thus leading to excessive energy consumption and much reduced lifetime.

Earlier papers [4], [5] have argued for a "push" architecture for continuous asset tracking applications in which a sensor is attached to every item in the system, and the sensor periodically (e.g., once a second) transmits a packet announcing its presence. Once a sensor is not heard for a period of time, the system can declare that this sensor, or the item that it is guarding, is absent. Though this architecture has been proven effective through simulations, practical studies on how to build such an architecture are lacking because some of the previously discussed algorithms are too complex to be implemented on a simple and low-cost platform. In this paper, we take a bottom-up approach, starting from a bare hardware platform [6], called *PIP*, which has similar components to a mote[7] and can be programmed in embedded C. We build low-power and reliable protocols and algorithms on this platform by taking advantage of the unique characteristics of fail-safe presence assurance systems. The PIP serves as a test platform that is general enough for us to implement different functionality while being similar enough to a final product to give us realistic feedback and has been used in field trials in the jewelry business and in the medical industry to track equipment, records, and personnel in an outpatient facility.

Our contributions in this paper are two-fold. The first is comparing several popular communication protocols using the PIPs: an ACK-based protocol, a carrier sensing based protocol,

a TDMA-like protocol, and a uni-directional heartbeat protocol in which a PIP transmits packets at regular intervals. The predictability of packet transmission time and the extremely short packets in our system (only a few bytes) make the uni-directional heartbeat protocol more suitable than the other three protocols in lowering energy consumption while providing reliable packet delivery. The experimental measurements are also validated by our analytical model results. Also, we show that we can predict the packet collisions in this protocol in software because the packet arrival times from the same sensor are regular and can be easily anticipated.

However, packets can also be lost due to the ambient radio environment, which can lead to incorrect detection results if the packet loss is assumed to correspond to an item loss. Our second contribution in this paper lies in the design and development of a technique that attempts to differentiate packet losses due to an absent sensor from packet losses due to ambient loss. Our first detection algorithm is based on the length of packet miss chains: it discerns the sensor as missing when the current miss chain is much longer than what has been observed in the history. Our second detection algorithm checks the probability of a miss chain by calculating the probability of an ambient loss based upon RSSI values. Using a trace collected over the course of a week, we show that our detection algorithms can yield both low false alarm rates and short detection latencies.

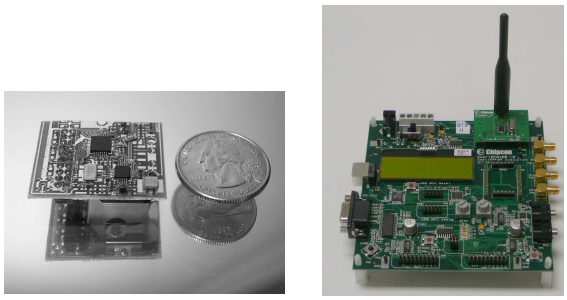
The remainder of this paper is organized as below. Section II describes the considered system architecture and the hardware platform used for implementation and evaluation. Section III compares several communication protocols and demonstrates the effectiveness of the unidirectional heartbeat protocol. Section IV discusses ambient loss and presents two robust detection algorithms with results. Finally, Section V provides concluding remarks and future direction.

II. System Model

A. RollCall Software Architecture Description

The target applications in this paper – fail-safe presence assurance systems – have two main requirements. First, the lifetime of the system must be long enough to be meaningful, which calls for carefully-designed, low-power communication protocol, especially if we want to keep the sensor size (and the battery size) small. Second, the detection of a missing item must be timely and robust. The detection algorithm must have a low false positive rate and low latency. These two goals will be our design objectives in this paper.

There are two models for tracking protocols - *push* and *pull*. A pull model is when a server or basestation initiates contact with sensors by polling them for data, such as in passive RFID systems. The pull model has inherent limitations for continuous presence assurance. Passive tags scavenge their energy from the transmitter and end up supporting only limited ranges and low item densities. Active tags using the push model must spend most of their energy listening for commands from the basestation and will have short lifetimes as a result. We make the important observation that, in general, receiving/listening consumes a significant amount of power in radios and the power to receive is comparable to the power to transmit in low power systems. Therefore, an extensive period of listening has a greater energy impact than the transmission of short packets. This suggests that *receiving is not energy-free and should be minimized as much as transmission*.



(a) (b)

Fig. 1. (a) PIP. (b) Base-station: Texas Instruments CC1100/CC1150-868/915 MHz Development Kit

On the other hand, in a push model, the sensors initiate data transfer by broadcasting beacons that signal their presence to any listening basestations. The total energy consumed is reduced since the sensors do not even need to have a receiver. However, multi-hop push networks are ruled out because receiving packets is necessary in these networks. We note that there are some protocols that are designed to minimize the receiving operations in general single and multi-hop sensor networks, such as TRAMA [8] and B-MAC[9], but the amount of time and energy spent receiving is still quite substantial. Therefore, we advocate the adoption of a single-hop push architecture for energy-conscious tracking applications. To support a large number of items that are spread out over a large area, we can increase the number of basestations. We refer to this architecture as *RollCall*^{TM} [4].

In RollCall, each sensor sends out a beacon announcing its presence during every epoch. As a result, if no basestation hears from a sensor for a number of epochs, as determined by the detection algorithm, it will declare that the corresponding item is missing. Using different epoch durations will affect both tag lifetime and report latency. Here, we assume that a sensor that runs low on energy will notify the basestation before completely draining its battery. The beacons transmitted by each sensor can be made very short to further conserve energy. In our design, we only include the ID number of the sensor with a sequence number in the beacon packet. In our experiments, we used a 1-byte ID field and a 1-byte sequence number field for simplicity. In a real deployment the ID field would be larger and the sequence number would be smaller but the packet payload would still be two bytes with longer IDs accommodated by distributing the ID across several packets. In addition, appropriate encryption can be included if needed to satisfy security requirements.

B. PIP Hardware Description

To keep our hardware testing platform generic and suitable for commercial use, we sought low-cost, low-power, and easy

to program devices. We chose the 16-bit Silicon Laboratories C8051F321 microprocessor[10] and a Chipcon CC1100 radio transceiver[11] powered by a 20 mm diameter lithium coin cell battery, the CR2032[12]. The sensors and basestations share the same chipset and radio and both are programmable in C. Figure 1(a) shows a *persistent identification packets* (PIP); a more detailed description can be found in [6].

Many small steps were involved in minimizing the PIP's power consumption but there were two major optimizations. First, we implemented a low-power sleep mode for the PIP during idle operation. In this mode, all of the peripherals, including the radio, are turned off and the microcontroller is driven by an external 32 KHz oscillator. Second, we carefully tuned the speed of the microcontroller clock and the internal bus clock to achieve minimum energy consumption during data transfer from the microcontroller to the radio.

We operated the radio link at 902.1 MHz with MSK modulation, at a 250kbps data rate and a programmed output power of 0dBm. CRC checks are enabled but forward error correction is disabled to save power. The CC1100 is used in many small radio platforms, such as the Berkeley Motes, so the power consumption characteristics of our platform will be similar to those of most other platforms (if care is taken with low-power programming and protocols).

The basestation (shown in Figure 1(b)) shares the same hardware as the PIPs. Since the basestations are not under the same space constraints, they have a tuned 900MHz monopole antenna attached. The current basestations are equipped with Ethernet, WLAN, or a standard USB connection for data transfer. Such a basestation is inexpensive, thus we can afford enough to ensure adequate coverage.

In some asset tracking systems motion activated hardware is used to save energy by allowing the sensors to adjust their transmission rate based upon detected movement. However, since our goal is a fail-safe presence assurance system we cannot reduce the transmission rate for fear of tampering or theft. If an adversary disables a sensor physically or blocks its transmissions (such as with an RF absorbing bag) while the sensor is transmitting infrequently, the time before the item's loss is detected will be long. Such a system would not be fail-safe.

III. Low-power, Reliable Communication Protocols

We must now find a protocol that is energy efficient but still reliable enough for fail-safe presence assurance. Many energy-efficient communication protocols have been proposed [8][13][9], but these protocols were created to deal with communications in generic wireless sensor networks. The characteristics of the RollCall system are very different from those of a general wireless network, and create unique opportunities for optimizing energy consumption during communication.

A. Comparison of Four Typical Communication Protocols

In this study, we compare four typical communication protocols on PIPs, and measured their energy consumption as well as communication reliability. We use experimental trials and derived analytical models to calculate the performance metrics.

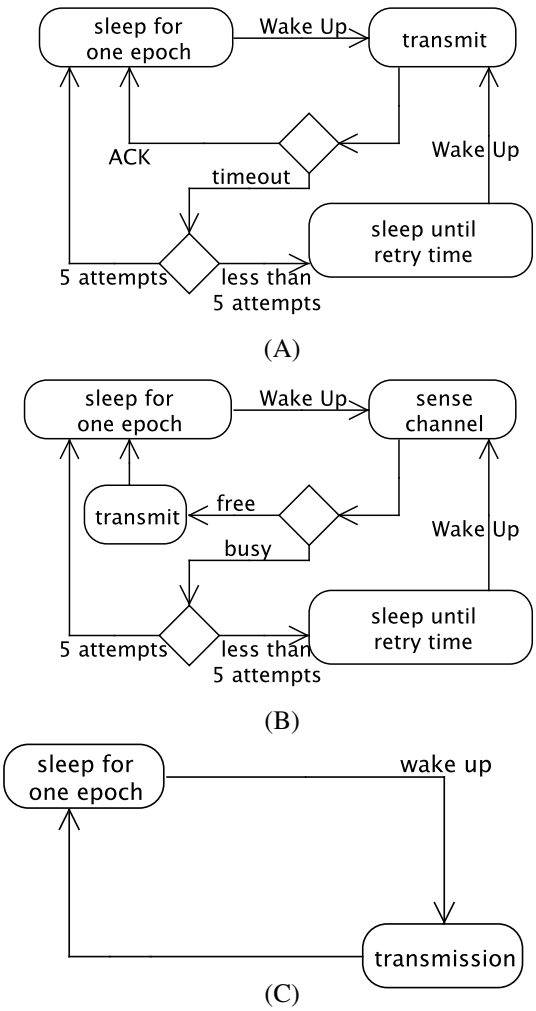


Fig. 2. State diagrams for (A)ACK, (B)carrier sensing, and (C)unidirectional heartbeat protocols.

1) Acknowledgement-based Protocol (ACK): In this protocol, a sensor transmits beacons at a random time slot and, to improve packet delivery reliability, has the basestation transmit an acknowledgement packet (or, ACK for short) for each packet it receives. After receiving the ACK, the sensor sleeps for an entire epoch to conserve energy, then wakes up and transmits another beacon with an incremented sequence number. If the sensor does not hear an ACK within a specified listen window it will retransmit the packet during the same epoch. The new retransmission time is calculated as

$$t_{retry} = t_{current} + rand() * (t_{epoch\_end} - t_{current})/2. \quad (1)$$

This randomly places the retransmission between the current time and halfway to the end of the current epoch, resulting in retransmissions more evenly spaced across the epoch and preventing bursts of transmissions near the end of the epoch. A sensor will stop retransmitting after 5 transmission attempts within a single epoch. Figure 2(A) summarizes the state

transition diagram for the ACK protocol.

The ACK packets must contain the ID of the sensor being acknowledged so the ACK packet is about the same size as the originally transmitted packet. Further, due to the varying delays affecting ACK packets and the transition from one radio mode to another, the listen window has to be a few μseconds longer than the time it spends transmitting the beacon. Consequently, the total energy spent transmitting a packet and then listening for an ACK is approximately twice that of transmission.

The energy consumption of the ACK protocol is dominated by the radio transmission and reception. Each beacon transmission consumes 20.4μJ, with 30 milliwatts being drawn for 80 μseconds during data transfer to the radio and 45 milliwatts being drawn for 400 μseconds during wireless transmission. Waiting for an ACK consumes 21.6μJ with 54 milliwatts of power drawn during an ACK listen window of 400 μseconds. Another 13.5μJ are spent turning on the radio and switching from transmit to receive mode. As a result, each beacon consumes a total of 55.5μJ. Consider a RollCall system with m transmitters, the epoch duration of N slots (a slot being the time unit here), and the transmission duration of k slots ($k < N$). A collision occurs if there is more than one transmission in the same slot i.e. a slot is occupied by either two or three or so on upto m transmitters. Assuming that a transmitter has already occupied a slot, the probability of having 1 or more transmitters, out of the remaining $m-1$, transmit in the same slot, P_{coll}, is calculated as

$$\begin{aligned} P_{coll} &= C_{m-1}^{1}(2\alpha)(1-2\alpha)^{m-2} \\ &\quad +C_{m-1}^{2}(2\alpha)^{2}(1-2\alpha)^{m-3} \\ &\quad +...+C_{m-1}^{m-1}(2\alpha)^{m-1} \\ &= 1-(1-2\alpha)^{m-1}, \end{aligned} \tag{2}$$

where $\alpha = \frac{k}{N}$. Given P_{coll} and the maximum retransmission count of 5, the expected number of total transmissions for a sensor per epoch is

$$N_{Tx} = \sum_{n=1}^{4} n(P_{coll})^{n-1}(1-P_{coll}) + 5(P_{coll}^{4}). \tag{3}$$

Therefore, the expected energy consumption for a sensor using the ACK protocol per epoch is $55.5N_{Tx}\mu\ J$. Finally, the probability for a sensor to successfully send a beacon within an epoch with up to 5 transmissions is

$$P_{succ/sensor} = \sum_{n=1}^{5} P_{coll}^{n-1}(1-P_{coll}) \tag{4}$$

We question the energy efficiency of this protocol in this application because listening for an ACK after each transmission more than doubles the energy cost of the transmission. Moreover, unnecessary retransmissions may occur when a packet is received by the basestation but the ACK is lost. Finally, since retransmissions will increase channel load they will lead to increased numbers of packet collisions.

2) Carrier Sensing Based Protocol (CS): This protocol attempts to avoid collisions by sensing the channel before any transmission. If the sensor determines that the channel is clear, it sends its packet and goes into sleep mode for the rest of the epoch. If the channel is not clear the sensor will choose another time to sense the channel again. The retry time is chosen using Equation 1 which spaces sensing attempts evenly across an epoch. Similar to the ACK protocol, a sensor will stop after 5 attempts within a single epoch. Figure 2(B) is the state transition diagram for the CS protocol.

In this protocol, counting the time it takes to bring the radio to full power in receive mode, a sensor consumes only 15.3μJ sensing the channel since the sensing time is less than 200μseconds. If the channel is clear, it will spend another 20.4μJ transmitting the beacon. When the channel is sensed it will be found busy with the probability P_{busy}, which can be calculated similarly as in Equation 2:

$$P_{busy} = 1-(1-2\alpha)^{m}, \tag{5}$$

where $\alpha = \frac{k}{N}$, and k is the number of slots a channel sensing takes. Since there is a delay for the radio to switch from sensing to transmission, there is a probability of colliding with another transmission even if the sensor only transmits after it senses the channel to be clear. This probability, P_{coll} is calculated as

$$P_{coll} = 1-(1-4\alpha)^{m-1}, \tag{6}$$

because the transition delay is roughly the same as the transmission time. Given P_{busy}, the expected number of channel sensing per sensor per epoch, N_{cs}, is calculated as

$$N_{cs} = (1-P_{busy})\sum_{n=1}^{4} nP_{busy}^{n-1} + 5(P_{busy}^{4}). \tag{7}$$

Similarly, the expected number of transmissions per sensor per epoch (note that some of these transmissions may end up in collision due to the delay for the radio to transit from channel sensing to transmission), N_{Tx}, is calculated as

$$N_{Tx} = 1-P_{busy}^{5}. \tag{8}$$

Therefore, the expected energy consumption for a sensor using the CS protocol per epoch is $E_{cs}N_{cs} + E_{Tx}N_{Tx}$, where E_{cs} is $15.3\mu J$ and E_{Tx} is $20.4\mu J$. Finally, the probability for a sensor to successfully send a beacon within an epoch with up to 5 attempts is

$$P_{succ/sensor} = (1-P_{busy})(1-P_{coll})\sum_{n=1}^{5} P_{busy}^{n-1}. \tag{9}$$

The CS protocol has the following issues. The transceiver cannot immediately switch from receive mode, where carrier sensing is performed, to transmit mode. This delay is comparable to the transmission time and reduces system capacity. This delay is also costly in terms of energy since the radio continues to draw power during the transition. Carrier sensing is also not 100% effective because sensing occurs at the tags but collisions occur at the basestations.

3) Time Division Multiple Access (TDMA): In TDMA systems, the epoch is divided into slots and each sensor transmits a beacon during an assigned slot. Advanced TDMA protocols have been proposed for ad hoc wireless networks [14][15][8] but none are suitable for this application. For example, TRAMA adopts energy-efficient methods for assigning slots during each period and for disseminating slot requests but such features are not necessary in the RollCall system due to the regular communication patterns. By combining carrier sensing with TDMA, Z-MAC [15] achieves very good performance in multi-hop networks with changing characteristics but at an energy cost that must be avoided in this system.

There are two simple ways to assign the slots that require little energy overhead for the sensors in this network. In the first method, the sensor queries the basestations multiple times to calculate the scaling factor between the basestation's clock and the sensor's local clock (e.g., $sf = (T_2^{bs} - T_1^{bs})/(T_2^{local} - T_1^{loacl})$), and then synchronizes its local clock based on this scaling factor. After the clock is synchronized, the sensor requests a time slot from the basestation. In the second method, the basestation transmits a packet during every "guard" period between slots. These packets serve dual purposes. First, they announce whether the following slot is occupied and provide the slot number. Second, by listening to several of these packets, a sensor can obtain the scaling factor and synchronize its local clock. After synchronizing its clock and identifying a free slot, a sensor can start transmission. An ACK from the basestation will further confirm ownership of the slot.

For a static system where the number of items/sensors is fixed and their locations are stationary, finding a slot using either of the above methods is a one-time action, and the resulting energy cost is trivial. However, in a more dynamic environment where the items are constantly moved in and out of a basestation's range, this cost rapidly accumulates, which can seriously degrade the system's lifetime or capacity. Most of the tracking applications we are interested in are dynamic because a static system does not require real-time tracking. Further, in order to maximize the capacity of the system, it is desirable to have small slots, but can entail a high clock synchronization overhead [16] because even slight drifts may result in misalignment. In this case, each sensor needs to listen for the timing packet from the basestation after its transmission to make sure its transmission is within the assigned slot. This leads to an energy consumption of 55.5μJ per sensor per epoch (the cost of a transmit and ACK) under a heavy load. The DESYNC based TDMA algorithm proposed in [14] does not require clock synchronization but its energy cost, due to large amounts of time spent listening for the timing information of neighbors, is prohibitive. As a result, current TDMA protocols are simply too expensive and inflexible for use in a system such as this one.

4) Unidirectional Heartbeat (Uni-HB): The previously discussed protocols are all concerned with improving packet delivery by spending additional energy, which has proven useful for traditional wireless communication systems. The fourth protocol, Unidirectional Heartbeat (Uni-HB), takes an opposite viewpoint, trying to cut down the energy consumption by accepting a slightly higher packet collision rate. The rationale behind this choice is that the reliability of individual packets is not as important as in other networks due to the inherent regularity of the communication: sensors transmit a beacon every epoch, and the beacons are short (2 bytes of payload). These properties naturally lead to a lower collision probability (which can compensate the higher collision rate from the protocol) and an easier recovery from collisions using software (as we shall detail in Section III-B).

In Uni-HB, the sensor does not get any feedback from the environment or from the basestation. It simply transmits a packet, sleeps for one epoch, and then repeats the process. As a result, two transmissions may collide with each other under a heavy load. To prevent a shadowing problem in which two sensors wake up and transmit at the same time repeatedly in every epoch, each sensor will sleep for a slightly different epoch. In this study, the exact epoch duration for a sensor with ID i, from a group of sensors with mean ID I, with a programmed epoch duration E is $[E + \frac{i-I}{10^6}]$seconds. This disperses the actual epochs of the sensors about the programmed epoch but guarantees that each epoch is slightly different, thus avoiding shadowing. Figure 2(C) is the state transition diagram for the Uni-HB protocol.

The energy usage for this protocol is very small, only costing 24.9μJ per sensor per epoch.

5) Comparison Results: In order to compare the energy consumption and packet reliability of these protocols, we implemented the protocols on the PIPs, and measured these two performance metrics as the load of the RollCall system increases. In implementing the protocols, we encountered two problems. The first problem is that we found it difficult to implement TDMA-like protocols on hardware that does not have a precise clocking mechanism, such as the PIP. Crystal oscillators are costly, even in large bulk, and therefore PIPs use an RC oscillator during sleep. Precisely timed TDMA slots are thus hard to implement on the PIPs and timing performance was poor. Since an efficient TDMA protocol requires either expensive hardware or energy costly clock synchronization we have deemed TDMA unsuitable, as discussed in Section III-A3, and have not included it in this evaluation.

The second problem lies in the limit of hardware availability. The ideal way of studying the performance is to fix the epoch duration (say, 1 second), and to scale up the number of sensors in the system. However, the number of PIPs we have is limited, and the programming time becomes prohibitive with large numbers. As a result, we fixed the number of PIPs to 25, and emulated an increasing load by scaling down the duration of an epoch. For instance, 25 PIPs using epochs of $250ms$ is approximately equivalent to 100 PIPs transmitting with one second epochs. All PIPs are positioned so that their average RSSI values fall in between $-60dBm$ and $-80dBm$, ensuring good, but not unrealistic packet reception rates. The PIPs are arranged in a half circle around a single basestation approximately 5 meters distant. In order to make sure that our treatment is valid, we also formulated the problem analyti-

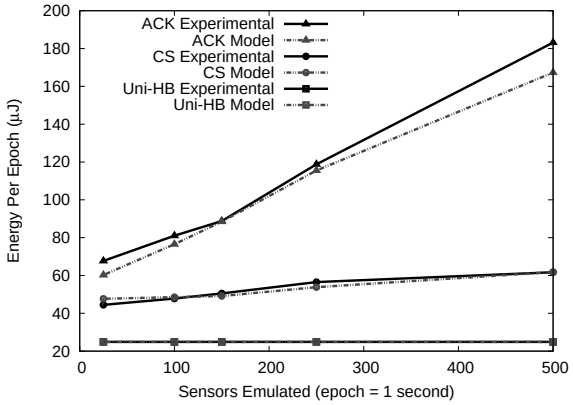


Fig. 3. The average energy consumption per epoch for different protocols. The solid lines are from the experimental measurements while the dotted lines are from the analytical model results.

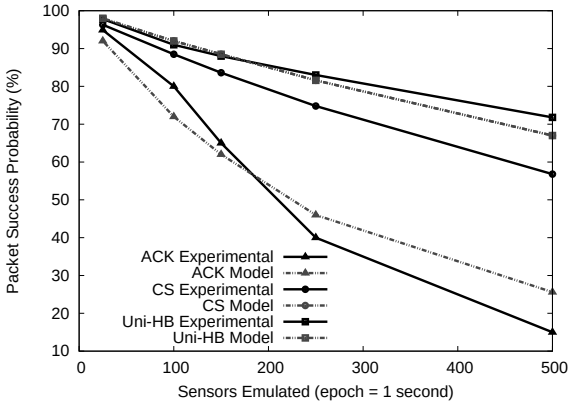


Fig. 4. The average probability of a packet successfully reaching the basestation for different protocols. The solid lines are from the experimental measurements while the dotted lines are from the analytical model results.

cally (in the previous section), and provide both experimental measurements and analytical results. Our analytical models incorporated the capture effect that is observed in a real system.

Figure 3 shows the average energy consumption per epoch for the three protocols when we emulate an increasing load. Figure 4 shows the average probability of a packet successfully reaching the basestation (i.e. packet delivery ratio) for the three protocols as the load is increased. We have the following observations. Firstly, the analytical results agree with the experimentation measurements closely across most of the cases. This confirms the correctness of both of our approaches. Secondly, ACK-like protocols are obviously not viable due to their high energy cost and low packet delivery ratio. Repeated transmissions in the ACK protocol can both increase the energy consumption and increase the load on the channel, thus leading to a poor delivery ratio. Thirdly, UNI-HB is the winner among all the protocols – it has lowest energy consumption and highest packet reliability.

Some of the results in Figure 4 are non-intuitive. The first one is that the measured packet delivery ratio of UNI-HB is *higher* than that of CS. This is caused by the following: (1) the RollCall beacons are so short that the benefit of channel sensing is not seen – the delay for the radio to switch from sensing mode to transmission mode is comparable to the packet duration so channel sensing will rarely predict the availability of the next time interval; (2) because carrier sensing is not always accurate in a noisy environment, the sensors avoided transmitting during "noisy" time intervals and were more likely to transmit at the same time. The second non-intuitive result is that the theoretical model results for CS and UNI-HB are very close to each other (their curves almost overlap). This is because the radio switch delay in CS is close to the transmission latency. Carrier sensing is only useful when the packets being sent are long.

B. Improving Reliability of Uni-HB Through Collision Prediction

In Uni-HB, each sensor sends out a beacon around the same time within each epoch, allowing the back-end processing software to accurately calculate the anticipated packet arrival times and to predict the collisions that will occur in the future epoch. Based on the predicted collision occurrence, the system can correct the received packet sequence from each sensor by marking the unreceived collided packet as received, thus leading to a higher packet delivery ratio. In this section, we discuss how this collision prediction process works.

A typical Rollcall system consists of hundreds to thousands of PIPs, tens of basestations, and one central processing station (CPS). Data flows, in the form of packets, from the PIPs to the basestations, and from there to the CPS. Basestations are deployed to have overlapping coverage, allowing each PIP to be heard by at least two basestations. Basestations listen for packets from the PIPs and place a timestamp on each packet before sending the received packet (together with the timestamp) to the CPS over an ethernet, WLAN, or USB connection. Timestamps from the basestation are most accurate when it merely places a time stamp on received packets and forwards them to the CPS. The timestamps should give twice the granularity of transmit events, so a $1/5$millisecond resolution is recommended for the $2/5$millisecond transmit times. The CPS will receive a stream of data from each basestation with each data entry consisting of the fields: source basestation ID, basestation timestamp, source PIP ID, packet sequence number, RSSI value, and CRC status.

The CPS first tries to synchronize all the timestamps from different basestations, and merge all the data entries into a single *arrived* queue according to the adjusted timestamps with all the redundant packets from the same PIP (but received by different basestations) removed. In addition to the arrived queue, the CPS also maintains the *expected arrival* queue, which includes the expected arrival time for each PIP in the system. Both queues are sorted in the increasing temporal order. The actual processing involves taking the first packet in the arrived queue and searching for its PIP ID in the expected arrival queue. Usually, this search only needs to check the first few expected arrival events in the queue. If the PIP is found, the CPS updates its expected arrival time, $T_{expected} = T_{now} + T_{epoch}$ and moves the event to the

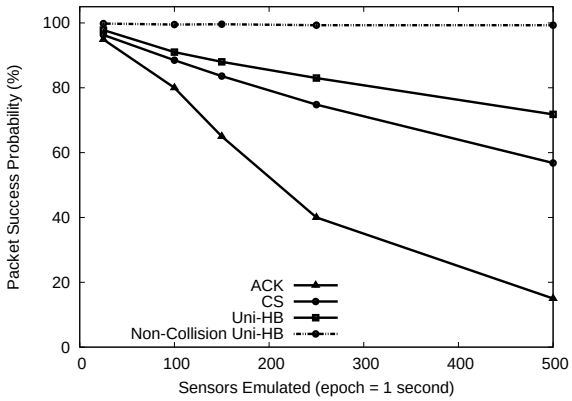


Fig. 5. The success rate per packet including the Uni-HB results with collision prediction and correction.

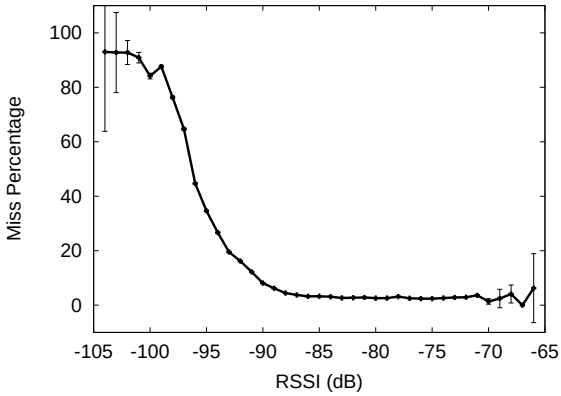


Fig. 6. Experimentally derived ambient miss percentage versus the RSSI, with the 95% confidence interval for each point.

appropriate location in the queue (most likely the end). If the search has to go through the entire queue and cannot find the corresponding PIP ID, it means that this is a new PIP and the new ID is added to the expected arrival queue.

In such a system, collisions can be easily predicted. Every time when an updated expected arrival event is relocated to a new position in the queue, we check its expected arrival time with the expected arrival times of the neighboring events in the queue. If the gap between two expected arrivals is less than the *hazard window*, we mark these two packets as being potential collision hazards. If the packet with a hazard mark is not received around its expected time, the CPS assumes it was a collision and the item/PIP is actually not missing. Each PIP's epoch duration slightly varies from each other so no two packets will transmit at the same time for a large number of epochs. Here, the size of the hazard window is an important parameter. A hazard window which is too small will under-predict collisions, while a hazard window which is too large will over-predict and might mistakenly treat packet loss due to other sources (e.g. the item missing) as collisions and ignore them. In deciding its value, we considered factors such as the beacon transmission latency, the time stamp resolution, and the clock drift over several missed packets when the exact packet transmission times are lost.

After applying the collision prediction technique as described above, we re-calculate the packet delivery probability for non-collision packets in Uni-HB and show the new results in Figure 5. The results show that the adjusted success probability for Uni-HB is close to 100% – less than 1% of losses were not predicted and corrected in this way. Note that the collision prediction technique cannot be applied to protocols like CS and ACK because their packet arrivals are not regular due to their feedback or collision avoidance mechanisms.

IV. Robust Detection of PIP Presence From Packet Loss

The ultimate goal of the RollCall system is to rapidly and reliably detect when an item is missing. If everything in a RollCall system worked ideally, a packet would arrive every epoch from every PIP in the system. If a sensor is lost or broken, then it ceases beacon transmissions and the system would realize immediately that the sensor has been stolen or broken. In reality though, one cannot, and should not, simply equate a missing packet to a missing sensor because packet loss occurs for other reasons. As a result, a robust detection algorithm tries to differentiate packet losses due to missing sensors and packet losses due to other causes.

A. Ambient Packet Loss

There are two kinds of ambient packet loss in RollCall, those caused by low received signal intensity being indistinguishable from background interference and thermal noise, referred to as *ambient loss*, and those caused by direct collisions between packets in the system, referred to as *collision loss*. As discussed in Section III-B, a large fraction of the collision loss can be identified by the CPS through the collision prediction technique. In this section, we will mainly focus our discussion on ambient packet loss.

Ambient loss defines the baseline packet loss for a sensor. Its magnitude varies signals from interfering RF devices (such as other wireless devices, microwave ovens, and electric motors) and radio propagation effects. The typical ambient loss rate observed during our trials and deployments is less than one percent. Even poorly placed sensors can communicate effecively if basestations are deployed so that the multiple basestations receive the sensor's packets.

B. Detection Algorithms

A good detection algorithm will identify that contact has been lost with a sensor quickly enough for a person to react to the problem but will not have so many false positives that the person who has to respond to the alarm will stop taking it seriously. In this paper, we measure the effectiveness of the detection algorithm of the RollCall system using two metrics: the probability of raising a false alarm – a false alarm means the system declares a sensor/item missing while the sensor is still present – P_f, and the time it takes for the system to raise an alarm after a sensor ceases transmitting packets, T_{alarm}. In this section, we consider detection based on the resulting packet sequence from the Uni-HB protocol with

collision prediction. As a result, packet misses observed by the CPS will be either due to ambient loss or a missing PIP.

There is a tradeoff between P_f and T_{alarm}. If we sought to minimize the detection latency T_{alarm} without considering P_f, then we would use a naïve technique where a sensor is considered lost whenever a single packet is missed. This gives the smallest possible T_{alarm} (1 epoch) at the expense of a very high P_f.

At the other extreme, if we only focus on minimizing P_f without regard to T_{alarm}, then we can simply wait very long time periods before reporting that a contact has been lost with a sensor. While waiting for five minutes to pass without seeing a single packet from a sensor will give a very low P_f which will be too long for anyone to respond to an item's loss after it is detected. As a result, this extreme is not desirable as well.

In this study, we explored the following detection heuristics:

1) Detection Based on Historical Maximum Miss Chain Length (Detect-MaxMiss).: Ambient packet losses often occur in bursts because the ambient factor may last for a period of time. In *Detect-MaxMiss*, for each sensor, we keep track of the length of the longest miss chain. The baseline detection is simple: once the current miss chain is longer than the previously observed maximum miss chain length, we declare the sensor is missing. In order to avoid the high false alarm rates when the miss chains are short, we declare a sensor is missing if the current miss chain length is at least K misses more than the previous maximum chain length. K can be determined heuristically. We call this *Detect-MaxMiss+K*.

2) Detection Based on RSSI (Detect-RSSI): . In this algorithm, we first have a training phase to build a mapping between the RSSI value of the current received packet and the likelihood that the next packet from the same sensor will be missing. For example, we find that if the current packet's RSSI value is low, then the probability to have an ambient loss in the following epoch is high. Figure 6 shows the ambient loss ratio with RSSI histogram we collected in our lab.

In the test phase, we detect whether a sensor is missing based on the ambient loss ratio with RSSI histogram. Next, let us look at an example to understand the detection algorithm. Suppose we have the following packet sequence for sensor i:
`Recv(-97dB), Miss, Recv(-95dB), Miss, Miss, ...`
Then the detection algorithm works as follows. In the first epoch, since the packet is received, the probability of the sensor missing $P_{missing}$ =0. In the second epoch, the packet is lost, but from the last RSSI value (-97dB), we know that the likelihood of having an ambient loss in this round is roughly 0.8. As a result, we estimate $P_{missing} = 0.2$. Since $P_{missing}$ is below the preset threshold 0.8, we do not raise an alarm. In the third epoch, again we have $P_{missing} = 0$. In the fourth epoch, the packet is lost, and $P_{missing} = 1 - 0.3 = 0.7$ based on the previous RSSI value. In the fifth epoch, the packet is again lost, and we calculate $P_{missing} = 1-(1-0.7)^2 = 0.91$. Here, since we do not have the RSSI value in the last round, we have to borrow the value two rounds ago to estimate $P_{missing}$. At this point, we will report the sensor being missing because $P_{missing}$ is above the threshold value. We note that we could also calculate $P_{missing}$ in the fifth epoch based on the likelihood of having 2 successive ambient losses following a received packet with RSSI value of -95dB. We will build such histograms in our future work.

C. Detection Results

We tested our detection algorithms using the trace that was collected using 31 PIPs over the period of a week. The PIPs were arranged in groups of three and were placed across several rooms in an office environment. Some of the PIPs were placed in metal bins and drawers. One PIP was intentionally placed so far away from the basestations that the basestation was on the edge of its radio range and its observed signal was extremely weak. During the testing week, those PIPs were moved, sometimes outside of the radio range of the basestation to test the system when an item is slowly removed, or had their batteries removed to test loss detection.

Table I shows the average P_f for the two proposed detection algorithm together with the naive algorithm *Detect-SingleMiss*. Here, P_f is defined as the ratio of the number of times the system declares a sensor is missing with respect to the number of miss chains in the trace. Since *Detect-SingleMiss* assumes a missing sensor every time there is a missed packet, the value of P_f is 100%. Compared to *Detect-SingleMiss*, both of our algorithms have substantially reduced the false alarm rate.

On the other hand, it is not meaningful to look at the average T_{alarm} over all the sensors as some of them were purposefully placed to have very poor signals. As a result, it is much harder to report that they are missing in a timely fashion – how can a police officer quickly detect that a person is missing if he only leaves his secret vault once a month? In this case, we look at T_{alarm} for each individual PIP. Figure 7 plots the percentage of packet misses due to ambient loss for each of the 31 PIPs, showing that three of the PIPs had very poor transmissions. Figures 8(a) and (b) show the resulting individual T_{alarm} for the two detection algorithms. From these results, we can clearly see that PIPs that have a poor ambient environment need a long period to be declared missing, while PIPs with a reasonable ambient environment can be reported missing in a much more timely fashion (around 10 epochs). This suggests that when deploying a tracking system, we need to make sure all the PIPs are covered by basestations. This will not cause any issue in the system we consider because the basestations share the same hardware as the PIPs, and as a result, share the same cost as well. Thus, having more basestations is easy to achieve. Further, we observe that the *Detect-RSSI* algorithm is more robust in rapidly changing environments and when sensors experience poor signal quality. The *Detect-MaxMiss+5* algorithm will adjust to a changing environment as its history window updates, but since radio environments can change rapidly the *Detect-RSSI* algorithm has an advantage.

V. Concluding Remarks and Future Direction

This paper studies energy-efficient communication protocols and robust detection algorithms that are essential to move from the asset tracking systems of today towards long-lived,

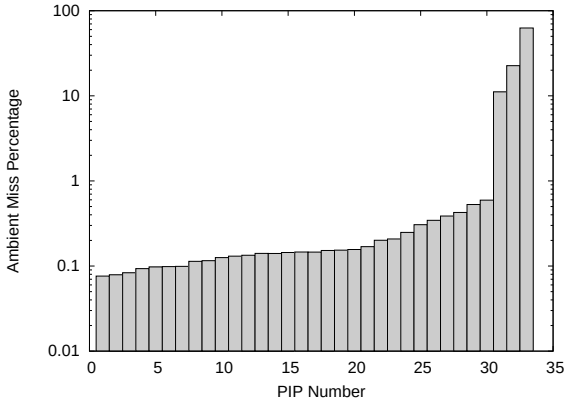
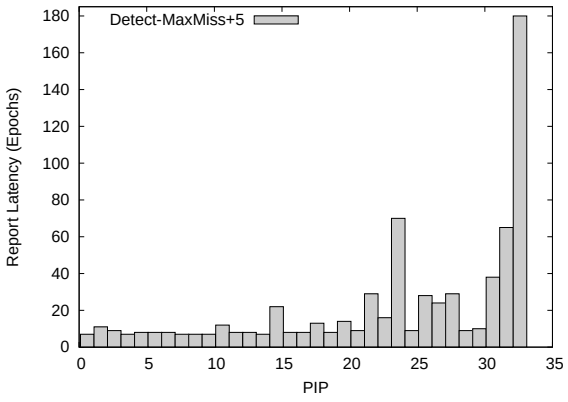


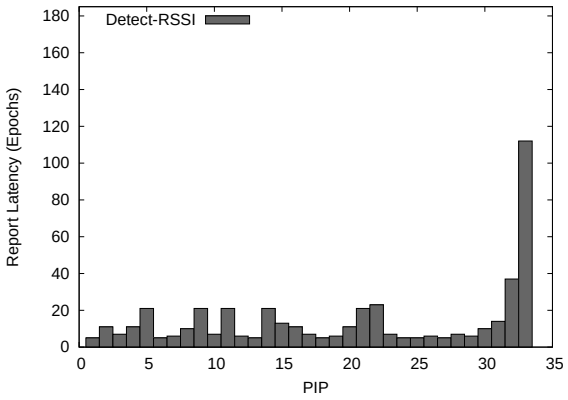
Fig. 7. The ambient miss percentages seen by the PIPs during the detection trial.

fail-safe presence assurance systems. Using the techniques discussed in this paper, we can extend the lifetime of a tracking system to 327 days using a coin cell battery, regardless of the number of items in the system. Further, for those items that are placed in harsh radio environments or reasonably close to the basestations, our detection algorithm can report a missing item within 20 epochs with a false alarm rate of .011% (per ambient loss) in the test environment.

However, this is only the first step towards deploying meaningful system. Future work will consist of reducing long miss chains from ambient interference by using dynamic spectrum techniques. We would also like to investigate the real-world issues faced during actual large-scale deployment in more realistic settings.



(a)



(b)

Fig. 8. T_{alarm} for each PIP in the detection trial: (a) $Detect - MaxMiss$, and (b) $Detect - RSSI$.

REFERENCES

[1] R. Want, "An introduction to rfid technology," *Pervasive Computing, IEEE*, vol. 5, no. 1, pp. 25–33, Jan.-March 2006.

[2] M. Gaynor, S. Moulton, M. Welsh, E. LaCombe, A. Rowan, and J. Wynne, "Integrating wireless sensor networks with the grid," *IEEE Internet Computing*, vol. 8, no. 4, pp. 32–39, 2004.

[3] M. Malinowski, M. Moskwa, M. Feldmeier, M. Laibowitz, and J. A. Paradiso, "Cargonet: a low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asychronous monitoring of exceptional events," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2007, pp. 145–159.

[4] Y. Zhang, G. Bhanage, W. Trappe, Y. Zhang, and R. Howard, "Facilitating an active transmit-only rfid system through receiver-based processing," *Sensor, Mesh and Ad Hoc Communications and Networks, 207. SECON '07. 4th Annual IEEE Comunications Society Conference on*, pp. 421–430, 18-21 June 2007.

[5] G. Bhanage, Y. Zhang, Y. Zhang, T. Wade, and R. Howard, "Rollcall : The design for a low cost and power efficient active rfid asset tracking system," in *Eurocon*, 2007.

[6] B. Firner, S. Medhekar, Y. Zhang, R. Howard, W. Trappe, P. Wolniansky, and E. Fenson, "Asset tracking with roll-call: Design, deployment, and analysis," in *Proceedings of HotEmNets*, June 2008.

Method	P_f (%)
$Detect - SingleMiss$	100
$Detect - MaxMiss + 5$	.013
$Detect - RSSI$	.011

TABLE I
AVERAGE P_f FOR DIFFERENT DETECTION ALGORITHMS.

[7] Intel, "Intel motes," http://techresearch.intel.com/articles/exploratory/1503.htm.

[8] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," *Wireless Networks*, vol. 12, no. 1, pp. 63–78, 2006.

[9] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 95–107.

[10] Silicon Laboratories, *C8051F320/1*, http://www.silabs.com/ mcu/.

[11] Texas Instruments Norway AS, *CC1100 Single Chip Low Cost Low Power RF Transceiver*, June 2006, data Sheet (Rev.1.1).

[12] "Panasonic," http://www.panasonic.com/.

[13] A. El-Hoiydi and J. Decotignie, "Low power downlink mac protocols for infrastructure wireless sensor networks," *Mobile Networks and Applications*, vol. 10, no. 5, pp. 675–690, 2005.

[14] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "Desync: self-organizing desynchronization and tdma on wireless sensor networks," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2007, pp. 11–20.

[15] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-mac: a hybrid mac for wireless sensor networks," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2005, pp. 90–101.

[16] S. PalChaudhuri, A. K. Saha, and D. B. Johnson, "Adaptive clock synchronization in sensor networks," in *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*. New York, NY, USA: ACM, 2004, pp. 340–348.