SLAM: Sleep-Wake Aware Local Monitoring in Sensor Networks

Issa Khalil, Saurabh Bagchi, Ness B. Shroff
Dependable Computing Systems Lab (DCSL) & Center for Wireless Systems and Applications (CWSA)
School of Electrical & Computer Engineering, Purdue University
Email: {ikhalil, sbagchi, shroff}@purdue.edu

Abstract

Sleep-wake protocols are critical in sensor networks to ensure long-lived operation. However, an open problem is how to develop efficient mechanisms that can be incorporated with sleep-wake protocols to ensure both long-lived operation and a high degree of security. Our contribution in this paper is to address this problem by using local monitoring, a powerful technique for detecting and mitigating control and data attacks in sensor networks. In local monitoring, each node oversees part of the traffic going in and out of its neighbors to determine if the behavior is suspicious, such as, unusually long delay in forwarding a packet. Here, we present a protocol called SLAM to make local monitoring parsimonious in its energy consumption and to integrate it with any extant sleep-wake protocol in the network. The challenge is to enable sleep-wake in a secure manner even in the face of nodes that may be adversarial and not wake up nodes responsible for monitoring its traffic. We prove analytically that the security coverage is not weakened by the protocol. We perform simulations in ns-2 to demonstrate that the performance of local monitoring is practically unchanged while listening energy saving of 30 to 129 times is achieved, depending on the network load.

Keywords: Sensor networks, local monitoring, sleep/wake techniques, wake-up antenna.

1. Introduction

It has been shown in the literature that sensor networks are vulnerable to a wide range of security attacks including the wormhole attack, rushing, and Sybil attacks [25]. Cryptographic mechanisms alone can not prevent these attacks since many of them such as the wormhole and the rushing attacks can be launched without needing access to cryptographic keys or violating any cryptographic check. To mitigate such attacks, many researchers have used the concept of cooperative *local monitoring* within a node's neighborhood ([17]-[24]). In local monitoring, nodes oversee part of the traffic going in and out of their neighbors. Different types of checks are done locally on the observed traffic to make a determination of malicious behavior. For the systems where arriving at a common view is important, the detecting node initiates a distributed protocol to disseminate the alarm. Many protocols have been built on top of local monitoring for intrusion detection (e.g., [16][17]), building trust and reputation among nodes (e.g. [20][21]), protecting against control and data traffic attacks (e.g. [22]-[25]) and in building secure routing protocols (e.g. [18][19][23]). Specifically, in [23] and [24] the authors have presented a technique for detection of control and data attacks in ad-hoc networks using local monitoring. Control attacks are launched by a node delaying, dropping, modifying, or fabricating control traffic that it is supposed to forward. Data attacks are similarly launched by performing these actions on data traffic. In [23] and [24], these attacks are detected by a group of nodes, called *guard nodes*, that perform local monitoring. The guard nodes are normal nodes in the network and perform the basic operations of sensing, in addition to monitoring.

Though local monitoring has been proposed by many researchers, it incurs a high energy cost since it requires the guard nodes to be awake all the time to oversee network behavior. To the best of our knowledge, no one has devised sleep-wake protocols for optimizing the energy overhead of monitoring while maintaining the quality of the monitoring service. This is the problem we address in this paper. The main challenge lies in having the sleep-wake performed securely so that an adversarial node cannot escape detection by causing its guard nodes to stay asleep.

In this paper we propose a set of mechanisms called Sleep-Wake Aware Local Monitoring (SLAM) that adapt the existing local monitoring technique to significantly reduce the time a node needs to be awake for the purpose of monitoring. The proposed mechanism adapts itself depending on the kind of sleeping protocol used in the network, henceforth referred to as the *baseline sleeping protocol* (BSP). For networks that use synchronized sleeping algorithms (e.g., [3] [8]-[10]), i.e., nodes wakeup and go to sleep in a synchronized manner, SLAM does not need to do anything since a node and its guards will be woken up by the BSP itself. There exist several application-specific sleeping algorithms, for example, to maintain a given sensing coverage (each point should be sensed by at least k nodes) or a given network connectivity level (each pair of nodes should have k disjoint paths). For these protocols (e.g., [1]-[3]), SLAM can support local monitoring by modifying an input parameter to the existing sleeping algorithm, such as the value of k in the connectivity or coverage preserving BSPs. Finally, we consider networks that use on-demand sleep-wake. On-demand sleep-wake means a node wakes up when it needs to communicate, and since the communication pattern can be arbitrary, the wakeup time is arbitrary in the general case. This provides the most challenging case and forms the most significant portion of the discussion in the paper.

For the third class of network, SLAM provides a generic on-demand sleeping algorithm, called On-Demand SLAM. This algorithm relies on each node having a passive or a low-power wake-up antenna in addition to the normal antenna. A node that is not involved in network activities, such as, data forwarding is ordinarily sleeping according to the BSP. However, for monitoring purposes, it is woken up on demand by a neighboring node using the wake-up antenna. The key challenge to this apparently simple scheme is that it now opens up the possibility of a new adversarial action, namely, a node not waking up a sleeping node(s) so that its own malicious action is not detectable. At a high level, our solution involves the following steps–adding to the list of behavior that a guard node needs to check and second, defining the mechanism through which the check is to be done (i.e., who checks, when, and for what).

We provide a theoretical analysis for energy saving using On-Demand SLAM compared to a baseline monitoring protocol [24]. We build a simulation model for SLAM using *ns-2* and perform a comparative evaluation of local monitoring with and without SLAM. The results show that the security of local monitoring is very close in both cases while the overhead of SLAM in terms of listening energy is between 30 to 129 times lower, depending on the network traffic. The results show the effect of the number of malicious nodes, the traffic load, and the fraction of data being monitored on the overhead of local monitoring. We summarize our contributions in this paper as follows:

1. We provide a technique for conserving energy while performing local monitoring without significantly degrading its security performance. This we believe is fundamental to deploying local monitoring in any energy conscious network.
2. We propose a generic on-demand sleep-wake algorithm for network monitoring in scenarios where either no BSP exists or the sleep-wake is based on arbitrary communication pattern.
3. We analytically prove that SLAM does not add any vulnerability to the existing local monitoring technique.
4. We show through simulations a significant reduction in monitoring cost with negligible degradation in the monitoring quality of service.

The rest of the paper is organized as follows. Section 2 presents related work in the field of sleep-wake protocols. Section 3 describes SLAM. Section 4 presents mathematical analysis of the energy overhead and security of SLAM. Section 5 presents the simulation experiments and results. Section 6 concludes the paper.

2. Related Work

Node sleeping is an important mechanism to prolong the life time of sensor networks. This topic has been discussed extensively in the literature and many protocols have been proposed for various types of applications such as object tracking ([1][2]). It has been realized that under current hardware designs, the maximum energy savings can be achieved through putting nodes to sleep—three orders of magnitude less current draw than in an idle node for the popular Mica mote platform for sensor nodes.

Primarily three different mechanisms are used to put nodes to sleep. The *first* is called *synchronized wakeup-sleep scheduling* in which the nodes in the network are put to sleep and woken up at the same time in a centralized (e.g., [10]) or a distributed manner (e.g. [3][8][9]). A disadvantage of such protocols is that the duty cycle is application dependent and not known *a priori*. Most importantly, they require the network to have an accurate time synchronization service. Furthermore, in scenarios with rare event detection, no event happens and the nodes enter sleep mode again in most of the wakeup periods. This means that nodes wake up too often resulting in wastage of energy. The *second* mechanism is based on selecting a subset of nodes to be woken up to *maintain some properties in the network*, such as sensing coverage (e.g., [4][5]), network connectivity (e.g., [3][6]), or both coverage and connectivity (e.g. [7]).

The *third* mechanism is based *on-demand sleep-wake* protocols. These on-demand sleep-wake protocols use either special purpose low-power wake-up antennas (e.g., [11]-[14]) or passive wake-up antennas [15]. These antennas are responsible for receiving an appropriate beacon from a neighbor node and waking up the node for its full operation. Thus, for environments where events of interest are relatively rare, the time for the low power operation with the wake-up antennas being on, dominates. Further details about the operation of the antennas are mentioned in Section 3.3.

To the best of our knowledge, we are the first to address local monitoring in a network where nodes may need to be put to sleep for energy conservation.

3. SLAM Protocol Description

The primary goal of SLAM is to minimize the time a node has to be awake to perform local monitoring. Local monitoring is used to make sure that packets are not dropped, delayed, modified, misrouted, or forged along the path from source to destination [23]. SLAM adds one more task to the list of events that a guard node needs to monitor—verifying whether the node being monitored wakes up the requisite guards or fails to do so due to malicious motivations.

3.1. System Model and Assumptions

SLAM assumes that the network is static and the links are bi-directional. SLAM requires a pre-distribution pair-wise key management protocol (e.g. [26]) such that any two nodes can acquire a key for encryption and authentication. In

On-Demand SLAM, each node is equipped with either a passive [15] or a low-power wakeup antenna [12]. Any two nodes that need to communicate, establish a route between them using an underlying routing protocol. We assume that the source node is honest. No assumption is made about the adversary nodes following the sleep-wake protocol, only the honest nodes follow it. Each node knows its first-hop neighbors and the neighbors of each neighbor, e.g., using a technique as in [23]. The malicious behavior of fruitlessly sending a wake-up signal to a node is not addressed since this potential exists in any on-demand wake-up protocol and SLAM neither exacerbates nor solves this problem.

3.2. Different Network Models for SLAM Protocol

Depending on the BSP used in the network, SLAM has three different mechanisms for proposing sleeping for networks with local monitoring—The *No-Action-Required* SLAM protocol, the *Adapted* SLAM *protocol*, and the *On-Demand* SLAM protocol. The No-Action-Required SLAM is applicable in networks with synchronous sleep-wake mechanisms. Examples of such protocols include Span [3], S-MAC [8], and habitat monitoring [10]. The guards for the communication would also be woken up since the guards are one-hop neighbors of the two nodes that form the link on which the communication is taking place. The Adapted SLAM protocol is applicable in networks with application-specific sleep-wake protocols that can be adapted to wake up and sent to sleep guards as well. Examples of such protocols include those that maintain a *k*-sensing or a *k*-communication coverage for given values of *k* [4][5] [7]. The adaptation process depends in the protocol itself, but for the connectivity or coverage problems, it involves increasing the value of *k* input to the protocol such that the requisite number of guards is awake in any part of the sensor field. For more details on these two kinds of networks, please refer to Sections 3.2 and 3.3 of our technical report [29].

3.3. The On-Demand SLAM Protocol

This protocol is used in a network that either has no BSP in operation or employs on-demand sleep-wake protocols. Therefore, we build a new sleep-wake protocol, called On-Demand SLAM that enables the guards to go to sleep when not required for monitoring. The approach we take is on-demand sleep-wake of the guards rather than scheduling the sleep-wake periods. The defining characteristic of on-demand sleep-wake protocols is that any node in the network may, at random, initiate communication with any other node in the network. The sleep-wake protocol does not rely on any fixed communication pattern in the network. On-Demand SLAM uses either low-power wake-up antennas (e.g., [11]-[14]) or passive antennas with circuitry that can harvest signal energy to trigger a node to wake up [15], as has been described in Section 2. These kinds of antennas are commercially available (e.g. [14]) as well as available as research prototypes in academia [15]. For example Austria Microsystems provides a low-power wake-up receiver (AS3931) with data rate of 2.731KB/s and current consumption in standby mode of 6.6μA [14].

In On-Demand SLAM, the low-power wake-up radio remains awake all the time while the normal radio is put to sleep when it is not sending or receiving data or is not required for monitoring. If a node is to send a packet out, it simply wakes up by itself; if a neighbor node is to send a packet to this node, the sender will send a short wake-up beacon using the wake-up radio channel, and on receiving this beacon the wake-up radio triggers the normal radio to be ready for the reception. The main disadvantage of the mechanism is that it still consumes extra energy. Even though the power consumed is small compared to the normal antenna (1uW compared to 10mW in [11]), the energy is non-negligible due to long time of operation.

Hence this mechanism has been modified to use passive wake-up antennas, known as radio-triggered power management mechanisms [15]. In this mechanism a special hardware component–a radio-triggered circuit–is connected to one of the interrupt inputs of the processor. The circuit itself does not draw any current and is thus passive. The node can enter sleep mode without periodic wake-up. The wake-up mode is the usual working mode with all the functional units ready to work, and the average wake-up mode current is 20mA. In sleep mode, a node shuts down all its components except the memory, interrupt handler, and the timer. The sleep mode current is 100μA. When a network node changes from sleep mode to wake-up mode, there is a surge current of 30mA for a maximum of 5ms. When a power management message is sent by another node within a certain distance, the radio-triggered circuit collects enough energy to trigger the interrupt to wake up the node. Except for activating the wake-up interrupt, the radio-triggered circuit is independent of any other components on the node. If supported by hardware, the wake-up packet is sent at a special radio frequency. If the nodes in a one hop neighborhood have unique frequencies each listens on, then other communication at a different radio frequency does not wake up the nodes. Note that hardware cost for adding multiple-frequency support is usually fairly low. Many recent low-end radio transceivers support multiple frequency operations [30] However, the unique frequency assignment is not necessary for the correctness of On-Demand SLAM, but improves the energy efficiency. In the rest of the paper, for ease of exposition we use the term "low-power wake-up radio" to mean either the low-power wake-up hardware or the passive wake-up hardware.

3.3.1. On-Demand SLAM: Basic Approach

The basic idea in designing On-Demand SLAM is for a node to wake up the requisite guard nodes to perform local monitoring on the communication that is going out from that node. The challenge in the design comes from the fact that any of the nodes (except the source) may be malicious and therefore, may not faithfully wake up the guards. As in [23] and [24], local monitoring is used to mitigate malicious activities manifested through dropping, delaying, modifying, or forging of data/control packets. In local monitoring, the sensor node is called a *guard* when performing traffic overhearing and monitoring of neighbors. The guards of a node A over the incoming packets from a transmitter X are the common neighbors of X and A. In Figure 1, α_1 and β_1 are the guards of n_1 over the link $S \rightarrow n_1$. Information for each packet sent from X to A is saved in a *watch buffer* at each guard for a time T_w. The information maintained depends on the attack to be detected (i.e., drop, delay, modify, or forge).

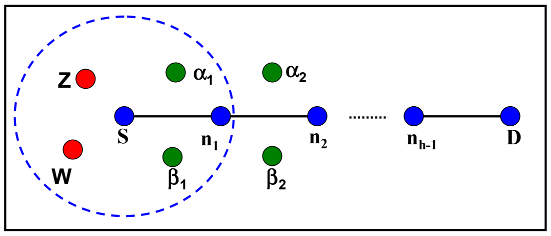

Figure 1: *h*-hop route between S and D, neighbors of S, and guards of n_1 and n_2

A malicious counter (*MalC(i,j)*) is maintained at each guard node, i, for every node, j, which i is monitoring. *MalC(i,j)* is incremented for any suspect malicious activity of j that is detected by i. To account for intermittent natural failures, a node is determined to be misbehaving, only if the *MalC* goes above a threshold. When *MalC(i,j)* crosses the threshold, node i isolates node j by refraining from sending or receiving any packet from j. Node j is said to be fully isolated from the network when all its neighbors isolate it.

We use Figure 1 to explain On-Demand SLAM. A source node S is sending data to a destination node D through an h-hop route $S \rightarrow n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_{h-1} \rightarrow D$. In a network where all the nodes are honest, S will wake up the next hop n_1 and the guard nodes (α_1 and β_1) before sending the packet to n_1. In turn n_1 will wake up n_2 and guard nodes α_2 and β_2 before sending the packet on the next hop and so on, till the packet reaches D. Formally, according to [24], the responsibility of a guard node α of n_{i+1} over a link $n_i \rightarrow n_{i+1}$ is to verify that:

1. n_{i+1} forwards the packet within time T_w.
2. n_{i+1} does not modify the packet it is forwarding.
3. n_{i+1} only forwards a packet if a packet is sent on the $n_i \rightarrow n_{i+1}$ link.

SLAM introduces a fourth responsibility.

4. n_{i+1} should wake up the guards for the $n_{i+1} \rightarrow n_{i+2}$ link *before* forwarding the packet on that link.

If a rule 1-3 is violated then the *MalC* value is incremented by appropriate amount; if rule 4 is violated, the *MalC* value increment is the maximum of the other MalC values because this rule violation may be used to mask violations of any of the rules 1-3.

In general, for any multi-hop route connecting a source node S to a destination node D, S is responsible for waking up the correct guards for n_1, and n_i is responsible for waking up the correct guards of n_{i+1} ($1 \leq i \leq$ h-2). The correct guards for n_1 are guaranteed to be woken up by the assumption of honest source S and whether n_i honestly wakes up the next hop guards is monitored by the guards of n_i according to rule 4 above.

In the following we present two variations of On-Demand SLAM depending on the wake-up mechanism a node follows to wake up the guards of the next-hop.

3.3.2. Guards-Only On-Demand SLAM (*G-SLAM*)

The high level design goal in G-SLAM is to minimize the energy wasted in waking up nodes that can not serve as guards. On average half of the nodes within a single transmission range are not guards over a certain link (according to Equation I in [24]). In G-SLAM, a node wakes up a subset of its neighbors—the nodes that *can* act as guards. For this, it is assumed that the wake-up antenna of each node is tuned to receive at its own code (as in [15]), which is distinct for all one-hop neighborhood nodes.

For a guard node to verify honest wake-up, G-SLAM requires each node in the network to know, in addition to the identities of its first-hop and second-hop neighbors that are required by local monitoring, the location of each node within twice its transmission range. In Figure 1, a guard of n_1, say α_1, knows the location of its neighbor n_1 and the location of all the neighbors of n_1 (S, β_1, β_2, α_2 and n_2). Using this information, α_1 knows the common neighbors of n_1 and n_2, α_2 and β_2, which can act as the guards of n_2 over the link $n_1 \rightarrow n_2$. Therefore, α_1 can not be deceived by n_1 waking up its nodes that can not be guards for n_2 (S,β_1). A disadvantage of G-SLAM is that it requires sophisticated wakeup hardware that can be addressed using an id-attached beacon [15].

We explain the G-SLAM algorithm with the help of Figure 1. Assume that node S has some data to be sent for the destination D over the route $S \rightarrow n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_{h-1} \rightarrow D$ connecting S to D. G-SLAM uses the following steps to wake up the correct guards along the route from S to D:

1. Node S sends a signal to wake up the first-hop node (n_1) and the guards for n_1 (α_1, β_1). This is a multicast signal that contains the identities of n_1, α_1, and β_1.
2. Node S sends the packets it has to n_1 following the timing schedule presented in Section 3.3.4.
3. Nodes n_1, α_1, and β_1 after being woken up continue to remain awake for T_w. T_w is a parameter of local monitoring that captures the maximum time by which an entry in the watch buffer is evicted (beyond that is evidence of malicious action). Each time a new packet is sent from S to n_1, T_w is reinitialized. After T_w expires at a node, it goes back to sleep.
4. Node n_1, after being woken up, uses the timing schedule in Section 3.3.4 and according to it sends a wake-up signal for n_2, the guards of n_2 over the link $n_1 \rightarrow n_2$ (α_2, β_2), and the guards of n_1 over the link $S \rightarrow n_1$(S,α_1,β_1). The guards of n_1 over the link $S \rightarrow n_1$ are responsible for verifying that n_1 fulfills this requirement. n_2 does not accept packets from n_1 if the wakeup signal of n_1 does not include all the necessary nodes (S,α_1,β_1,α_2, β_2,n_2).

5. If n_1 fails to send the wakeup signal, the guard of n_1 with the lowest ID sends a two-hop broadcast of the wakeup signal. If that guard fails, the guard with the next smallest ID sends the signal, and so on. This design ensures that if there is a chain of colluding malicious nodes then all the nodes will be suspected.
6. The process continues at each step up to the destination.

3.3.3. All-Neighbors On-Demand SLAM (*A-SLAM*)

The high level design goal of A-SLAM is to relax the assumption that every node knows the location of its first-hop and second-hop neighbors, and to simplify the wake-up hardware and wakeup signal. Again considering Figure 1, A-SLAM uses the following steps to wake up the guards along the route from S to D:

1. Node S broadcasts a wake-up signal to all its first-hop neighbors (Z,W,n_1,α_1,β_1). The wake-up signal includes the identity of both the current sender (S), the next-hop (n_1), and the previous-hop (empty for S).
2. Each neighbor of S, after being woken up, decides whether to stay awake or go back to sleep based on the role that it may play on the ongoing communication. If that neighbor is the next-hop (n_1), it stays awake to forward the data and to monitor the next-hop from it (n_2). If that neighbor is a guard (α_1,β_1) for the next-hop n_1 over the link $n_1 \rightarrow n_2$, it stays awake to monitor the behavior of n_1. If the node is a guard of a forwarding node over the previous-hop, it stays awake to detect fabrication by the forwarding node. A node can independently make this determination based on first- and second-hop neighbor information. If none of these cases hold, the node goes back to sleep immediately.
3. Node S sends the data packet to n_1 following the timing schedule presented in Section 3.3.4.
4. Nodes n_1, α_1, and β_1 after being woken up continue to stay awake for T_w. After that, it goes back to sleep.
5. n_1 does the same steps that S did to wake up the next-hop (n_2), n_2's guards (α_2,β_2) and n_1's guards (S,α_1,β_1).
6. If n_1 fails to send the wakeup signal, the guard of n_1 with the lowest ID sends a two-hop broadcast of the wakeup signal through. If that guard fails, the guard with the next smallest ID sends the signal, and so on. This design ensures that if there is a chain of colluding malicious nodes then all the nodes will be suspected.
7. The process continues at each step till the destination.

This scheme results in an increase in the energy consumption compared to G-SLAM due to the needless wake-up of the neighbors that are not guards.

3.3.4. Timing of the wakeup signal

In this section we generate the timing schedules for sending the wake-up signal to nodes using On-Demand SLAM. This is important because the wake-up antennas have a warm-up period that could increase the end-to-end delay of the communication. We design the schedule such that the additional delay due to the sleep-wake protocol is not cumulative with the number of hops for Case II shown below. It is a constant independent of the number of hops. Moreover, the increase in delay with the number of hops is small for Case I. The coefficient of increase per hop is the difference between the time to send a control packet and the time to send a data packet over one hop.

Let $T_{control}$ be the time to send the signal to the wake-up antenna, T_{warmup} be the time for a node to be fully awake from the time it receives the wake-up signal (5 ms for the antenna in [15]), and T_{data} be the time to send a data packet over one hop. This time includes the transmission time and the forwarding time. Thus, within T_{data}, an intermediate node completely receives a data packet. Finally, let T_{wake} be the time a node continues to be awake after being woken up.

Consider an *isolated* flow between S and D, separated by h hops. The intermediate nodes are n_1, n_2, …, n_{h-1}. Let g_i represents the guards of node n_i over the link $n_{i-1} \rightarrow n_i$. Let v_i represents the neighbors of n_i that are not guards of n_{i+1} over the link $n_i \rightarrow n_{i+1}$. Consider the following two disjoint cases based on the relation between ($T_{control}$ + T_{warmup}) and T_{data}.

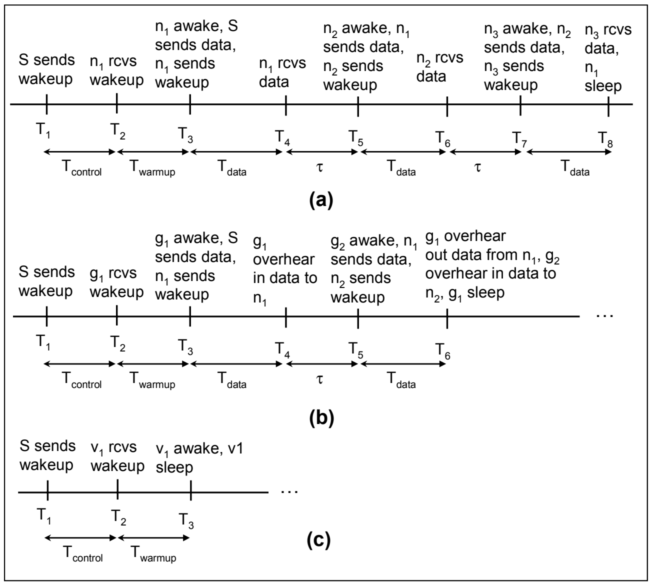
S sends wakeup
n1 rcvs wakeup
n1 awake, S sends data, n1 sends wakeup
n1 rcvs data
n2 awake, n1 sends data, n2 sends wakeup
n2 rcvs data
n3 awake, n2 sends data, n3 sends wakeup
n3 rcvs data, n1 sleep
T1 T2 T3 T4 T5 T6 T7 T8
Tcontrol Twarmup Tdata τ Tdata τ Tdata
(a)
S sends wakeup
g1 rcvs wakeup
g1 awake, S sends data, n1 sends wakeup
g1 overhear in data to n1
g2 awake, n1 sends data, n2 sends wakeup
g1 overhear out data from n1, g2 overhear in data to n2, g1 sleep
T1 T2 T3 T4 T5 T6
Tcontrol Twarmup Tdata τ Tdata
(b)
S sends wakeup
v1 rcvs wakeup
v1 awake, v1 sleep
T1 T2 T3
Tcontrol Twarmup
(c)

Figure 2: Case I wakeup-sleep timing schedule for (a) a node in the data route; (b) a guard node; (c) a neighbor to a node in the data route that is not valid guard (for A-SLAM only)

Case I: ($T_{control}$+T_{warmup}) > T_{data}, τ= ($T_{control}$+T_{warmup}) – T_{data}

Figure 2 above shows the timing schedule for this case. Figure 2(a) shows the timing schedule for a node in the route between the source and the destination. The node, n_1, wakes up at T_3 and goes to sleep at T_8, where T_8-T_3 = T_{data} (to receive data) + τ (wait for the next-hop to be ready to receive the data) + T_{data} (send the data to the next-hop) + {τ + T_{data}} (as a guard for n_2) = $3T_{data}$+2τ. Figure 2(b) shows the timing schedule for a guard node. The guard, g_1, wakes up at T_3 and goes to sleep at T_6, where T_6-T_3 = T_{data} (to overhear incoming data to the node being monitored, n_1) + τ (wait for the next-hop to be ready to receive the data) + T_{data} (to overhear outgoing data from the node being monitored, n_1) = $2T_{data}$ +τ. Figure 2 (c), only meaningful for A-SLAM, shows the schedule for a node that is a neighbor to a node in the route from the source to the destination but is not a guard node. The node, v_1, wakes up at T_3, determines that it can not be a guard, and thus goes back to sleep immediately.

From Figure 2, it can be seen that per hop, the delay incurred is $T_{control} + T_{warmup}$ and at the last hop, the delay due to data (T_{data}) gets exposed (see Equation (1) below).

Case II: ($T_{control}$+T_{warmup})≤T_{data}, τ = T_{data} – ($T_{control}$ + T_{warmup}) Please see Case II, Section 3.4.3 of our technical report [29].

4. Security and Performance Analysis

4.1. Security Analysis

We will prove the following proposition.

Proposition: Due to the sleep-wake mechanism for guards in SLAM, no loss in detection coverage occurs w.r.t. [24].

For this we first prove the following lemma.

Lemma: For any node n_i in the path $S \rightarrow D$ (i =1, …, h-1), the guards for n_{i+1} on the link $n_i \rightarrow n_{i+1}$ are woken up when the communication over the link takes place.

We prove this lemma using mathematical induction.

Let the guards of n_1 over the link $S \rightarrow n_1$ form the set G_1, $n_{h-1} \rightarrow D$ the set G_h, and $n_{i-1} \rightarrow n_i$ the set G_i.

Base case: The source S is honest and therefore it wakes the guard nodes in G_1.

Inductive hypothesis: For n_1, …, n_i ($i \geq 1$), $\forall G_k$ ($k \leq i$) has been woken up at the time when the communication over the link $n_{k-1} \rightarrow n_k$ takes place either directly or indirectly. In the later case n_{k-1} is suspected of malicious action.

To prove: G_{i+1} is woken up at the time of $n_i \rightarrow n_{i+1}$ communication.

Proof: If n_i is honest, it wakes up G_{i+1} (step 4 of G-SLAM/step 5 of A-SLAM).

Else, G_i wakes up G_{i+1} (step 5 of G-SLAM/step 6 of A-SLAM). In the later case n_i is suspected of malicious action (not sending the wakeup signal).

Thus, the lemma is proved by mathematical induction.

The detection of the guards according to rules 1-3 is not changed from baseline local monitoring. Combining the lemma with this observation proves that no loss of detection coverage happens due to SLAM.

4.2. Energy and End-to-End Delay Analysis

For both energy and delay we compare our scheme On-Demand Slam to bring out its worst case behavior. For end-to-end delay, we compare it with local monitoring without sleep-wake (Baseline-LM). For the energy we compare it to a protocol with on-demand sleep-wake for communication and no monitoring (Baseline-OD).

In addition to the notations defined in Section 3.3.4, let $A_{transmit}$ be the current to transmit (at the middle of the transmit range), which is 27mA for Mica2 motes [28]. Let A_{warmup} be the current consumed during the transition from sleep to wakeup (warm up), which is 30mA for Mica2 motes [15]. Finally, let A_{active} be the current in the computationally active mode = the current in the idle listening mode = the current in receive mode (8mA for Mica2 motes).

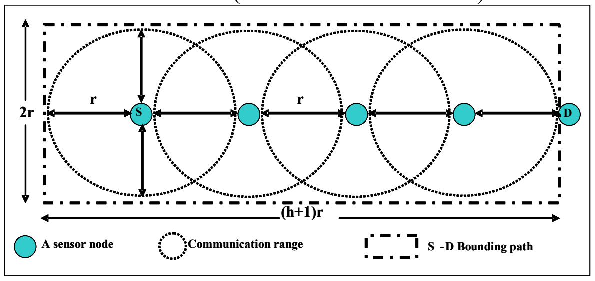

Figure 3: A bounding box over the path S → D

Let us consider a flow between S and D, separated by h hops. The intermediate nodes are n_1, n_2, …, n_{h-1}. The bounding box around S and D covers all possible nodes, including forwarding and guard nodes that may be involved in the communication between S and D. The size of the bounding box is $2r(h+1)r = 2r^2(h+1)$, where r is the transmission range, Figure 3. For On-Demand SLAM, consider the wakeup-sleep scheduling cases of Section 3.3.4.

End-to-end-Delay:

Case I: ($T_{control}$ + T_{warmup}) > T_{data} with τ = ($T_{control}$ + T_{warmup}) – T_{data}

From Figure 2 it can be seen that delay at the first link ($S \rightarrow n_1$) is $T_{control}$ + T_{warmup} + T_{data}. Over each of the succeeding links, the delay is $T_{control} + T_{warmup}$ since the delay due to data (T_{data}) gets exposed. This is due the sleep-wake schedule process that SLAM uses where the wake-up signal is sent at the earliest opportunity. Therefore, the end-to-end delay in SLAM, $\Omega_{SLAM}(h)$, for the link from S to D is

$$\begin{aligned}\Omega_{SLAM}(h) &= T_{contol} + T_{warmup} + T_{data} + (h-1)(T_{control} + T_{warmup}) \\ &= h \cdot (T_{control} + T_{warmup}) + T_{data}\end{aligned} \tag{1}$$

The end-to-end delay in Baseline-LM is

$$\Omega_{Base-LM}(h) = h \cdot T_{data} \tag{2}$$

In this case, the additional end-to-end delay imposed by SLAM depends on the number of hops between S and D

$$\Omega_{SLAM-Add}(h) = \Omega_{SLAM}(h) - \Omega_{Base-LM}(h) = h \cdot \tau + T_{data} \tag{3}$$

Case II: ($T_{control}$ + T_{warmup}) ≤ T_{data} with τ = T_{data} – ($T_{control}$ + T_{warmup}). The end-to-end delay of SLAM is given by

$$\begin{aligned}\Omega_{SLAM}(h) &= T_{contol} + T_{warmup} + T_{data} + (h-1)(T_{data}) \\ &= h \cdot T_{data} + (T_{control} + T_{warmup})\end{aligned} \tag{4}$$

For more details, please refer to Case II, Section 4.2 in our technical report.

In Figure 4, we plot the extra delay of SLAM over Baseline-LM for cases I (Equation(3)) and II (Equation (12) in the technical report [29]) above with T_{data} = 7ms and τ = 1ms. The figure shows that the additional delay due to SLAM increases linearly with the number of hops for Case I while it remains constant for Case II.

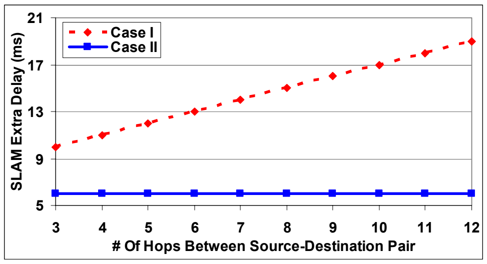

Figure 4: Extra delay due to SLAM over Baseline-LM

Energy overhead:

Case I: $(T_{control} + T_{warmup}) > T_{data}, \tau = (T_{control} + T_{warmup}) - T_{data}$

Baseline-OD: Here only the forwarding nodes are involved in the sleep-wake protocol; all other nodes are asleep. Using Figure 2(a), a forwarding node n_i (i = 1, …, h-1) spends $T_{warmup} = T_3 - T_2$ warming up with current consumption of A_{warmup}, $T_{data} = T_4 - T_3$ receiving data with current consumption of A_{active}, $\tau = T_5 - T_4$ idle waiting for the next-hop to be ready with current consumption of A_{active}, and $T_{data} = T_6 - T_5$ sending data with current consumption of $A_{transmit}$. Therefore, the energy expended by a forwarding node n_i (i = 1, …, h-1) is (we ignore the constant voltage term for the energy comparison since that is the same through all the cases)

$$\varepsilon_{f,base} = T_{warmup} \cdot A_{warmup} + (T_{control} + T_{warmup}) \cdot A_{active} + T_{data} \cdot A_{transmit} \quad (5)$$

Node S spends $T_{control} + T_{warmup} = T_3 - T_1$ idle waiting for n_1 to wake up with A_{active} and $T_{data} = T_4 - T_3$ transmitting data with $A_{transmit}$. Therefore, the energy expended by S is

$$\varepsilon_{S,base} = (T_{control} + T_{warmup}) \cdot A_{active} + T_{data} \cdot A_{transmit} \quad (6)$$

Node D spends T_{warmup} warming up with A_{warmup} and T_{data} receiving data with A_{active}. Thus, the energy expended by D is

$$\varepsilon_{D,base} = T_{warmup} \cdot A_{warmup} + T_{data} \cdot A_{active} \quad (7)$$

On-Demand SLAM: Here the sleep-wake protocol involves, in addition to S and D, the forwarding nodes, the guard nodes, the neighbors of the forwarding nodes that are not guards. We will compute separately for the three kinds of nodes (i) forwarding nodes; (ii) guard nodes that do not act as forwarders; (iii) remaining nodes. The energy of S and D is the same as that in Baseline-OD.

i. Energy expended by a forwarding node n_i (i = 1, …, h-1) $\varepsilon_{f,SLAM} \leq \varepsilon_{f,base} + T_w \cdot A_{active}$. The additional energy is consumed because n_i has to find if n_{i+1} forwards the packet that it was just handed by n_i. The inequality comes in because T_w is the worst-case time in case n_{i+1} is malicious.

ii. Energy expended by a guard node that is not a forwarding node $\varepsilon_{g,SLAM} \leq T_{warmup} \cdot A_{warmup} + T_{data} \cdot A_{active} + T_w \cdot A_{active}$. Consider for example the guard g_1 of n_1 over the link $S \rightarrow n_1$. g_1 has to listen to the communication between S to n_1 and then has to stay listening for a maximum of T_w to see that n_1 forwarded the packet.

iii. Energy expended by a node in the bounding box around S and D that is neither a forwarding node nor a guard node (the "other node", hence the notation "o" in the subscript). For G-SLAM where the wake-up signal is directed to the relevant guard nodes $\varepsilon_{o,G\text{-}SLAM} = 0$. For A-SLAM $\varepsilon_{o,A\text{-}SLAM} = T_{warmup} \cdot A_{warmup}$.

Case II: $(T_{control} + T_{warmup}) \leq T_{data}$ with $\tau = T_{data} - (T_{control} + T_{warmup})$. Please refer to Case II, Section 4.2 in our technical report for the energy consumption of this case.

5. Simulation Results

We use the *ns-2* simulator [27] to simulate a data exchange protocol over a network with local monitoring enabled. We simulate two scenarios individually without A-SLAM (the *baseline*) and with A-SLAM. The baseline is an implementation of a state-of-the-art local monitoring protocol presented in [24]. A-SLAM scenario is built on top of the baseline scenario to provide sleep-wake service for the guards. Nodes are distributed randomly over a square area with a fixed average node density, 100 nodes over 204m×204m. Each node acts as a source and generates data according to a Poisson process with rate μ. The destination is chosen at random and is changed using an exponential distribution with rate λ. A route is evicted if unused for $TOut_{Route}$ time. The experimental parameters are in Table 1. The results are averages over 30 runs. The malicious nodes are randomly chosen so that they are more than 2 hops away.

Table 1: Default simulation parameters

Parameter	Value	Parameter	Value
Avg. number neighbors (N_B)	8	Destination change rate (λ)	0.02/sec
Tx Range (r)	30 m	# malicious nodes (M)	4
Fraction of data monitored (f_{dat})	0.6	Packet generation rate ($1/\mu$)	0.1/sec
Channel BW	40 kbps	Warm up time (T_{warmup})	5ms
Simulation time	1500 s	Watch time (T_w)	30ms
$TOut_{Route}$	50 sec	Number of nodes (N)	100

Adversary model: We are simulating a selective forwarding attack launched by a group of malicious nodes that collude and establish wormholes in the network [25]. During the wormhole attack, a malicious node captures packets from one location in the network, and "tunnels" them to another malicious node at a distant point, which replays them locally. This makes the tunneled packet arrive either sooner or with a lesser number of hops compared to the packets transmitted over normal multihop routes. This creates the illusion that the two end points of the tunnel are very close to each other. The two malicious end points of the tunnel may use it to pass routing traffic to attract routes through them and then launch a variety of attacks against the data traffic flowing on the wormhole, such as selectively dropping the data packets. Unless otherwise mentioned, each node selectively drops a packet passing through it with uniform probability of 0.6

Variable Input metrics: (i) *Fraction of data monitored* (f_{dat})–each guard node randomly monitors a given fraction of the data packets. At other times, it can be asleep from the point of view of a guard's responsibility. (ii) *Data traffic load (μ)*. (iii) *Number of malicious nodes (M)*.

Output metrics: (i) *Delivery ratio*–the ratio of the number of packets delivered to the destination to the number of packets sent out by a node averaged over all the nodes in the network. (ii) *% wakeup time*–the time a node has to wake up specifically to do monitoring averaged over all the nodes as a percentage of the simulation time; (iii) *Average end-to-end delay*–the time it takes a data packet to reach the final destination averaged over all successfully received data packets; (iv) *% True isolation*–the percentage of the total number of malicious nodes that is isolated; (v) *% False isolation*–the percentage of the total number of nodes that is isolated due to natural collisions on the wireless channel; (vi) *Isolation latency*–the time between when the node performs its first malicious action to the time by which *all* its neighbors have isolated it, averaged over all isolated malicious nodes.

Note that our goal is not to show the variation of the output metrics with the input parameters for local monitoring, since that has been covered in [23][24]. Our goal is to study the relative effect on local monitoring with A-SLAM and without.

5.1. Effect of fraction of data monitored

The amount of data traffic is typically several orders of magnitude larger than the amount of control traffic. It may not be reasonable for a guard to monitor all the data traffic in its monitored links. Therefore, a reasonable optimization is to monitor only a fraction of the data traffic. In this set of experiments, we investigate the effect of this optimization.

Figure 5 shows the variations of delivery ratio, % true isolation, and end-to-end delay as we vary f_{dat}. Figure 5(a) shows that the delivery ratio is almost stable above 90% irrespective of the value of f_{dat}. This desirable effect is achieved by proper selection of the *MalC* increment for each value of f_{dat}. The *MalC* increment is designed with an inverse relation to the f_{dat} Figure 5(b) shows that the % of true isolation is almost stable as we vary f_{dat} due to the same reasoning as for Figure 5(a). Importantly, the delivery ratio and the % true isolation in A-SLAM are close to the baseline for all values of f_{dat}.

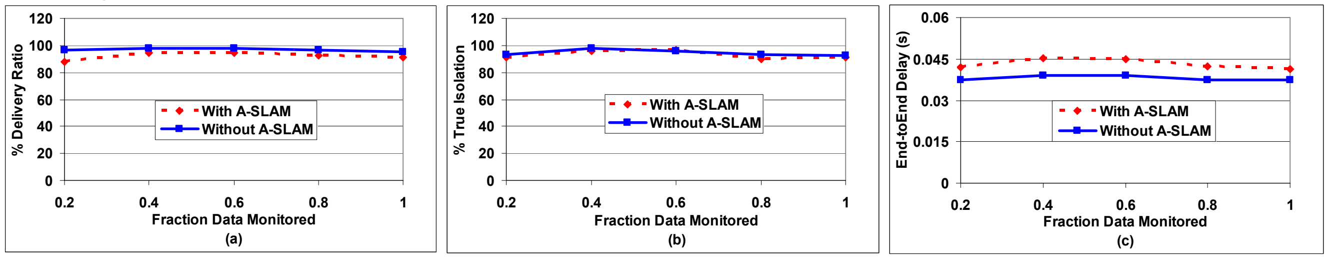

Figure 5: Effect of fraction of data monitored on (a) delivery ratio, (b) % true isolation, and (c) end-to-end delay

However, in Figure 5, the results of A-SLAM are slightly worse than those of the baseline. This is because some of the data packets are additionally dropped in A-SLAM by forwarding, destination, or guard nodes that happen to be asleep when the data packet arrives. This erroneous extra sleep may occur due to collision in the sleep-wake control channel which prevents the respective nodes from waking up. Although the control channel is a separate channel contention still occurs, where a guard of two consecutive links are sent separate wake-up signals concurrently. Figure 5(c) shows that the end-to-end delay is slightly higher for A-SLAM due to the additional warm up time required when the source sends a packet to the first hop.

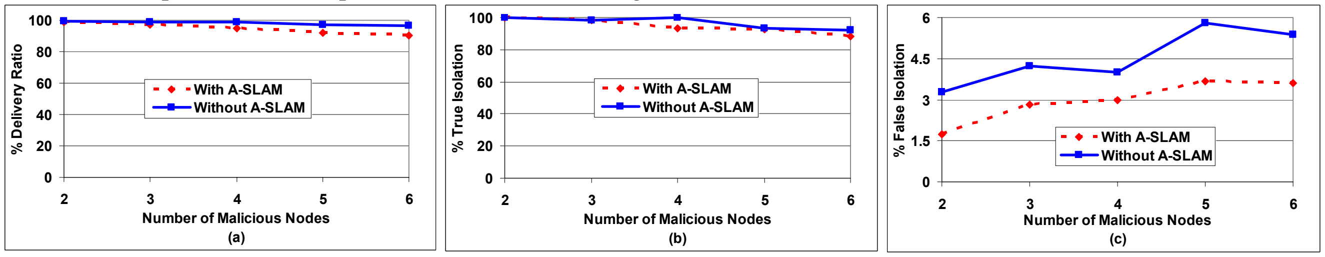

Figure 6: Effect of number of malicious nodes on (a) % delivery ratio, (b) % true isolation, and (c) % false isolation

5.2. Effect of number of malicious nodes

Figure 6 above shows the variations of % delivery ratio, % true isolation, and % false isolation as we vary the number of malicious nodes (*M*). Figure 6(a) shows that the % delivery ratio slightly decreases as *M* increases. This is due to the packets dropped before the malicious nodes are detected and isolated. As the number of malicious nodes increases, this initial drop increases and thus the delivery ratio decreases. Figure 6(b) shows that the % true isolation also slightly decreases as we increase *M*. This is because the number of available guards in the network decreases as more and more nodes get compromised. These two metrics in A-SLAM are slightly lower than those of the base line due to the erroneous extra sleep described in Section 5.1. Figure 6(c) shows that the % false isolation increases as we increase *M*. This is because not all guard nodes come to the decision to isolate a malicious node at the same time. Thus, a guard node may suspect another guard node when the latter isolates a malicious node but the former still has not. The occurrence of this situation increases with M and hence the % of false isolation increases with M. For example, a guard node G_1 detects a malicious node Z earlier than the other guard nodes for the link to Z. G_1 subsequently drops all the traffic forwarded to Z and is therefore suspected by other guard nodes for Z. This problem can be solved by having an authenticated one-hop broadcast whenever a guard node performs a local detection. The % false isolation in A-SLAM is lower than that of the baseline. Again, this is due to some of the packets that may falsely identify a node as malicious may get lost in A-SLAM due to the erroneous extra sleep.

5.3. Wakeup time variations

In this section, we study the effect of varying the fraction of data monitored (f_{dat}), the number of malicious nodes (M), and the data traffic load ($1/\mu$) on the percentage of time that a node needs to stay awake for monitoring using A-SLAM.

Figure 7(a) shows that the percentage of wakeup time required for monitoring increases as the fraction of monitored data increases due to the increase in the number of data packets that a node needs to overhear in its neighborhood. Figure 7(b) shows that the percentage of wakeup time decreases as we increase the number of malicious nodes. As the number of malicious nodes increases, the number of data packets in the system decreases since the malicious nodes are isolated and disallowed from generating data packets. Therefore, the number of packets that need to be monitored decreases, which results in a decrease in the average percentage of wakeup monitor time. Figure 7(c) shows that the average % of monitoring wakeup time increases as the data traffic load increases due the increase in data that needs to be monitored.

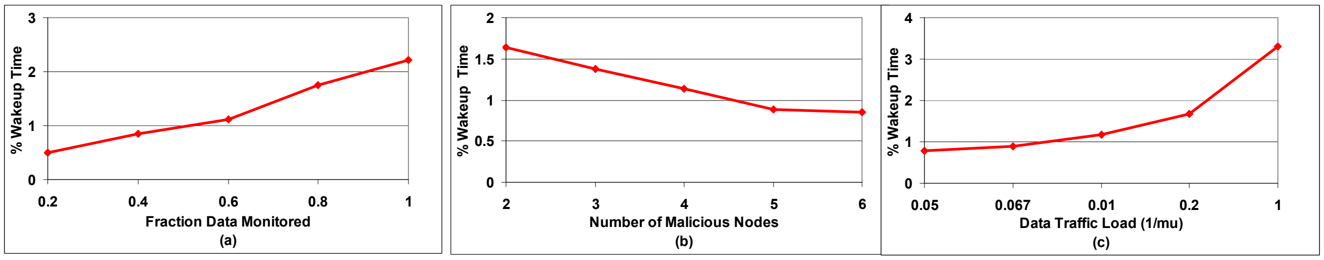

Figure 7: Variations on the percentage of monitoring wakeup time as we vary (a) the fraction of data monitored (f_{dat}); (b) number of malicious nodes (M); and (c) data traffic load (μ)

Overall, from Figure 7(c), compared to the no sleeping case, A-SLAM saves 30-129 times in listening energy for different amounts of data traffic load ($1/\mu$).

5.4. Effect of distance on delay

In this section, we evaluate the variations of the end-to-end delay with the number of hops between the source and destination pairs.

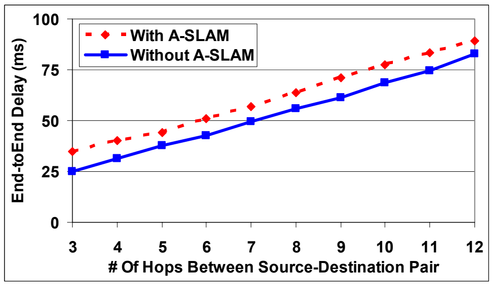

Figure 8: Variation of the end-to-end delay with the hop count

Figure 8 above shows that the end-to-end delay in A-SLAM is always higher than that of the baseline due to the warm-up time needed to wake up the nodes before sending the data. However, due the scheduling strategy in A-SLAM in which each node sends a wake-up signal at the earliest possible opportunity (Section 3.3.4), the warm-up time is only in the critical path at the first hop and therefore, the difference in delay between A-SLAM and the baseline case is not cumulative with the number of hops. The trend of the additional delay due to SLAM follows the trend obtained analytically in Section 4.2 for the case when ($T_{control}$ + T_{warmup}) < T_{data} which is true in these simulation settings.

6. Conclusion

In this paper, we have presented a protocol called SLAM to make local monitoring in sensor networks energy-efficient while maintaining the detection coverage. We classify the domain of sleep-wake protocols into three classes and SLAM correspondingly has three manifestations depending on which baseline sleeping protocol (BSP) is used in the network. For the first class (synchronized sleep-wake), local monitoring needs no modification. For the second class (connectivity or coverage preserving sleep-wake), local monitoring can call the BSP with changed parameter values. For the third class (on-demand sleep-wake), adapting local monitoring is the most challenging and requires hardware support as low-power or passive wake-up antennas. We propose a scheme whereby before communicating on a link, a node awakens the guard nodes responsible for local monitoring on its next hop. We design the scheme to work with adversarial node behavior. We prove analytically that On-Demand SLAM does not weaken the security property of local monitoring. Simulation experiments bring out that over a wide range of conditions, the performance of local monitoring with SLAM is comparable to that without SLAM, while listening energy savings of 30-129 times is realized, depending on the network load. Our ongoing work is looking at providing security guarantees in mobile ad hoc networks and building trust framework for such networks.

Acknowledgements

This work was supported in part by the National Science Foundation under grant no. ECS-0330016 and the Indiana 21st Century Fund under grant no. 512040817. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

7. References

[1] W. Zhang and G. Cao, “DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks,” IEEE Trans. on Wireless Communication 3(5), pp.1689-1701, 2004.

[2] S. Pattem, S. Poduri, and B. Krishnamachari, “Energy-quality tradeoffs for target tracking in wireless sensor networks,” in the second workshop on Information Processing for Sensor Networks (IPSN), pp. 32-46, 2003.

[3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, “Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks,” in Wireless Networks, vol. 3 (5), pp. 48-494, 2002.

[4] S. Bhattacharya, G. Xing, C. Lu, G.-C. Roman, O. Chipara, and B. Harris, “Dynamic wake-up and topology maintenance protocols with spatiotemporal guarantees,” in the fourth workshop on Information Processing for Sensor Networks (IPSN), pp. 28-34, 2005.

[5] S. Kumar, T. H. Lai, and J. Balogh, “On k-coverage in a mostly sleeping sensor network” in the ACM Intl. Conference on Mobile Computing and Networking (MOBICOM), pp. 144-158, 2004.

[6] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Microsensor Networks,” in the 33rd Hawaii Intl. Conference on System Sciences (HICSS), pp. 3005-3014, 2004.

[7] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, “Integrated coverage and connectivity configuration for energy conservation in sensor networks,” in ACM Trans. on Sensor Networks (TOSN), Vol. 1 , Issue 1, pp. 36-72, 2005.

[8] W. Ye, J. Heidemann, and D. Estrin, "An energy efficient MAC protocol for wireless sensor Networks," in the IEEE Conference on Computer Communications (INFOCOM), pp. 1567- 1576, 2002.

[9] R. Naik, S. Biswas, and S. Datta, “Distributed Sleep-Scheduling Protocols for Energy Conservation in Wireless Networks,” in the 38th HICSS, pp. 285b - 285b, 2005.

[10] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, “Wireless sensor networks for habitat monitoring,” in the ACM Intl. Workshop on Wireless Sensor Networks and Applications, pp. 88-97, 2002.

[11] C. Guo, L. C. Zhong, and J. M. Rabaey, “Low power distributed MAC for ad hoc sensor radio networks,” in IEEE Global Telecommunications Conference (GLOBECOM '01), pp. 2944–2948, vol.5, 2001.

[12] J. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, and T. Tuan, “Picoradios for wireless sensor networks: The next challenge in ultra-low-power design,” in the Intl. Solid-State Circuits Conference, pp. 200-201, 2002.

[13] J. Silva., J. Shamberger, M. J. Ammer, C. Guo, S. Li, R. Shah, T. Tuan, M. Sheets, J. M. Rabaey, B. Nikolic, A. S.-Vincentelli, and P. Wright, “Design methodology for picoradio networks,” in Design Automation and Test in Europe (DATE), pp. 314-323, 2001.

[14] http://www.austriamicrosystems.com/03products/data/AS3931Product_brief_0204.pdf.

[15] L. Gu and J.A Stankovic, “Radio-Triggered Wake-Up Capability for Sensor Networks,” in Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 27-36, 2004.

[16] Y. Huang and W. Lee, “A cooperative intrusion detection system for ad hoc networks,” in the 1st ACM workshop on Security of ad hoc and sensor networks, pp. 135-147, 2003.

[17] A. Silva, M. Martins, B. Rocha, A. Loureiro, L. Ruiz, and H. Wong, “Decentralized intrusion detection in wireless sensor networks,” in the ACM Intl. workshop on Quality of service & security in wireless and mobile networks, pp. 16-23, 2005.

[18] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” in MOBICOM, pp. 255-265, 2000.

[19] S.J. Lee and M. Gerla, “Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks,” in IEEE Intl. Conference on Communications (ICC), pp. 3201-3205, 2001.

[20] A. A. Pirzada and C. McDonald, “Establishing Trust In Pure Ad-hoc Networks,” in the proceedings of 27th Australasian Computer Science Conference (ACSC'04), pp. 47-54, 2004.

[21] S. Buchegger, J.-Y. Le Boudec, “Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes - Fairness In Distributed Ad-hoc NeTworks,” in the ACM Intl. Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), pp. 80-91, 2002.

[22] I. Khalil, S. Bagchi, and N. B. Shroff, “Analysis and Evaluation of SECOS, a protocol for Energy Efficient and Secure Communication in Sensor Networks,” in Elsevier Ad Hoc Networks Journal, vol. 5(3), pp. 360-391, April 2007.

[23] I. Khalil, S. Bagchi, and C. Nina-Rotaru, “DICAS: Detection, Diagnosis and Isolation of Control Attacks in Sensor Networks,” in the IEEE/CreateNet Intl. Conference on Security and Privacy in Communication Networks (SecureComm), pp. 89-100, 2005.

[24] I. Khalil, S. Bagchi, and N. Shroff, “LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks,” in the Intl. Conference on Dependable Systems and Networks (DSN ’05), pp. 612-621, 2005.

[25] C. Karlof and D. Wagner, “Secure Routing in Sensor Networks: Attacks and Countermeasures,” in the 1st IEEE Intl. Workshop on Sensor Network Protocols and Applications, pp. 113-127, 2003.

[26] D. Liu and P. Ning, “Establishing Pair-wise Keys in Distributed Sensor Networks,” in the ACM Conf. of Computer and Communication Security, pp. 52-61, 2003.

[27] “The Network Simulator ns-2,” At: www.isi.edu/nsnam/ns/

[28] http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf...

[29] I. Khalil, S. Bagchi, and N. B. Shroff, “SLAM: Sleep-Wake Aware Local Monitoring for Sensor Networks,” TR ECE 06-14, Purdue University, November 2006.

[30] Chipcon CC1000 Datasheet, Chipcon Inc. http://www.chipcon.com/files/CC1000DataSheet21.pdf.