ActSee: Activity-Aware Radio Duty Cycling for Sensor Networks in Smart Environments

Shao-Jie Tang* Debraj De† Wen-Zhan Song† Diane Cook‡ Sajal Das♭
*stang7@iit.edu, †dde1@student.gsu.edu, †wsong@gsu.edu, ‡djcook@wsu.edu, ♭das@cse.uta.edu
*Department of Computer Science, Illinois Institute of Technology
†Sensorweb Research Laboratory, Department of Computer Science, Georgia State University
‡School of Electrical Engineering and Computer Science, Washington State University
♭Department of Computer Science and Engineering, University of Texas at Arlington

***Abstract*—In this paper we present *ActSee*, an activity-aware radio duty cycling protocol that utilizes the learned event activity pattern information to intelligently adjust radio duty cycles in wireless sensor networks. The goal is to minimize data delivery latency and maximize throughput, while still conserving energy in the network. Based on the collected intelligence about activity patterns in the form of an Activity Transition Probability Graph ($ATPG$), the $ActSee$ protocol solves a linear programming problem to select the duty cycles for a set of sensor nodes. The non-uniform and dynamic duty cycle assignments are distributed by the active node(s) in a defined neighborhood. As a case study, we have evaluated *ActSee* using the activity patterns from a real Smart Home testbed CASAS [1]. Experimental results from both real sensor network testbed and the sensor network simulator TOSSIM validate that our proposed activity-aware radio duty cycling protocol outperforms traditional MAC protocols, uniform duty cycling and reactive duty cycling strategies in terms of data delivery latency, throughput and network lifetime.**

***Index Terms*—Activity-awareness, radio duty cycle, data delivery latency, energy consumption, linear programming, smart home environment.**

I. Introduction

Significant advancements in wireless communications, microelectronic technologies and distributed protocols have revealed great potential of wireless sensor network systems for a wide variety of pervasive computing applications. Unlike traditional networks (which are mainly focused on end-to-end data transfer), sensor networks are deeply embedded in the physical environments and hence their computation and data communications are triggered by environment situations or activities. In many applications such as smart environments, the event activity information shows certain learnable patterns in the long run (as shown in Figure 1). However, most of the sensor network designs till date do not fully exploit the activity patterns in the environment for performance improvement of the considered applications or the deployed network. Here, by activity we mean events sensed and reported by the sensor nodes. For example, in a smart home environment, the detected motion events of the residents constitute activities.

These sensed activities are collected by the network with an application goal. But *there remains a missing link in the sensor network design: how to utilize the feedback from the sensed and analyzed activity patterns for efficient network operations and resource usage?* The activity pattern information, if utilized in an intelligent manner, can improve the sensor network performance while reducing resource usages. For instance, Figure 1 depicts daylong average activity (motion detection) rates of two motion sensor nodes (in different rooms) in the real-time *CASAS* Smart Home testbed [1]. The triggering rates for these motion sensors, one in the bedroom and another in the dining room, are computed based on the 24 hour data collected for 57 days of operation. It can be observed that there are distinct activity patterns that vary in time (say, throughout a day) and space (say, bedroom vs. living room). This means at different time, the activity patterns in the same

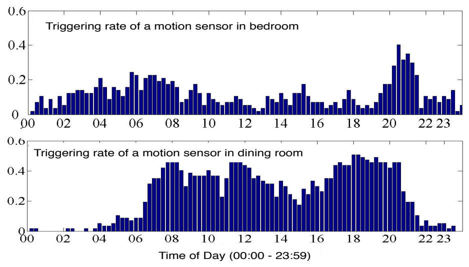


Fig. 1. Change of activity patterns in a smart home in 24 hours.

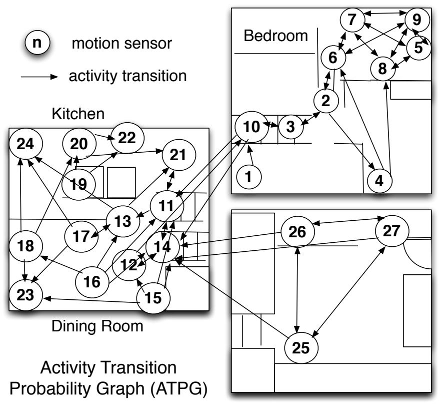


Fig. 2. Activity Transition Probability Graph ($ATPG$) learnt from the *CASAS* Smart Home testbed [1]. The significant transition probability from node 27 to nodes 14, 25 and 26 are 12%, 45% and 40% respectively.

space are different, and at the same time the activity patterns at different space are also different. These patterns can be learnt and utilized for distributed and autonomous control of communication resource (say, wireless bandwidth or battery power) usage in the deployed sensor networks. In this scenario, our activity (i.e., context) aware design attempts to adapt the network resources using intelligence gathered from the detected activity patterns.

To this end, we have designed an efficient activity-aware wireless (radio) duty cycling protocol, called *ActSee*. The novelty of this protocol is that it learns and adapts from the sensed activities in the network, with a goal to decide on the policy of updating the sensor node's duty cycle. This is distinct from most of the existing duty cycling methods which are based on the generated data traffic in the network and state of communication medium. In the *ActSee* protocol, on the other hand, an intelligent agent is trained to extract patterns from the periodic event activities. The agent generates what is called an Activity Transition Probability Graph (ATPG), as shown in Figure 2, which contains information about the transition probability of daily activities in a smart environment. The figure shows the ATPG with 28 nodes, learnt by an agent from the *CASAS* Smart Home testbed [1]. Using this intelligence about predicted activities, the radio duty cycle of the nodes are dynamically configured for optimizing resource usage and performance, through sleep/wakeup scheduling of the radio. Note here that the goal of $ActSee$ is not to propose a new MAC protocol for sensor networks, instead it proposes the design of an activity-aware duty cycle adjustment algorithm that can run on top of any existing duty cycling MAC protocol.

Additionally, our proposed *ActSee* protocol reduces event delivery latency (to the base station or sink node), while minimizing the network energy consumption, by utilizing the available activity context information. In the system model, there is a transition in event activity at the end of each time period. During the current time period, *ActSee* pre-computes the duty cycle of nodes for the next period (of activity transition) based on the current activity state information. Note that an *activity state* means the conditions or the status of activity detected by the sensor nodes (those are actively sensing events) in the network. This updated duty cycle assignment is then propagated to the corresponding sensor nodes during the current period or stage. Thus each sensor node can receive its "future" duty cycle assignment in the current stage or period. The problem of selecting proper duty cycle assignment for the next stage for each sensor node, is formulated as a Constrained Markov Decision Process (CMDP) [11]. After solving the CMDP, *ActSee* applies optimal duty cycle assignments to the network.

The rest of the paper is organized as follows. In section II, we discuss the related work and our motivation behind this study. Section III presents the preliminaries such as the system and latency models, followed by the details on the proposed *ActSee* protocol. Section IV deals with experimental results and performance evaluation of *ActSee*. Finally we conclude the paper in section V.

II. Related Work and Background

Sensor networks have been adopted in smart environments (such as CASAS [1], MavHome [2]) to extract activity patterns and detecting different events for automating and facilitating application services (e.g, assisted living). But, in all these, the detected activity patterns are not utilized in the network for optimizing network operation and resource usage.

In the existing literature, the following radio duty cycling strategies are used: (a) The first strategy is *uniform duty cycling* that lets each node operate at the same radio duty cycle. However, this simple strategy does not leverage the underlying activity transition pattern. (b) The second strategy is *reactive duty cycling*. After a node successfully detects the presence of an activity, it starts to increase it's own duty cycle and informs those sensor nodes on the routing path as well. Obviously, this strategy outperforms the uniform duty cycling in terms of lower data delivery latency and higher energy efficiency. However, the main shortcoming of this "reactive" strategy is due to the delay involved in delivery of data packets indicating detected events ("decision propagation phase"). In particular, it may take long time to inform all the sensor nodes on the routing path to increase their duty cycle. When the activity transition appears to be frequent, or the network size is large, the decision propagation delay may become a significant bottleneck to the application performance. This is often unacceptable in time sensitive applications such as real-time tracking or monitoring. (c) The third strategy are those existing synchronous or asynchronous MAC protocols that have been investigated extensively. Synchronous MAC protocols specify the wake-up and sleep periods for communication to reduce the energy waste in idle listening. Nodes periodically exchange SYNC packets to synchronize the data transfer. S-MAC [4] and T-MAC [3] are examples of synchronous MAC protocols. B-MAC [6] and X-MAC [5] are asynchronous MAC protocols and rely on low power listening (LPL) mechanism. B-MAC utilizes a long preamble to achieve low power communication, X-MAC uses short preambles with target address information to reduce the excessive preamble. When a receiver wakes up and detects a short preamble, it examines the target address that is included in the preamble. If a node is the intended recipient, it starts to receive the incoming data, otherwise it goes to sleep immediately. These MAC protocols do not generally learn and utilize activity patterns to optimize performance. So activity-aware radio duty cycling in sensor networks is a relatively unexplored area. This motivates our work in this paper.

Motivated by the shortcomings of existing strategies, in this paper we have proposed an energy-efficient duty cycling strategy, called $ActSee$ that tries to achieve two goals: low data delivery latency and higher network lifetime. Instead of proposing yet another new MAC protocol, $ActSee$ adapts effective radio sleep interval or duty cycle based on activity patterns, and can run on top of those existing MAC protocols. Besides the novelty of activity-awareness, $ActSee$ is also unique in achieving both of these goals as follows. It maintains

(i) high duty cycle for nodes on the routing paths for active and predicted (predicted to be active in next stage or period, according to the activity pattern) data sources for fast and reliable data delivery; (ii) low duty cycle (for energy saving) for potentially idle nodes which are not predicted to be active next; and (iii) minimum decision propagation delay to the base station. The big challenge for maintaining such non-uniform duty cycling is that, the distribution of duty cycle of the nodes has to dynamically adapt with change of active and predicted data sources. $ActSee$ is inspired from CMDP [11] that has been widely used in optimizing network performance subject to resource constraints. It has been applied to study traffic problems [9], analyze the throughput maximization problem under delay constraints in telecommunication applications [8], and investigate optimal sampling problem [10].

III. Activity-Aware Duty Cycling

A. System Model

The Activity Transition Probability Graph ($ATPG$) for a smart environment can be constructed based on the observed activity patterns of sensed events and their transitions. In $ATPG$, a node represents a sensor node in the environment and an edge denotes a pair of sensor nodes that can physically be reached directly from each other. These node pairs are connected with a weighted edge in the $ATPG$ graph that denotes transition of activity between them. In this way, we can estimate the probability of transition between two sensor nodes, say x and y, based on the relative frequency of events at x followed by event at y. Figure 2 shows the floorplan and layout of sensor node distribution for the $CASAS$ smart home testbed [1]. The weight associated with edge (x, y), denoting the probability of activity transition from x to y, is estimated as:

$$p(x,y) = \frac{number\ of\ events\ for\ x\ followed\ by\ y}{number\ of\ events\ for\ x} \quad (1)$$

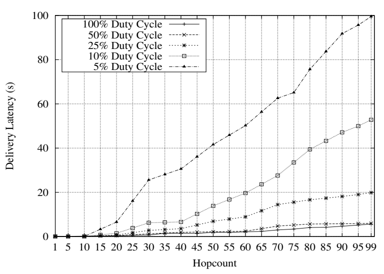
More specifically, we model $ATPG$ as a discrete time N-state Markov chain, $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$, with transition probabilities $p(i, j)$ from state x_i to x_j (for $1 \le i, j \le N$). Each state x_i is associated with a set of active sensor nodes at that state of activity. For simplicity of presentation, let there be only one active sensor node at each state of activity. In practice, there exists a cluster of active nodes at the same activity state, the cluster can be treated as a single node for analysis. Such activity node clusters can be formed by finding the relatively densely connected component in the activity transition graph. Example of such cluster in a smart home can be the set of nodes in the kitchen, bedroom or bathroom etc. So the nodes closely related to the same activity context usually fall into the same cluster. We denote the sensor node(s) associated with state x_i as v_i. For any state x_i, we define the neighbors of x_i as those states $x_j \in \mathbf{X}$ with positive transition probability $p(i, j) > 0$ from x_i.

The term *state report latency* is defined as the time it takes from the moment an activity enters a new state to the moment the sink node is informed of that activity. In a typical sensor network, the *state report latency* usually includes state detection latency and state delivery latency. The *state detection latency* is defined as the time duration from the moment an event activity (e.g., motion) occurred, to the moment a sensor has detected this event activity. In real-systems the *state detection latency* can be negligible as the new-generation sensor node design achieves wake-on capability [7]. Then *state delivery latency* is defined as the time duration from the moment the activity event has been detected by sensor node to the moment the sink node receives the event data successfully. By assuming the *state detection latency* is zero, in this work the focus is to design a activity-aware radio duty cycling protocol that minimizes the expected *state delivery latency*. Ideally, if the radio works with 100% duty cycle, the activity events can be reported with minimum latency.

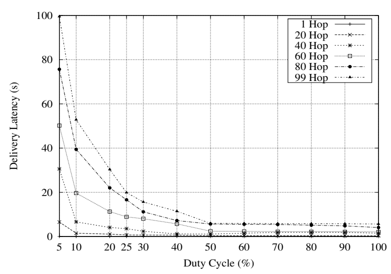
Now in the system model, for each node v, we define a finite candidate set for duty cycle assignment $\mathcal{D}(v) = \{d_1, d_2, \cdots, d_n\}$, where $d_i \in \mathcal{D}(v)$. Specifically, $\mathcal{D}(v)$ defines all possible duty cycle assignments for node v. For simplicity of notation, we assume $\mathcal{D}(v) = \mathcal{D}\ \forall\ v$. Typically, $\mathcal{D} = \{2\%, 5\%, 8\%, 10\%, 15\%, 20\%, 25\%\}$. We have formulated the optimal duty cycle assignment problem as a Markov Decision Process, where decisions are made at points of time, referred to as the decision stages or periods. Thus, time is divided into *stages*: $T = \{t | t = 0, 1, 2, \cdots\}$. At the start of each stage, the decision maker observes the system in a state, say $x_i \in \mathbf{X}$, and then chooses action a_j from the set of allowable actions in that state. In this work, an *action* specifies a duty cycle assignment for each sensor node for the next stage.

B. Validation of Activity State Delivery Latency Model

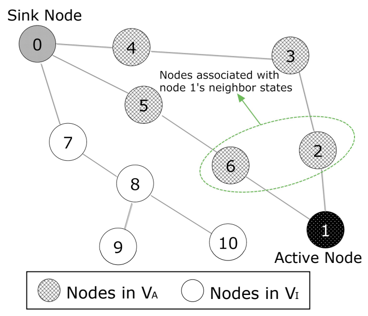
In this subsection, we validate the estimated formulation for state delivery latency in terms of node hop distance (to sink) and node duty cycle. We have performed experiments in a 100 node sensor network using the *TOSSIM* simulator to compute the state delivery latency (f_L) in terms of hopcount (h) and duty cycle (d). The link-layer model in *TOSSIM* is valid for static and dynamic practical scenarios. From the experimental data as in Figure 3(a) and Figure 3(b), it can be observed that, f_L varies almost linearly with h, and is inversely proportional to d. Therefore, if the routing path is known and the duty cycle is determined for the nodes on the routing path, the state delivery latency can be estimated as: $f_L \propto \frac{h}{d}$, so $f_L = c \cdot \frac{h}{d}$, where c is a constant factor depending on the system set up factors such as network topology. This estimation of state delivery latency can also be verified theoretically in standard network model. Assuming that all the nodes are synchronized, the nodes on the selected route maintain the same duty cycle d. Then the message can travel $d \cdot T/T_{trans}$ hops during one radio ON interval, where T is the time period of radio duty cycling and T_{trans} is the transmission time for one packet over a link. After that time, the message has to wait during the radio sleep interval in the intermediate forwarder node. Then it is forwarded again via $d \cdot T/T_{trans}$ hops. Since the nodes are strictly synchronized, their radio duty cycle periods align in time. This forwarding of data through radio ON period is



(a)



(b)



(c)

Fig. 3. (a) State delivery latency (seconds) vs. Hopcount with varying Duty Cycle. (b) State delivery latency (seconds) vs. Duty Cycle (%) with varying hopcount. (c) Illustration of *ActSee*: node 0 is the sink, node 1 is currently active node, node 2 and node 6 are neighbors of nodes 1. The routing path from node 2 to sink is 2→3→4→0, and that one from node 6 to sink is 6→5→0. In this example, V_A =2, 3, 4, 6, 5, and the set V_I contains the remaining nodes. In $ActSee$, node 1 will pre-select a duty cycle assignment for all nodes in V_A and propagate its decision to them during current stage. All the remaining nodes will work with lowest duty cycle.

performed $\frac{h}{d \cdot T/T_{trans}}$ times before arriving at the sink, where h is the hopcount of sink from the source node. Therefore the total estimated state delivery latency $f_L \propto \frac{h}{d \cdot T/T_{trans}}$. Since T_{trans} is constant for fixed message size and T is also fixed, the formulation of f_L validates our estimated model for data delivery latency: $f_L = c \cdot \frac{h}{d}$. It is worth noting that besides hop distance and duty cycle, collisions and congestions among links may also affect the delivery latency. But in the considered network scenario, collision and congestion are usually negligible. In the network of concern, data traffic is usually low or moderate, e.g., 48 byte packet generation in every 5 seconds during activity detection. At any moment there is only a limited number of active nodes generating data at low rate. Also the wireless frequency can be chosen so that it does not overlap with other sources of radio signals (e.g., Wi-Fi, microwave) in a smart environment.

C. Proposed ActSee Protocol

In $ActSee$, we define the term E as the *energy budget* which is the maximum expected average duty cycle allowed. We study the following constrained optimization problem: *considering a long state evolving process, given a device energy consumption budget E and an activity state transition matrix P, what is the optimal radio duty cycle assignment strategy μ, such that the expected activity state delivery latency (say, $E[L]$) is minimized, and the expected average duty cycle (say, $E[d]$) is maintained under the budget such that $E[d] \leq E$?*

In the *ActSee* protocol each node keeps track of its hopcount to the sink, as well as the next hop on the route to the sink. No matter what routing policy is used by the system, it will not affect the results derived in *ActSee* which requires any change in routing path and hopcount to the sink to be informed to the system. For ease of explanation, during any state x_i, we partition all the sensor nodes into two disjoint sets: inactive set V_I and active set V_A, where V_A contains all the sensor nodes that are associated with active node x_i and its neighbor states in ATPG graph, as well as those nodes appearing on the routing paths from them to the sink. (If x_i has a self loop, x_i is also the neighbor state of itself.) Essentially, the active set V_A contains all possible sensor nodes that may become active or help in relaying in the next stage or time period. The rest of the nodes belong to V_I. The main idea behind *ActSee* is to increase the duty cycle of V_A in the next stage, while keeping V_I in low duty cycle.

Now to eliminate the decision propagation delay existing in reactive strategy (discussed in Section II), *ActSee* works as follows. Based on the available nodes in the network and the collected information about activity patterns in form of $ATPG$ graph, a back-end system (connected to sink) runs a linear program routine to select the action set (duty cycle assignments for the nodes in V_A for each possible active node). Note that for continuous events like motion, all the neighbor nodes in ATPG are also the neighbors in communication topology (but not necessarily vice-versa). Now for a deployed network, the back-end system calculates only once, the action set of each possible active node. This action set information is disseminated once, stored in the nodes, and updated when necessary. Once a node is active, it reconfigures the duty cycles of its neighbors and also far-off nodes. It is worth noting that $ActSee$ *exploits the existing beacon message* in a routing protocol to piggyback the duty cycle assignment information. Thus it saves energy for distributed duty cycle assignment task. Until some node is dead or some new node is added, the back-end system does not need to recompute or disseminate the action set. Otherwise it recomputes the action set based on periodically collected regular node status information. The activity pattern typically repeats in smart environments after a reasonable learning period. Thus the linear program routine to select the action set is not required to be run often. After the learning phase, *ActSee* conserves energy in a long period, since it is computed based on the knowledge of activity patterns in $ATPG$.

At the beginning of each stage, the currently active sensor nodes v_i use their duty cycle set for assigning duty cycles to nodes in V_A for the *next* stage. The detailed selection of a proper duty cycle assignment is explained in the next

Algorithm 1 Pseudo code for *ActSee* in each node v

Input: Optimal duty cycling strategy μ (computed from Algorithm 2) and current state information

Output: The duty cycle assignment during next stage

```
1:  if $v$ does not receive any updated duty cycle information then
2:      Keep itself in lowest duty cycle in the next state;
3:  end if
4:  if $v = v_i$ then
5:      Randomly choose a duty cycle assignment (based on duty cycling strategy $\mu$ calculated from Algorithm 2) for all sensor nodes in $V_A$ for the next stage;
6:      Propagate the decision to the rest of the nodes in $V_A$ immediately;
7:  end if
8:  if $v \neq v_i$ and it receives the decision from $v_i$ then
9:      Adjust duty cycle correspondingly in the next stage;
10: end if
```

subsection based on the strategy μ for duty cycle selection. Then the decision is propagated to V_A immediately during the current stage. By executing the decision propagation phase during current stage, $ActSee$ is able to reduce the decision propagation delay. For the remaining sensor nodes which do not receive any updated duty cycle information, they operate at the lowest duty cycle during the next stage to save energy.

Let us illustrate with the example in Figure. 3(c) in which node 1 is the currently active node, while nodes 2 and 6 are neighbors of node 1 in the $ATPG$. More precisely, the states of both 2 and 6 are neighbors of 1's state in the $ATPG$. In this example, the routing path from node 2 is 2→3→4→0, and that from node 6 is 6→5→0. Also $V_A = \{2, 3, 4, 6, 5\}$ and $V_I = \{7, 8, 9, 10\}$. In *ActSee*, node 1 will pre-select a duty cycle assignment for all nodes in V_A and will propagate its decision to them during the current stage, while all the remaining nodes will work at lowest duty cycle. Furthermore, an active sensor node can make a decision immediately based on local information, given it knows its action set. Algorithm 1 provides the pseudo code for node. Here, μ specifies the duty cycle assignment of the nodes at each state.

We model the computation of optimal duty cycling strategy problem as a Constraint Markov Decision Process (CMDP). By solving the corresponding Linear Program (LP) in polynomial time [11], we obtain an optimal strategy μ for each state. Additionally, the selection of different duty cycle for each state is randomized under fixed distribution.

1) Constraint Markov Decision Process: Markov decision processes (MDP), also known as controlled Markov chains, constitute a basic framework for dynamically controlling systems that evolve in a stochastic way. In a standard MDP, the current action may also affect the transition probability for the next time period, but this is not the case in this paper since the transition graph does not depend on the current duty cycle algorithm. MDP is a generalization of (non-controlled) Markov chains, and many useful properties of Markov chains carry over to controlled Markov chains. The model and problem that we consider in this paper is especially challenging in the sense that more than one objective cost exist, and the controller minimizes one of the objectives subject to constraint on the other.

To apply the above to our problem scenario, we define a tuple $\{O, \mathbf{X}, \mathcal{A}, \mathcal{P}, L, D\}$, where $O = \{t | t = 1, 2, \cdots\}$ denotes the set of decision epochs (note that decisions are made at the beginning of each stage), and $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$ is a countable state space. Although we limit our study to discrete activity state transitions, the continuous case can also be handled by dividing it into discrete space. $\mathcal{A}$ is a metric set of actions. We denote $\mathcal{A}(x_i) = \{a_1^i, a_2^i, \cdots\}$ as the action set allowable at state x_i. Each action $a_j^i \in \mathcal{A}(x_i)$ defines a duty cycle assignment for a sensor node in the *next* stage. Let $d(a_j^i, v)$ denote the duty cycle assignment for sensor node v under action a_j^i. Theoretically, each sensor node has $|\mathcal{D}|$ possible duty cycle assignments, thus the action space could be as large as $N^{|\mathcal{D}|}$. In order to reduce the search space, we again leverage the underlying transition graph to facilitate our study. Specifically we restrict the action set $\mathcal{A}(x_i)$ as follows: (i) Only the nodes in V_A, the possible active nodes in the next state, will be considered to increase the duty cycle. The rest of the nodes operate with lowest duty cycle by default. (ii) All relay nodes that appear on the routing path from the same source node, have the same duty cycle as that of their source. (iii) For those nodes appearing on the crossover point of multiple routing paths, $ActSee$ sets their duty cycle to the maximum one among all crossing paths.

We use the previous example in Figure 3(c) for illustration. In this example, a possible action a_j^1 selected by node 1 could be $d(a_j^1, 2) = d(a_j^1, 3) = d(a_j^1, 4) = 10\%$ and $d(a_j^1, 6) = d(a_j^1, 5) = 15\%$. Clearly, the size of searching space is $2^{|\mathcal{D}|}$ where 2 is the number of neighbor states of the current state in this example. Now let $\rho(x_i, a_j^i)$ denote the "occupation measure" of state x_i and action a_j^i. It denotes the probability that such state-action pair ever exists in the decision process. $E_v(d)$ denotes the expected average duty cycle of sensor node v, which is expressed as in Eq. (2). Notice that the "occupation measure" $\rho()$ is decided by corresponding duty cycling strategy. The term $\mathcal{P}$ are the transition probabilities. We define $\mathcal{P}_{iaj}$ as the probability of moving from system state i to j, when action a is taken. Since different duty cycling strategies will not affect the actual transition process of the event activities, given the activity state transition probability matrix P, it is easy to conclude that $\mathcal{P}_{iaj} = P_{ij}$. Let L be the immediate cost. In this paper, we define $L(x_i, a_j^i)$ as the expected average delivery latency during the next stage by taking action a_j^i, where $L(x_i, a_j^i)$ is expressed as in Eq. (2). Recall that in terms of hop count and duty cycle, $f_L(v_k \rightsquigarrow v_0, a_j^i)$ denotes the average delivery latency through a fixed routing path $v_k \rightsquigarrow v_0$ under action a_j^i, and $\mathcal{N}(x_i)$ denotes the neighbor set of state x_i. Then the expected average delivery latency $E[L]$ can be computed as in Eq. (2). E is the maximum allowed expected average duty cycle budget. Therefore for each node v, we have $E_v[d] \leq E$.

$$\begin{aligned} E_v[d] &= \sum_{x_i \in X} \sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) \cdot d(a_j^i, v) \\ L(x_i, a_j^i) &= \sum_{x_k \in \mathcal{N}(x_i)} P_{ik} \cdot f_L(v_k \rightsquigarrow v_0, a_j^i) \\ E[L] &= \sum_{x_i \in X} \sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) \cdot L(x_i, a_j^i) \end{aligned} \tag{2}$$

2) Optimal Duty Cycling Policy μ : In order to compute the optimal strategy of the CMDP with expected average cost criteria, we formulate it as a linear programming (LP) problem. After solving the corresponding linear program, we obtain the optimal strategy through normalization [11]. The following presents how to formulate the duty cycling optimization problem as a linear program. The constraints (1) and (3) ensure that $\rho(x_i, a_j^i)$ is a feasible probability measure. The energy budget can be restricted under the constraint (2) by setting the expected average duty cycle less than E. In inequality (4), $\delta_{x_j}(x_i)$ is the delta function of x_i concentrated on the state x_j.

$$\delta_{x_j}(x_i) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

The constraint (4) describes that the outgoing rate and incoming rate for a state must be the same. At the same time, it emphasizes the property for ergodic processes. After solving the linear program, we get an optimal occupation measure $\rho()$ in terms of delivery latency minimization for each state/action pair. However, since $\sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) \leq 1$, we can not directly use $\rho(x_i, a_j^i)$ as the probability of taking action a_j^i at state x_i. Instead, the stationary optimal duty cycling strategy μ can be determined from $\rho(x_i, a_j^i)$ as follows:

$$\mu(a_j^i | x_i) = \frac{\rho(x_i, a_j^i)}{\sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i)}$$

Here $\mu(a_j^i | x_i)$ describes the probability of taking action a_j^i at state x_i. It is easy to verify that $\sum_{a_j^i \in A(x_i)} \mu(a_j^i | x_i) = 1$. For any number of input states, Algorithm 2 can return an optimal strategy μ in polynomial time. As the input to Algorithm 1, $\mu(a_j^i | x_i)$ for all j will be propagated to each corresponding sensor node v_i.

Problem: *LP-Minimizing Expected Delivery Latency*
Objective: *Minimize* $E[L]$
subject to:

$$\begin{cases} (1)\ \ \rho(x_i, a_j^i) \geq 0,\ \ \forall x_i,\ \forall a_j^i \\ (2)\ \ E_v[d] \leq E,\ \ \forall v \\ (3)\ \ \sum_{x_i \in X} \sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) = 1 \\ (4)\ \ \forall x_j \in X \\ \quad \sum_{x_i \in X} \sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i)(\delta_{x_j}(x_i) - P_{ij}) = 0 \end{cases}$$

Algorithm 2 Computation of Optimal Duty Cycling Strategy μ

Input: Energy budget E, transition matrix P, underlying wireless sensor network topology G
Output: Optimal duty cycling strategy μ.

```
1: Solve corresponding CMDP linear programming to get the
   occupation measure $\rho(x_i, a_j^i)$, $\forall x_i \in X, \forall a_j^i \in A(x_i)$;
2: Calculate optimal duty cycling strategy $\mu$ from $\rho(x_i, a_j^i)$
   as:
```

$$\mu(a_j^i | x_i) = \frac{\rho(x_i, a_j^i)}{\sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i)}$$

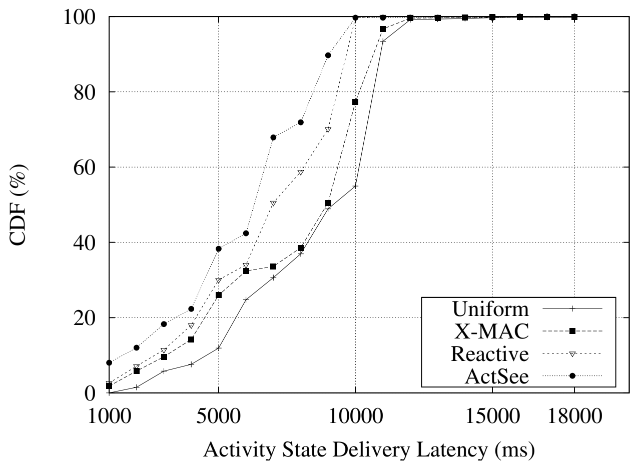
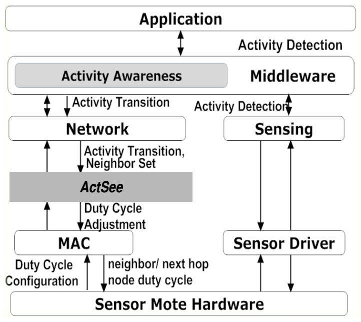


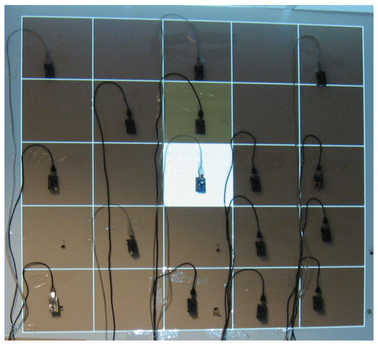
Fig. 4. Distribution of data delivery latency.

IV. Experimental Study and Performance Evaluation

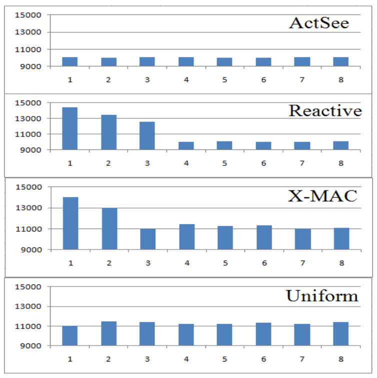
The performance of the proposed *ActSee* protocol is evaluated both in (a) a real sensor network testbed, and (b) a sensor network simulator TOSSIM [12]. The experiments with probabilistic event activity transitions are performed with networkwide data collection at the sink node. In the experiments, the active source nodes send activity detection data to the sink node through forwarder nodes on a multi-hop path. In addition to the activity data, each node periodically reports the energy level and other status information to the sink. As the base routing protocol we use the shortest path routing, although *ActSee* can operate with any routing protocol. The whole system is implemented in a manner suitable for real-time applications. The network topology information and event activity data are collected at the sink node, and transferred to the back-end system for $ATPG$ graph generation and solving the Linear Program (using standard method). The back-end system computes the optimal duty cycling strategy, as described in Algorithm 2. Then the resulting action set (duty cycle assignments) from Linear Programming solver, is disseminated back into the network once. The nodes perform



(a)

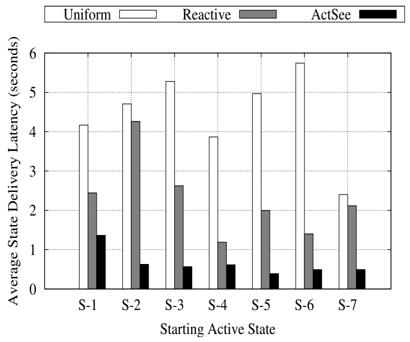


(b)

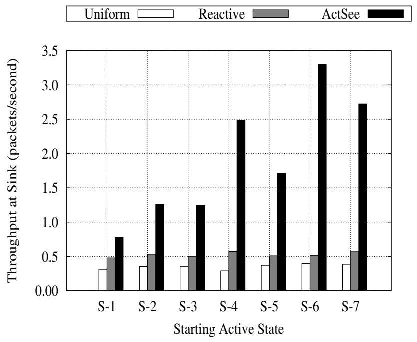


(c)

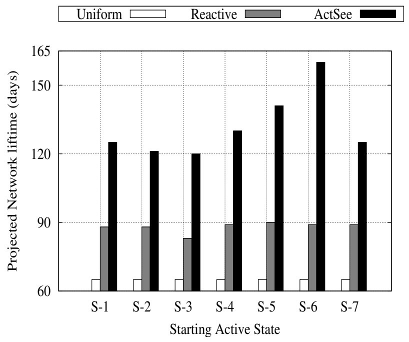
Fig. 5. (a) TinyOS node software stack with activity-aware design for *ActSee*. (b) Testbed emulating motion activity event with projected light beam. (c) Delivery latency of packets after event occurrence for various duty cycling strategies.



(a) State delivery latency (seconds) for different starting active state



(b) Throughput at Sink (packets/second) for different starting active state



(c) Network Lifetime (days) for different starting active state

Fig. 6.

the routine described in Algorithm 1. It is important to note that after forming $ATPG$, the duty cycle assignments are disseminated into the network only once. Afterwards, if a node is active, it sets the duty cycle of neighbors and far-off nodes through communication of local beacon message sharing. So until a node dies, or a new node is added, or the activity pattern changes (that can happen only in long time period, typically at least several days, in smart environments), the duty cycle assignment strategy stored in the nodes is not changed. Therefore, the networkwide dissemination of the duty cycle set is performed very rarely. Thus there is minimum communication overhead for setting of duty cycles. The software system is implemented in TinyOS-2.x.

Figure 5(a) shows the sensor node software architecture for $ActSee$. The activities detected by the sensing layer is processed in the application layer. The middleware stores the action set for duty cycle assignment. Now with the current action set, the membership of node (being in the active set V_A or inactive set V_I) and the neighbor set information (from network layer) are transferred to the link layer. Then the node reconfigures its own duty cycle and sends beacon message to the neighbors to let them reconfigure their duty cycles accordingly. The energy consumption is calculated using the relevant model of radio transmission, reception and radio idle states. In our experiments, the performance of *ActSee* is compared with *Uniform* duty cycling, *X-MAC* (frequently used in sensor network applications) and *Reactive* strategy. Recall that *X-MAC* represents the class of asynchronous MAC protocols. Moreover, the performance of the Synchronous TDMA MAC protocol performance will be equivalent to *Uniform* where the individual node duty cycles are fixed.

A. Evaluation on Real Sensor Network Testbed

A network of 16 TelosB sensor motes is deployed according to the physical deployment layout of motion sensor nodes in the kitchen and dining room of *CASAS* smart home [1]. The transmission power of the radio is controlled to generate a multi-hop network in the testbed area. Now light beam is projected in the space, and is programmed to move according to the learned activity transition patterns in $ATPG$. This emulates the detected motion activities of the corresponding smart home residents. The standard one-hop broadcast beacon message is utilized in *ActSee* and reactive duty cycling protocols to piggyback extra information to share among the neighbors. A node sensing light intensity above a threshold emulates detected activity. Each experiment with different duty cycling protocols was performed for two hours, in order to generate a variety of probabilistic order of event activities. The experimental set up for comparing different protocols are as follows.

The fixed duty cycle in the *Uniform* strategy is selected at the value to exactly satisfy the previously described *device energy consumption budget* (for achieving at least a minimum network lifetime). In the *Reactive* strategy, the nodes by default maintain minimum duty cycle in the duty cycle assignment set $\mathcal{D}$. But they reactively reconfigure the duty cycle of nodes on an active route to the maximum duty cycle in $\mathcal{D}$. In *X-MAC*, the sleep period is selected in order to maintain a projected average duty cycle satisfying the *device energy consumption budget*. Figure 4 shows the distribution of the activity state delivery latency. In *Uniform* and *X-MAC*, around only 50% of the packets are delivered with latency within 9000 ms. In *Reactive*, 70% of the packets are delivered within latency of 9000 ms. Finally, in the *ActSee* protocol, 92% of the packets are delivered within 9000 ms. Therefore *ActSee* provides much reduced data delivery latency.

Figure 5(c) shows instances of delivery latency of 8 consecutive packets delivered at the sink after an event occurs. *Uniform* has the same high latency. *X-MAC* has better latency after initial wake-up with preambles, but still suffers from high latency due to insufficient duty cycles of nodes on the active route. *Reactive* performs better from 4th packet, after reactive setting of duty cycles. But *ActSee* maintains low duty cycle for all the packets delivered after the event occurrence. This is very important for time critical applications, where the delivery latency of the very first packet (after event occurrence) is equally or more important than the subsequent ones. *ActSee* achieves this improvement in delivery latency by setting active paths from possible next active nodes to the sink, with high duty cycle. This also leads to better throughput of the collected data at the sink node in *ActSee*, as compared to others. *ActSee* also intelligently saves energy by configuring the idle nodes in the network with low duty cycle. *ActSee* provides the best expected network lifetime (the time between network boot-up and the time when the first node dies).

	Uniform	*X-MAC*	*Reactive*	*ActSee*
Throughput (packets/sec)	7.54	8.52	9.65	12.55
Network Lifetime (days)	91	112	136	221

B. Evaluation on Sensor Network Simulator

In the standard sensor network simulator $TOSSIM$ set up, we use a network of 28 nodes in the same layout as the real-time *CASAS* smart home testbed (Figure 2). The computed ATPG information is used to generate probabilistic activity transitions among the nodes. Each experiment with different starting states was 60 minutes long, and repeated 20 times to get the average performance and to vary the probabilistic activity transitions.

Figure 6(a), 6(b) and 6(c) show the mean data delivery latency (in seconds), the data throughput at sink (packets/second), and the projected network lifetime (in days) respectively, for each starting state or active node $S-1$, $S-2$, $S-3$, $S-4$, $S-5$, $S-6$ and $S-7$. A state $S-i$ $(1 \leq i \leq 7)$ denotes one select node selected from the rooms (kitchen, bedrooms, dining room, etc.) in *CASAS* smart home layout. So if the starting state of activity is different, the order of active states (due to motion of smart home residents) will be different. Our experiments clearly demonstrate that in terms of all the relevant parameters (latency, throughput, lifetime), the proposed protocol *ActSee* significantly outperforms both *Reactive* and *Uniform* duty cycling strategies. For example, the mean state delivery latency in *ActSee* is 67% to 92% better than *Uniform*, and 44% to 85% better than *Reactive*. The data throughput at the sink in *ActSee* is 2.4 to 8.3 times better than *Uniform*, and 1.6 to 6.4 times better than *Reactive*. Similarly, *ActSee* outperforms others in the network performance metric (e.g., network lifetime). The projected network lifetime in *ActSee* is 1.8 to 2.4 times higher than *Uniform*, and 1.3 to 1.8 times higher than *Reactive*.

V. Conclusion

This paper presents *ActSee*, a novel activity-aware radio duty cycling protocol for wireless sensor networks. This activity-aware protocol design learns from event activities in smart environments and utilizes knowledge from an activity transition probability graph to dynamically configure the optimal duty cycling strategy in order to provide improved data delivery latency and throughput, while enhancing energy efficiency and hence network lifetime. The experimental results from real sensor network testbed and simulation validate the advantages of the *ActSee* protocol. There are several interesting directions for possible future work such as relaxing the fixed routing policy constraint, thereby allowing dynamic routing strategy.

References

[1] CASAS Smart Home Project. http://ailab.wsu.edu/casas/.

[2] Mavhome. http://ailab.uta.edu/mavhome/.

[3] T. Dam and K. Langendoen, An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks, In *ACM SenSys* 2003.

[4] W. Ye, J. Heidemann and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 2002.

[5] M. Buettner, G. Yee, E. Anderson and R. Han. X-MAC: a short preamble MAC protocol for duty cycled wireless sensor networks. In *ACM SenSys* 2006.

[6] J. Polastre, J. Hill and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *ACM SenSys* 2004.

[7] G. Lu, D. De, M. Xu, W-Z. Song and B. Shirazi. TelosW: Enabling Ultra-Low Power Wake-On Sensor Network. In *IEEE INSS* 2010.

[8] A. Lazar. Optimal Flow Control of a Class of Queuing Networks in Equilibrium, In *IEEE transactions on Automatic Control,* 1983.

[9] P. Nain and K. W. Ross. Optimal Priority Assignment with Hard Constraint. In *IEEE transactions on Automatic Control,* 1986.

[10] Y. Wang, B. Krishnamachari, Q. Zhao, and M. Annavaram. Markov-Optimal Sensing Policy for User State Estimation in Mobile Devices. In *ACM/IEEE IPSN,* 2010.

[11] E. Altman. Constrained Markov decision processes. in *Chapman & Hall*

[12] P. Levis, N. Lee, M. Welsh and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *SenSys*, 2003.