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

Debraj De† Shaojie Tang⋆ Wen-Zhan Song† Diane Cook‡ Sajal Das*
†Department of Computer Science, Georgia State University
⋆Department of Computer Science, Illinois Institute of Technology
‡School of Engineering and Computer Science, Washington State University
*Department of Computer Science and Engineering, University of Texas at Arlington
†{debraj_de, songwz}@wsu.edu, ⋆stang7@iit.edu, ‡djcook@wsu.edu, *das@cse.uta.edu

***Abstract*—In this paper we present *ActSee*, an activity-aware radio duty-cycling protocol that utilizes the learned activity pattern information to intelligently adjust radio duty-cycles. 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 transition patterns in form of Activity Transition Probability Graph ($ATPG$), $ActSee$ solves a linear programming problem to select the duty cycles for a set of 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 pattern information from real Smart Home testbed CASAS [1]. The activity events are generated based on that pattern, in the form of $ATPG$. Experimental results from both real sensor network testbed and sensor network simulator TOSSIM validate that our proposed activity-aware radio duty cycling protocol *ActSee* outperforms the existing designs (traditional MAC protocols, uniform duty cycling, reactive duty cycling) 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 environment, smart home.**

I. Introduction

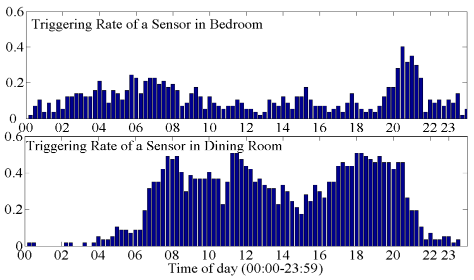


Fig. 1. Change of activity pattern in 24 hours with time and space.

Significant advancement in wireless communication, micro-electronic technologies and distributed protocols have revealed the great potential of wireless sensor network systems for pervasive computing applications. But many existing works ignore a fundamental difference between sensor networks and traditional networks. Unlike traditional network (which is mainly focused on end-to-end data transfer), being deeply embedded in the physical environments, sensor networks' computation and data communication are triggered by environment situations or activities. In many applications such as Smart Environments, the information of event activity shows certain patterns in long run (as shown in Figure 1). Most of the sensor network designs till date under-utilized the activity pattern for performance improvement of application as well as the network. To note that in this paper activity is meant by the events that are sensed and reported by the nodes in sensor network. For example in a Smart Home environment, one type of activity is detected motion events of the residents. These sensed activites are collected by the network for application purpose. But *there remains a missing link in sensor network design: the feedback from sensed and analyzed activity pattern, back to the network operation for resource usage*. The activity pattern information, if utilized in an intelligent manner, can improve the sensor network performance while reducing resource usages. For an instance, Figure 1 shows an example of daylong average activation (motion detection) rate of two motion sensors (in different rooms) in real-time *CASAS* Smart Home testbed [1]. The triggering rates for two motion sensors, one in bedroom and another in dining room, are calculated using 24 hour data collected from 57 days of operation. It can be observed that there are distinct activity patterns in time and space. These patterns can be intelligently utilized for distributed and autonomous control of communication resource usage in such sensor networks. In this scenario our

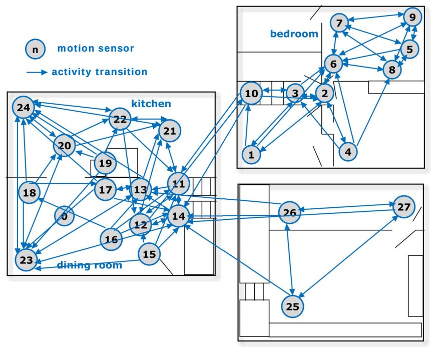


Fig. 2. Activity Transition Probability Graph ($ATPG$) learnt from the *CASAS* Smart Home testbed [1]. The significant transition probability for example from node 27 to nodes 14, 25 and 26 are 12%, 45% and 40% respectively

activity context aware design tries to intelligently adapt the network resources using intelligence from detected activity patterns.

In this perspective we have designed an efficient activity-aware wireless radio duty cycling protocol *ActSee*. The novelty of *ActSee* is that it learns and adapts from the sensed activities in network, to decide the policy of updating node duty cycle. But most of the existing duty cycling methods update duty cycle based on the generated data traffic in the network and in the communication medium. In our proposed system, an AI agent is trained to extract activity patterns from the periodic event activities (e.g. daily activities and situations in smart home). The agent provides Activity Transition Probability Graph (a real-time example is shown in Figure 2), which contains information about transition probability of daily activities in a smart home. The figure shows the Activity Transition Probability Graph with 28 nodes (learnt by an AI agent from the *CASAS* Smart Home testbed [1]) in the Smart Home layout. Using this intelligence about predicted activities, the radio duty cycle of the nodes are dynamically configured for optimizing resources usages and performance, through radio sleep/wakeup scheduling.

To note that energy efficiency is a very critical requirement for sensor networks, even for the Smart Environments. Factors like Plug-and-play, large scale, flexible location deployments etc. require battery powered sensor nodes, thus limiting energy availability. The main contribution of this work is designing an activity-aware radio duty-cycling protocol for sensor networks, to reduce event delivery latency (to the base station), while meeting energy efficiency requirement, all by utilizing available activity context information. In the system model there is a transition in event activity (change in set of active nodes) at the end of each time period. For each sensor, during current time period *ActSee* pre-computes the duty cycle of nodes for the next period (next transition of activity) based on the current activity state information. To note that *activity state* means the conditions or the status of motion activity (e.g. ON/OFF) detected by set of all the nodes in the network. Now this updated duty cycle assignment is propagated to the corresponding sensor nodes during current period or stage. Thus each sensor node can receive its "future" duty cycle assignment in the current stage. The problem of selecting proper duty cycle assignment for the next stage or period 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 constructed as follows. In section II we discuss the related works and motivation behind our work. Then in section III we present background work for our protocol, followed by proposal of *ActSee* protocol in details. Next in section IV we present experimental results and performance evaluation of *ActSee*. Finally we conclude this paper in section V.

II. Related Work and Background

There is considerable amount of sensor networks deployments in Smart Environments, such as CASAS [1], MavHome [2] etc. Most of these works deal with extracting activity patterns and detecting different events for automating and facilitating application services e.g. assisted living etc. But in all these, detected activity patterns are not utilized back in form of activity-awareness for optimizing network operation and resource usage.

There are significant amount of works done on MAC (Media Access Control) protocol for wireless sensor networks. Existing MAC protocols can be categorized into two types: synchronous and asynchronous approaches. Synchronous MAC protocols specify the period of wake-up and sleep for communication to reduce the unnecessary time and energy wasted in idle listening. Nodes periodically exchange SYNC packets for synchronization and data transfer in the common active schedule. S-MAC [4], T-MAC [3] etc. are examples of synchronous MAC schemes. The other type, the asynchronous MAC protocols have no control overhead for synchronization unlike synchronous schemes, in order to improve the energy efficiency compared. Examples of asynchronous schemes are B-MAC [6], X-MAC [5] etc. They rely on low power listening (LPL), also called channel sampling, to let a sender communicate to a receiver which is duty cycling. B-MAC utilizes a long preamble to achieve low power communication. In X-MAC, short preambles with target address information are used to reduce the excessive preamble, instead of a long preamble. When a receiver wakes up and detects a short preamble, it looks at the target address that is included in the preamble. If the node is the intended recipient, it keeps awake for the incoming data, otherwise it goes to sleep immediately. Most of these MAC protocols are just based on data traffic generated in the network, and they don't generally learn from event activity patterns and exploit them. So activity context aware radio duty cycling in sensor networks is relatively unexplored and under-utilized.

Based on the existing works the following radio duty cycling strategies can be adopted for activity detection sensor network systems. (a) The first strategy is *Uniform duty cycling* that lets each node operate at the same radio duty cycle. However this simple and easy to implement strategy is inefficient under activity context aware environment. This is because it fails to leverage the underlying activity transition pattern. For example in Smart Home, during early morning when the Smart Home residents are expected to be waking up from sleep, it is not necessary to keep all the sensors active with higher duty cycle, but only those sensors which are located in the bedroom, and nodes on active route from them. (b) The second strategy is *Reactive duty cycling*. Only when a node successfully detects the present of an activity, it starts to increase the duty cycle of itself and the sensors on the routing path from itself to the sink node (i.e. base station). Otherwise, the nodes operate at a low duty cycle. 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 comes from the delay involved in delivery of data packets indicating detected events ("decision propagation phase"). In particular, it may take longer 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 very significant bottleneck to the application performance. This is often unacceptable in critical time sensitive applications such as *e.g.*, real-time tracking or monitoring. (c) The third strategy can be existing synchronous or asynchronous MAC protocols. These classes of MAC protocols reacts, based on the decision making after event activity data is generated in the network. But they don't have intelligence to exploit the typical activity patterns of the environment. This is also validated through experimental results presented later.

CMDP [11] has been widely used in optimizing network performance subject to resource constraint. Nain and Ross [9] used it in studying traffic problems. Lazar [8] applied CMDP to analyze the throughput maximization problem under delay constraints in telecommunication applications. Very recently, Wang *et al.* [10] has investigated optimal sampling problem based on CMDP.

To our knowledge there doesn't exist much work on duty cycling protocol that learns detected activity patterns and then adapts according to it. Motivated by the shortcomings of existing strategies, in this paper we have proposed an efficient duty cycling strategy that tries to achieve two goals, that are apparently hard to satisfy together: low data delivery latency and higher network lifetime. Besides the novelty of activity-awareness, $ActSee$ is also unique in achieving both these goals. $ActSee$: (i) maintains high duty cycle for nodes on the routing paths for active and predicted (predicted to be active in next stage or period, according to activity pattern) data sources (for fast and reliable data delivery), (ii) maintains low duty cycle (for energy saving) for potentially idle nodes which are not predicted to be active next, and (iii) has 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.

III. Activity-Aware Duty-Cycling

A. System Model

The $ATPG$ for a particular Smart Environment can be constructed based on knowledge of observations collected in the environment. The graph node pairs which represent pairs of sensors that can be reached directly from each other (without triggering another sensor) would be connected by an edge in the graph. Our approach builds an $ATPG$ based on observed sensor events that occur in the environment. In this way, we can estimate the probability of transition between two sensor nodes x and y based on the relative frequency with which residents of the environment actually trigger sensor node x followed by sensor node y. Figure 2 shows the floorplan and sensor layout for the CASAS smart home testbed that we refer in this work. When an $ATPG$ graph is generated, a node is generated for each sensor that exists in the environment. The probability associated with edge x, y is estimated using the formula in equation 1:

$$p(x,y) = \frac{\mid events\ for\ sensor\ x\ followed\ by\ sensor\ y \mid}{\mid events\ for\ sensor\ x \mid} \tag{1}$$

The system model is described with respect to a Smart Home. This applies generally to all Smart Environments. In a Smart Home for assisted living and health monitoring, at certain time of the day, a resident's Activities of Daily Living (ADL) in a region is monitored by a set of sensors. Based on the sensor(s) detecting the current activities, the system has a state, say x_i. Then after some time the system has another state x_j due to motion activity of a person. In specific, 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 $1 \leq i, j \leq N$). Each state x_i is associated with a set of active sensors at that state, *e.g.*, those sensors which detect the activity. For simplicity of presentation, we assume that there is only one active sensor at each state. In practical, there may exist a cluster of active sensors at the same state, then we can treat this cluster as a single sensor in current work. We denote the sensor that is associated with state x_i as v_i, and we say x_i is v_i's state. For any state x_i, we define the neighbors of x_i as those states $x_j \in \mathbf{X}$ which have positive transition probability $p(i,j) > 0$ from x_i.

The *state report latency* is defined as the time it takes from the moment a activity enters a new state to the moment the sink node has been informed of that activity. In a typical wireless sensor networks, a state report latency usually includes *state detection latency* and *state delivery latency*. The *state detection latency* is defined as the time duration from the instance there is a motion activity occurred, to the instance a sensor has detected this activity. In this work, we achieve almost zero detection latency mainly through hardware sensor wake-on capability [7]. The *state delivery latency* is defined as the time duration from the moment the activity event has been detected by sensor to the moment the sink receives the event data successfully. By assuming the state detection delay is zero, in this work, our focus is designing a context-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.

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 cycles for each node v. We restrict the model by requiring that $\mathcal{D}_v = \mathcal{D}$ for all v. This simplifies notation and does not impact the theory. Typically $\mathcal{D}$ can be: $\mathcal{D} = \{2\%, 5\%, 8\%, 10\%, 15\%, 20\%, 25\%\}$. As described later, we have formulated the optimal duty cycle assignment problem problem as a Markov Decision Process (MDP), where decisions are made at points of time referred to as decision stages or periods. Time is divided into *stages*, $T = \{t | t = 0, 1, 2, \cdots\}$. At the start of each stage, the decision maker observers the system in state $x_i \in \mathbf{X}$, and choose action a_j from the set of allowable actions in state x_i. 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

The expression of state delivery delay is utilized in formulating *ActSee*. For this purpose, in this subsection we have validated 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 network in sensor network simulator (*TOSSIM*) for computing 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 in Figure 3(a) and Figure 3(b), it can be approximated that, f_L is fairly linear with h ($f_L \propto h$), and is inversely proportional to d ($f_L \propto \frac{1}{d}$). Therefore, if routing path is known and the duty cycle is determined for the nodes on 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 system set up factors like network topology. This estimation of state delivery latency can also be verified theoretically (in standard network model). Assuming all the nodes are synchronized, the nodes on selected route maintain same duty cycle d. Then the message can travel $d \cdot T / T_{trans}$ hops during one period, where T is the radio duty cycling period and T_{trans} denotes the transmission time for one packet over one link. After that the message has to wait during radio sleep time in the intermediate forwarder node. Then it is forwarded again through $d \cdot T / T_{trans}$ hops. Since nodes are strictly synchronized, so their radio duty cycle periods align in time. This forwarding of data through radio ON period is performed $\frac{h}{d \cdot T / T_{trans}}$ times before arriving at the base station, where h is the hopcount of source node from base station. 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 a fixed value, it validates our estimated model for delivery latency: $f_L = c \cdot \frac{h}{d}$. Besides hop distance and duty cycle, collisions and congestions among links may also affect the delivery latency. Here we have assumed that there are negligible collisions and congestions effect, since collisions and congestions are unlikely to happen in our sensor networks scenario, where the traffic is low or moderate (e.g. 48 byte packet generation in every 5 seconds during activity detection in our Smart Home sensor network deployment). The collision and congestion are low in our deployed system because, at a time there are only a limited number of active nodes generating data at low rate, and the wireless frequency chosen doesn't overlap with other sources of radio signals in a Smart Environment (WiFi, microwave etc.).

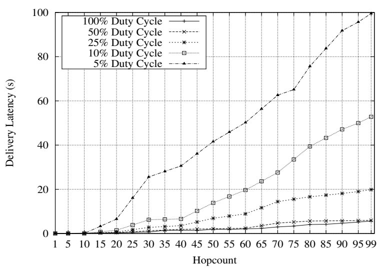
C. Activity-aware Radio Duty-cycling

The goal of $ActSee$ is first reinstated with formulation. We define D (different from duty cycle set parameter $\mathcal{D}$) 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 D and activity state transition matrix P, what is the optimal radio duty cycling strategy μ, such that the expected activity state delivery latency $E[L]$ is minimized, and the expected average duty cycle is maintained under the budget, e.g., $E[d] \leq D$?*

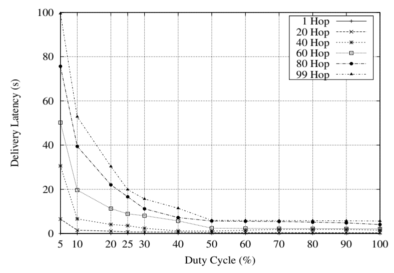
ActSee needs each node to know its hopcount to base station, and the next hop on route. It is worth mentioning that whatever routing path the nodes choose, it will not affect the results derived in *ActSee*. The program running *ActSee* is just needed to be informed of any change in routing path (with changed hopcount to sink). For ease of explanation, during any state x_i, we partition all the sensors into two disjoint sets: inactive set V_I and active set V_A. Thus V_A contains all the sensors that are associated with x_i's neighbor states (in ATPG), as well as those appearing in their routing paths. Notice that 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 sensors that may become active or help to relay 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. To eliminate the decision propagation delay existing in reactive strategy (discussed in Section II), *ActSee* works in a way as described next.

Based on the available nodes in the network and the collected intelligence about activity transitions in 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). To 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). So for a deployed network, the back-end system calculated 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 disseminates the computed slot assignment among neighbors and further nodes. It's worth noting that $ActSee$ *exploits the existing beacon message* in routing protocol to piggyback the slot assignment information. Thus it saves energy for distributed slot assignment task. Until some node is dead or some new node is added, the back-end system doesn't need to recompute and disseminate the action set. Otherwise it recomputes action set based on periodically collected regular node status information. The activity pattern typically repeats at each smart environment after a reasonable learning period, thus the linear program routine to select the action set does not run often. After learning phase *ActSee* conserves energy in a long period, since the it is computed based on $ATPG$.

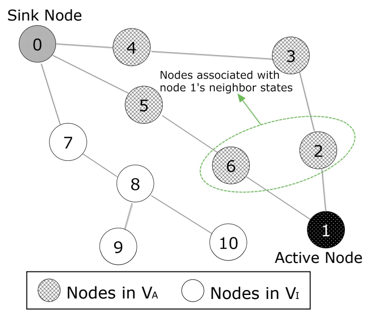
Now, (a) At the beginning of each stage, based on the current active sensor v_i uses it's duty cycle assignment for each sensor in V_A for the <u>next</u> stage. The detailed selection of a proper duty cycle assignment is explained in the next subsection based on the duty cycle selection strategy μ. Then the decision will be propagated to V_A immediately during



(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 current active node, node 2 and node 6 are 1's two neighbors. The routing path from node 2 is 2-3-4-0, and the one from node 6 is 6-5-0. In this example, V_A contains nodes 2, 3, 4, 6, 5; V_I contains the remaining nodes. 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 at lowest duty cycle

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 sensors 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
```

current stage. By executing the decision propagation phase during current state, we are able to reduce the decision propagation delay. (b) For the remaining sensors which do not receive any updated duty cycle information, they will work at the lowest duty cycle during the next stage to save energy.

We next use the example in Figure. 3(c) for illustration. In Figure. 3(c) node 1 is the current active node, nodes 2 and 6 are node 1's two neighbors in ATPG. More precisely, the states of both the nodes 2 and 6 are neighbors of node 1's state in ATPG. In this example, the routing path from node 2 is 2-3-4-0, and the one from node 6 is 6-5-0. Also $V_A = \{2, 3, 4, 6, 5\}$, $V_I = \{7, 8, 9, 10\}$. By following *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, while all the remaining nodes will work at lowest duty cycle. In *ActSee* an active sensor can make a decision immediately based on local information, given it knows it's action set. Please refer to Algorithm 1 for pseudo codes.

In algorithm 1, μ specifies the duty cycle assignment of the nodes at each state. We model the computation of optimal duty cycling strategy problem as Constraint Markov Decision Process (CMDP). By solving the corresponding Linear Programming (LP) in polynomial time [11], we obtain an optimal strategy μ for each state. For each state, the selection of different duty cycle 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. We focus on discrete time models: we observe the system at stages $t = 1, 2, \cdots, n$ where n is called horizon, and may be either finite or infinite. A controller has an influence on both the costs and the evolution of the system, by choosing at each time unit some parameters, called actions. As is often the case in control theory, we assume that the behavior of the system at each time is described by a "state", as well as the "action". The system moves sequentially between different states in a random way, current state fully determines the probability to move to any given state in the next time period unit. In a standard MDP, current action may also affect the transition probability for the next time unit, but this is not the case in this paper where the transition graph does not depend on current duty-cycle algorithm. MDP is thus a generalization of (non-controlled) Markov chains, and many useful properties of Markov chains carry over to controlled Markov chains. The model that we consider in this paper is special in a sense that more than one objective cost exists, and the controller minimizes one of the objectives subject to constraint the other.

To make the above fit our problem setting, we define a tuple $\{O, \mathbf{X}, \mathrm{A}, \mathcal{P}, L, D\}$, where: (a) $O = \{t | t = 1, 2, \cdots\}$ denotes the set of decision epochs. Decisions are made at the beginning of each stage. (b) $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$ is a countable state space. Although we limit the study in this work to discrete activity state transitions, the continuous case can also be handled by dividing it to discrete space. A is a metric set of actions. We denote $\mathrm{A}(x_i) = \{a_1^i, a_2^i, \cdots\}$ as the compact action set allowable at state x_i. Each action $a_j^i \in \mathrm{A}(x_i)$ defines a duty cycle assignment for each sensor in the next stage. Let $d(a_j^i, v)$ denote the duty cycle assignment for sensor v under action a_j^i. Theoretically, each sensor has $|\mathcal{D}|$ possible duty cycle assignment, thus the action space could be as large as $N^{|\mathcal{D}|}$. In order to reduce the searching space, we again leverage underlying transition graph to facilitate our study. Specifically we restrict the action set $\mathrm{A}(x_i)$ in this work as follows: (i) only the nodes in V_A (possible active nodes in the next state, please refer to Section III-C for details) will be considered to increase the duty cycle, the rest nodes work with lowest duty cycle by default. (ii) all relay nodes that appear at the routing path from the same source node have the same duty cycle as their source node. (iii) for those nodes appearing at the crossing point of multiple routing paths, we set 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 current state in this example. Now: (A) Let $\rho(x_i, a_j^i)$ denote the "occupation measure" of state x_i and action a_j^i, e.g., the probability that such state-action pair ever exists in the decision process. $E_v(d)$ denotes the expected average duty cycle of sensor v, which is expressed as in Equation 2. Notice that the "occupation measure" $\rho()$ is decided by corresponding duty cycling strategy. (B) $\mathcal{P}$ are the transition probabilities. Define $\mathcal{P}_{iaj}$ as the probability of moving from system state i to j, when action a is taken. Since different duty cycling strategy will not affect the actual transition process of the resident, thus given the activity state transition probability matrix P, it is easy to conclude that $\mathcal{P}_{iaj} = P_{ij}$. (C) L is 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, which is expressed as as in Equation 2. $f_L(v_k \rightsquigarrow v_0, a_j^i)$ 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 neighbor set of state x_i. Please refer to Section III-B for detailed calculation of f_L under given hopcount and duty cycle. Then the expected average delivery latency $E[L]$ can be computed as as in Equation 2. (D) D is the maximum allowed expected average duty cycle. Therefore for each node v, $E_v[d] \le D$.

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

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

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

This constraint describes that the outgoing rate and incoming rate for a state must be same, at the same time it emphasizes the property for ergodic processes. After solving the linear programming, we get an optimal occupation measure $\rho()$ in terms of delivery latency minimization for each state/action pair. However, since $\sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) \le 1$, 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 that 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 state, Algorithm 2 can return an optimal strategy μ in polynomial time. As input of Algorithm 1, $\mu(a_j^i | x_i)$ for all j will be propagated to each corresponding sensor v_i.

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

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

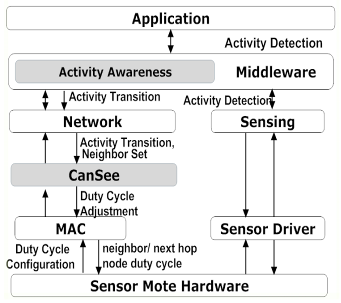
Algorithm 2 Computation of Optimal Duty-Cycling Strategy μ

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

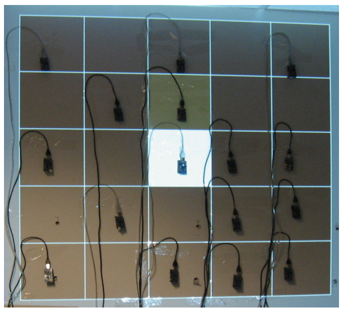
```
1: Solve corresponding CMDP linear programming to get the occupation measure $\rho(x_i, a_j^i)$, $\forall x_i \in X, \forall a_j^i \in A(x_i)$;
2: Calculate optimal duty cycling strategy $\mu$ from $\rho(x_i, a_j^i)$ as:
       $\mu(a_j^i | x_i) = \frac{\rho(x_i, a_j^i)}{\sum_{a_j^i \in A(x_i)} \rho(x_i,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

IV. System Development and Performance Evaluation

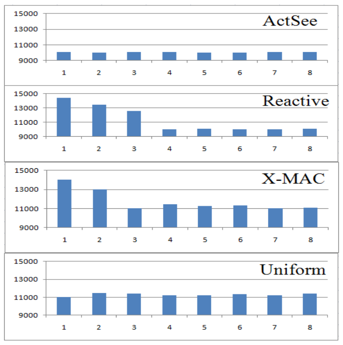
Performance of *ActSee* is evaluated both in: (a) real sensor mote testbed, and (b) sensor network simulator (TOSSIM) [12]. The probabilistic activity transition experiments are performed with data collection at sink node. In the experiments



(a)

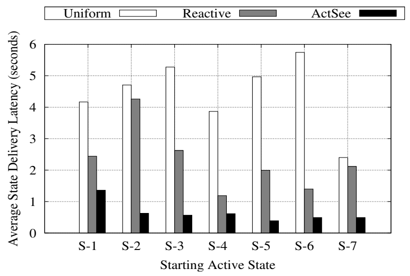


(b)

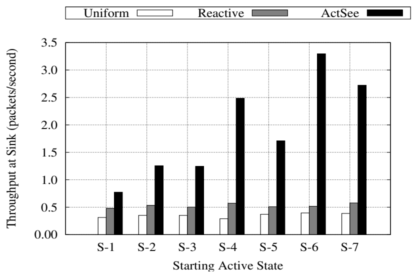


(c)

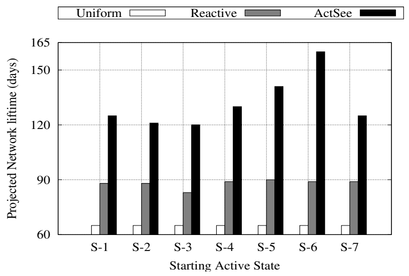
Fig. 5. (a) TinyOS node software stack included with activity aware design for *ActSee* (b) Testbed emulating motion activity event with projected light beam (c) Delivery latency of packets after event occurrence



(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.

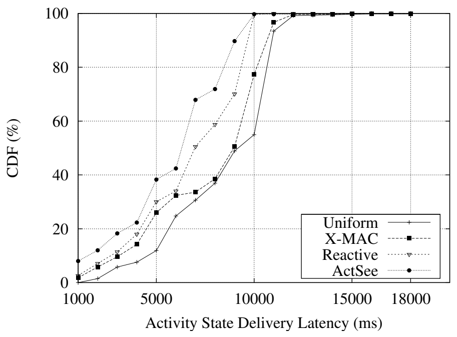


Fig. 4. Distribution of data delivery latency

the active sources send activity data to base station through multi-hop path. In addition each node periodically reports the energy level and other status information to sink. The shortest path routing is used in experiment, 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 initial event activity data is periodically collected at sink and transferred to back-end system for $ATPG$ graph generation and solving the Linear Program (using standard methods). The back-end system computes the optimal duty cycling strategy (as described in algorithm 2). Then the resultant action set for duty cycle assignments from Linear Program solver, is disseminated back into the network once. So the nodes perform the routine described in algorithm 1. It is important to note that after forming $ATPG$, duty cycle assignments are disseminated into the network only once. Afterwards, if a node is active, it sets the duty cycle of neighbors and further 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), duty cycle set strategy stored in nodes is not changed. So the networkwide dissemination of duty cycle set is done very rarely. Therefore there is minimum communication cost for setting of duty cycle. The software system is implemented in TinyOS-2.x (Figure 5(a)). The activities detected by sensor layer is processed in application layer. The middleware stores the action set for duty cycle assignment. Now the current action set, the membership of node (being in active region V_A or in-active region V_I) and neighbor set information (from network layer) are transferred to the link layer. Then the node reconfigures it's own duty cycle and sends beacon message to neighbors to let them reconfigure their duty cycle accordingly. The energy consumption is calculated using the relevant model of radio transmission, reception and radio idle states. In experiments the performance of *ActSee* is compared with *Uniform* duty cycling, *X-MAC* (frequently used in sensor network applications) and *Reactive*. *X-MAC* represents the class of asynchronous MAC protocols. Synchronous TDMA MAC protocol performance will be equivalent to *Uniform*.

A. Evaluation on Real Sensor Network Testbed

A network of 16 TelosB Motes is deployed according to layout of motion sensors in kitchen and dining room of *CASAS* smart home. The transmission power of radio is controlled to generate a multi-hop network in the area of testbed. Now the a light beam is projected and programmed to move according to the learned activity transition pattern in ATPG. 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 detecting light intensity above threshold emulates detected activity. Each experiment with different duty cycling protocols were 2 hours long, in order to generate a variety of probabilistic order of activities. The experimental set up for comparing protocols are as follows. The fixed duty cycle in *Uniform* is selected at the value to just satisfy the previously described *device energy consumption budget* (minimum network lifetime). In *Reactive*, the nodes by default maintain minimum duty cycle in duty cycle assignment set $\mathcal{D}$. But they reactively set the duty cycle of nodes on 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 activity state delivery latency. In *Uniform* and *X-MAC*, around 50% of the packets are delivered with latency within 9000 ms. In *Reactive*, 70% of the packets are delivered within latency of 9000 ms. But in *ActSee* protocol 92% of the packets are delivered within 9000 ms. *ActSee* provides much reduced delivery latency. Figure 5(c) shows instance of delivery latency of 8 packets after an event occurs. *Uniform* has 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 after event occurrence. This is very important for time critical applications, where delivery latency of very first packet (after event occurrence) is equally or more important than subsequent ones. *ActSee* achieves this improvement in latency by setting active paths from possible next active nodes to the sink with high duty cycle. This also leads to better throughput of collected data at sink node in *ActSee*, compared to others (shown in Table IV-A). But *ActSee* also intelligently saves energy by configuring idle nodes in network with low duty cycle. As shown in Table IV-A, *ActSee* provides the best expected network lifetime (the time between network boot-up and the time when 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 TOSSIM

In standard sensor network simulator $TOSSIM$ set up, a larger network of 28 nodes in the same layout as real-time *CASAS* smart home testbed (Figure 2). The computed ATPG information is used to generate probabilistic activity transitions in nodes. Each experiment with different starting states were 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 state data delivery latency (in seconds), the data throughput at sink (packets/second) and projected network lifetime (in days) respectively, for each starting state (starting active node) $S-1$, $S-2$, $S-3$, $S-4$, $S-5$, $S-6$ and $S-7$. The states $S-i$ ($1 \leq i \leq 7$) are one individual nodes for each of 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. Now it is clearly observed that in terms of all the parameters (latency, throughput, lifetime) *ActSee* significantly outperforms both *Reactive* and *Uniform*. Mean state delivery latency in *ActSee* is 67% to 92% better than *Uniform*, and 44% to 85% better than *Reactive*. The data throughput at sink in *ActSee* is 2.4 to 8.3 times better than *Uniform*, and 1.6 to 6.4 times better than *Reactive*. Now for the network performance metric (network lifetime) also, *ActSee* outperforms others. 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 that learns in Smart Environment wireless sensor networks. The activity-aware design utilizes knowledge from 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 for longer network lifetime.

References

[1] Casas smart home project. http://ailab.wsu.edu/casas/.

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

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

[4] Ye, Wei and Heidemann, John and Es, Deborah 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] Buettner, Michael and Yee, Gary V. and Anderson, Eric and Han, Richard X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks In *ACM SenSys* 2006.

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

[7] Lu, Gang and De, Debraj and Xu, Mingsen and Song, Wen-Zhan and Shirazi, Behrooz 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] Altman, E. Constrained Markov decision processes in *Chapman & Hall*

[12] Levis, Philip and Lee, Nelson and Welsh, Matt and Culler, David TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications In *SenSys*, 2003.