Abstract—This paper presents an efficient mechanism to achieve significant energy savings in heterogeneous wireless sensor networks such as HPWREN. Environmental sensors are typically located in remote and hard-to-reach locations far from the main high-bandwidth data links. Therefore, the sensed data needs to be routed through multiple hops before reaching the backbone. The routing is done by battery-powered nodes using license free radios such as 802.11. In this context, minimizing energy consumption is critical to maintaining operational data links. We propose a distributed low-power scheduling algorithm that limits the number of active nodes and allows a large portion of nodes to sleep. The algorithm works on a completely distributed manner, requiring handshaking only when a new entrant exits the network. Routing of packets relies on a dynamically changing backbone of nodes. The nodes of the backbone provide connectivity and are responsible for delivering the packets to the proper destinations. The proposed solution sits on top of the unmodified MAC layer so that legacy network devices can be used, and expensive hardware/software modifications are avoided. Thus our solution is cheap yet easily deployable. Our results show up to 60% energy savings per battery operated node while maintaining efficient data delivery.

I. INTRODUCTION

Environmental research and observation gain significantly from the ease of access to data and control enabled by heterogeneous wireless sensor networks such as HPWREN [3]. The High Performance Wireless Research and Education Network (HPWREN) is a collaborative cyber infrastructure for environmental research, education, and first responder activities deployed in southern California (covering nearly 20,000 square miles). Project researchers use commercial off-the-shelf components (COTS) to create access networks to the backbone of HPWREN for numerous sensors placed in the field. HPWREN is used by many scientific disciplines that monitor and sense the environment, ranging from environmental sciences, oceanography, and astronomy to rural education and first responder units. Its sensors come with varying resource requirements, such as large bandwidth requirements of Palomar observatory, medium bandwidth but tight real-time traffic deadlines of video cameras tracking wildlife, and long battery lifetime requirements of small and remotely deployed weather stations.

There are a number of challenges in such wireless networks. Ensuring that node battery lifetime is long enough so that data collection and delivery can happen in timely manner is of critical importance. Furthermore, because some of the applications have data that urgently needs to be delivered (e.g. first responders in HPWREN), there needs to be a method to trade off energy with QoS. Lastly, the network is built out of COTS to ensure low cost and ease of deployment. Thus, the main contribution of this work is an adaptive and energy efficient scheduling and routing backbone creation algorithm capable of saving up to 60% in energy while ensuring timely data delivery and use of COTS. In contrast to previous work [5], our algorithm does not require any changes to the MAC and provides better performance in terms of latency.

A heterogeneous wireless sensor network such as HPWREN can be described using a three-layer structure shown in Figure 1. The top layer represents the wireless mesh backbone of HPWREN. The links between the backbone nodes (or parent cluster heads - parent CH) are provided by high-speed wireless directional antennas that are typically deployed on mountaintops and have line power accessible. In this layer, policy based routing and a level of QoS is provided by the routers. The bottom layer contains sensors. There are many algorithms that have been developed to address the energy efficiency of such sensor networks, and thus this is not the focus of our work.

The middle layer of Figure 1 is composed of a wireless network of child cluster heads (child CHs). Each child CH node gathers the data coming from the underlying sensors and delivers it to the proper locations in the field. These nodes frequently also perform data analysis and processing. Data can be routed through other child CHs before reaching the parent CH mesh network (the top layer). The child CHs use license exempt radios and are typically battery powered. As a result, child CHs need to maximize battery lifetime to ensure timely delivery of the sensor data. For example, Santa Margarita Ecological Reserve (SMER) uses child CHs with 802.11b connectivity to collect real time weather data from an array of weather sensors that cover several different microclimates in SMER.

The solution described in this paper is designed to fit the needs of networks such as the one found in the middle layer of the HPWREN. We focus on optimizing data delivery via wireless network interfaces (NIC) since that accounts for a significant fraction of the overall energy consumption. Frequent battery replacements are an expense not only in terms of hardware but also in terms of manual labor, since child CHs can be located in very remote areas. Figure 2 shows the two main components of our solution: the scheduler and the backbone creation and maintenance algorithm, while Figure 3
Fig. 1. HPWREN: three layer structure

Fig. 2. High level overview of the proposed solution

Fig. 3. Overview of the proposed solution

shows an example of how our solution affects nodes for a specific time slot. The scheduler limits the number of active nodes in the network for each TDMA slot (Scheduled-ON nodes in Figure 3), allowing a large portion of them to switch off the NIC and save power (Sleep-OFF nodes). Scheduler overview is in Section 3. The backbone algorithm creates a dynamically changing network of a subset of remaining active nodes (Coordinator modes) to ensure reliable data forwarding. The nodes of the backbone are selected periodically in special communication slots (BcastSlots) based on their remaining battery lifetime and their utility (a measure of how many pairs of neighbors the node would connect if it becomes part of the backbone). The information of the scheduled nodes and the coordinator nodes is merged as shown in Figure 3 (Result) and made available to the routing layer.

Since our solution is between the MAC and the routing layers, we ensure an inexpensive, quickly deployable, and flexible solution. MAC layer changes tend to be expensive as they usually involve the design of new hardware, firmware and device drivers. Also, since the routing layer is given the information regarding the active nodes in the network, we provide the flexibility to implement the routing algorithm most suitable for any specific network. To test our ideas with simulations, we implement a greedy geographic algorithm in which a node first attempts to forward a packet to a coordinator that is closer to the destination. If such coordinator doesn’t exist, it then tries its nearest scheduled neighbor. If a forwarder is not found, then a hole is encountered and the packet is dropped. The remainder of the paper is organized as follow. We first discuss the related work in Section 2. Then we present the details of our solution: the scheduling algorithm (Section 3) and the forwarding backbone creation mechanism (Section 4). In Section 5, first we describe the simulation setup and then present results of our experiments. Finally we conclude in Section 6.

II. RELATED WORK

Recent deployments of heterogeneous wireless sensor networks with applications sensitive to latency and/or throughput have raised interest in research activities focusing on QoS-aware and/or energy efficient techniques. Inherent unpredictability of the wireless channel and limitations in design of commonly used MAC protocols lead to difficulty in guaranteeing QoS. For example, IEEE 802.11 MAC uses a random backoff mechanism when collisions occur, thus reducing the overall throughput, increasing the power-consumption and the delay. Much of recent work focused on modifying MAC protocol to improve performance in terms of throughput and/or energy consumption (e.g. PAMAS [7] powers off wireless NIC during transmissions of packets not addressed to the node). Generally, altering MAC implies significant changes in hardware, firmware, and device drivers. It cannot be easily applied to the previously deployed networks without significant additional cost. On the other hand, scheduling above MAC layer gives more flexibility. It can be implemented through software modifications; hence it is more cost effective. Some of the recent work focuses on scheduling data delivery above MAC layer, which we summarize next.

Overlay MAC layer (OML) [15] adds an additional conceptual layer over the existing 802.11 MAC, thus enabling the use of COTS. OML use loosely synchronized time clocks to divide the time in equal size slots and employs a distributed algorithm to allocate these slots among competing nodes. By allowing only one host to access wireless media for a time slot, OML alleviates the unfairness problems including the throughput imbalance among asymmetric sender transmit rates; it uses a fair allocation algorithm with support for arbitrary weights to nodes. SWAN [16] is a rate control mechanism for TCP and UDP traffic which works on the best-effort MAC.

SWAN improves throughput and achieves fairness. Both Overlay MAC and SWAN assume that the nodes in the network continuously listen to a channel, leading to high energy consumption. The TDMA based protocol in [14] gives simpler control scheme; a server periodically broadcasts a
control packet which contains scheduling information of each client station. A client awakes at a predetermined time to transmit a series of data packets after which it transitions to a power-save mode. Since only one station is activated at any given time, it can complete transmission during short interval and stay in power-save mode for a long time. In contrast, we use a distributed mechanism in which nodes can decide when to transit in a sleep state without exchanging control packets with a central server node or the neighbors.

Our scheduling algorithm is based on the ideas presented in [2] and [10]. Scheduling is used to reduce contention and interference in the network. These goals are achieved respectively by using the node-level scheduling algorithm and the cell-level scheduling algorithm. The results reported focused on one-hop performance and energy consumption. In this work we present how a similar scheduling algorithm can be coupled with a novel dynamic backbone creation algorithm to enable multihop routing of data in energy efficient and timely manner.

SPAN [5] also builds a dynamic backbone of nodes to deliver the data throughout the network. While SPAN assumes that every node is always able to transmit and receive, our solution uses scheduling based on TDMA to reduce the energy consumption by placing a large fraction of nodes to sleep. All nodes are active only during infrequent BeastSlots during which the forwarding backbone is created as described in Section 4. Furthermore, SPAN relies on the 802.11 Power Saving Mode (PSM) [4] as its power saving mechanism while we use our low-power scheduling algorithm. SPAN also applies a set of modifications to the PSM thus requiring a new MAC design. These modifications aim to improve performance and energy savings. The optimizations made to the PSM give more opportunities to the nodes to go to sleep, thus increasing power savings. A node with an unmodified PSM would have significantly higher power consumption since it would have to stay awake for a larger proportion of time. Instead, in our solution, we don’t apply any modification to the MAC layer but still achieve significant energy savings with better latency relative to SPAN.

Once a routing backbone is created, an actual routing strategy should be implemented to decide how packets will be delivered. While this is not the focus of our work, many energy-aware routing algorithms have been presented in literature. In [8] several power aware routing metrics that increase the lifetime of the nodes and the network are described. Geographic based routing is described in [9]. In it forwarding decisions are based on the position (geographic coordinates) of a node, its neighbors and the destination. This algorithm [9] draws a line that intercepts the current node and the destination. Next, one candidate above and one below this line are selected by using heuristics that minimize power, cost and the angle formed between the current node, candidate node, and the destination. The next hop that is chosen has a higher probability to be closer to the direction of the destination.

In summary, when compared to previous work, our solution offers a more flexible and low-cost solution that is independent of the specific medium access protocol used in the network, and is fully distributed. It achieves large power savings while delivering data with lower latency. We next outline the scheduling algorithm designed to minimize the energy consumption. Section 4 discusses how to couple our scheduler with a novel dynamic backbone creation algorithm. We analyze properties of our algorithms using geographic routing on an example derived from HPWREN in Section 5, and conclude in Section 6.

### III. The Scheduling Algorithm

The scheduling algorithm gives opportunity to the nodes to save energy by powering off the wireless communication device. At each TDMA slot, it determines in a distributed fashion, which nodes must stay active and which ones can switch off their NIC. The TDMA scheme assumes that the timers of the nodes are loosely synchronized; we allow an error margin of up to a few milliseconds. Therefore, we can use any lightweight timer synchronization mechanism. For instance, the 802.11b TSF [4] is such a synchronization mechanism whose maximum clock offset is within our error margin as shown in the simulation study in [17].

As shown in [2][10], limiting the number of active nodes in a wireless network not only saves power, but also reduces contention achieving higher aggregate throughput. The efficiency of this scheduling technique is proved by the measurements made on a testbed network as reported in [10]. The measurements in [10] validate the simulation results shown in [2]. In summary, the scheduling technique can improve throughput up to 10.3 %, with maximum power saving of 85.54 %.

We define the network graph $G = (V, E)$ where $V$ is the set of vertices that represent the nodes, and $E$ is the set of edges representing the neighbor relationship between the nodes. If a node $v_i \in V$ is a one-hop neighbor of the node $v_j \in V$, then $(v_i, v_j) \in E$. Let $AV$ be the schedule assignment (output of the algorithm), where $AV \subseteq V$. Let $N(v_i)$ be the set of neighbor nodes of $v_i \in V$. At $v_i$, let the set of active nodes which are in $\{v_i\} \cup N(v_i)$ be $AV(v_i)$. Then the set of active nodes in the network is the set $AV$ that is the union of all the $AV(v_i)$. The basic constraint in the scheduling problem is that the number of neighboring active nodes should not exceed the given parameter $S$. Formally,

$$|AV(v_i)| \leq S \quad (1)$$

As we show in the result section (Section V), in terms of energy consumption, an optimal number for $S$ is quite small, which ensures that at any point in time the majority of nodes are actually sleeping, resulting in high energy savings (60%).

Work in [2] shows that an assignment that maximizes the size of $AV$ (called maximum assignment) for $S \geq 1$ is NP-complete. Thus, the scheduler implements maximal assignment: nodes are scheduled so that the number of active nodes is maximal (it does not exceed $S$). To implement scheduling, each node needs only the knowledge of its two-hop neighbors. Given a node $v_i \in V$, the two-hop subnetwork $G_i = (E_i, V_i)$ for node $v_i$ is defined as the set of neighbors in
one- and two-hop distance from \( v_i \), where \( E_i \subseteq E \), \( V_i \in V \). This information is exchanged during the \textit{BeastSlot} used also for the backbone creation. The pseudo-code for the scheduling algorithm is given in Algorithm 1. As a first step, it assigns tickets and pseudo-random numbers to each node in its subnetwork (steps 1-2). The pseudo-random numbers are generated using as seed the sum of the node IDs and the current sequential slot number. The number of initial tickets in each node is equal to the parameter \( S \) of our algorithm. Whenever the algorithm runs, the number of tickets \( tk(v_j) \) is initialized to \( S \) and the random numbers are generated again. By using randomization, all the nodes are given opportunity to be scheduled, as the results in [2][10] show. In step 3, it creates a set of unchecked nodes \( V \) that includes all nodes in \( V_i \). In step 4 the algorithm extract the node \( v_j \) with the greatest pseudo-random number from \( V \). To determine if the node is schedulable (step 5), it checks if the tickets of the active nodes that are neighbors of the unscheduled node \( v_j \), and \( v_j \) itself, is equal to or greater than 1. If so, node \( v_j \) is added to the set of active nodes \( AV \) and the tickets of \( v_j \) and of all its neighbors are decreased (step 6). If the number of active nodes that are neighbors of the unscheduled node \( v_j \) is equal to or greater than \( S \), the number of tickets at \( v_j \) cannot be greater than zero, \( tk(v_i) < 1 \). If the latter is the case, \( v_j \) is not scheduled because \( v_j \) already consumed its tickets. In either cases, \( v_j \) is removed from the set of unchecked nodes \( V \) (step 7). When all nodes in the subnetwork have been checked for schedulability (\( V \) is empty, step 8), the algorithm returns the list \( AV \) of active nodes. The list of active nodes \( AV \) is then made available to the routing layer during the \textit{BcastSlot} used for the backbone creation. In the next section we describe how the forwarding backbone is created and maintained.

IV. FORWARDING BACKBONE

The backbone creation and maintenance algorithm defines the rules to select enough coordinators needed to keep the nodes in the network connected. The task of a coordinator is to forward the packets coming from its neighbors.

A. The coordinator election process

A node volunteers to become a coordinator during the periodic \textit{BcastSlot} (Figure 4) by sending an announcement message. The announcement message also has the list of its two-hop neighbors, specifying which ones are coordinators and which ones are not. This list is used by our scheduling algorithm. All nodes are required to stay awake during the \textit{BeastSlot}. In all other slots, those nodes that are not coordinators follow our scheduling algorithm described in the previous section to determine if to stay active.

If two neighbors of a non-coordinator node cannot reach each other directly or via maximum two coordinator hops, then such a node is elected to become a coordinator. The nodes compete during the \textit{BcastSlot} by carefully timing their announcement for becoming a coordinator. This delay is computed as a function of the residual energy currently available at the node and its \textit{utility}. The \textit{utility}, defined below, is the number of additional pairs of nodes \( C_i \) among the neighbors \( N_i \) that would be connected if node \( i \) became a coordinator.

\[
utility = \frac{C_i}{\binom{N_i}{2}}
\]

Let \( E_r \) denote the amount of energy at a node that still remains and \( E_m \) maximum amount of energy available at the same node. We define the delay of the announcement message as:

\[
delay_i = ((1 - \frac{E_r}{E_m}) + (1 - utility_i)) \times N_i
\]

The delay is normalized to the duration of the slot. As more energy is available at a node, its \textit{utility} is higher, then the delay to sending an announcement message is lower, therefore giving the node a higher chance of becoming a coordinator. While a node waits for the delay to expire, it might receive other announcement messages from its neighbors, possibly changing the knowledge of the status of its neighbors. Therefore, just before broadcasting the announcement message, a node must check if the coordinator eligibility rule still holds. Then the header of the announce message is filled out with the information regarding the nodes neighbors with their status and the status of the node itself. Finally, the announcement message is broadcasted. At the end of the \textit{BeastSlot}, those nodes that are coordinators stay awake until the next \textit{BeastSlot}. Finally, the information about the nodes that are coordinators is made available to the routing layer. Collisions in the \textit{BeastSlot} may result in additional elected

---

**Algorithm 1** Pseudo-code of the scheduling algorithm

Given the two-hop distance subnetwork \( G_i = (E_i, V_i) \), node \( v_i \):

1. Assign \( S \) tickets to each node in the subnetwork:
   \( tk(v_j) \leftarrow S \) for \( \forall v_j \in V_j \)
2. Generate pseudo-random numbers for each node in the subnetwork:
   \( m_j = \text{rand}(id_j + \text{slotno}), \forall v_j \in V_i \)
3. Add the nodes into a set of unchecked nodes:
   \( V' \leftarrow V \)
4. Pick the node \( v_j \) from \( V' \) with the greatest pseudo-random number
5. Determine if \( v_j \) can be scheduled. \( v_j \) is schedulable iff
   \( tk(v_j) \geq 1, \forall v_j \in \{v_j\} \cup (N(v_j) \cap AV) \)
6. If \( v_j \) is schedulable, add it to the assignment of active nodes, and decrease the tickets of \( v_j \) and all its neighbors:
   \( AV \leftarrow AV \cup \{v_j\} \)
   \( tk(v_j) = tk(v_j) - 1, \forall v_j \in \{v_j\} \cup N(v_j) \)
7. Remove \( v_j \) from the unchecked nodes set:
   \( V' \leftarrow V' - \{v_j\} \)
8. if \( (V' \) is empty) then Go to step 4
   else Return the list of scheduled nodes \( AV \).
TABLE I
SIMULATION PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PTx$</td>
<td>Transmission power</td>
<td>1400 mW</td>
</tr>
<tr>
<td>$PRx$</td>
<td>Receive power</td>
<td>1000 mW</td>
</tr>
<tr>
<td>$P_{idle}$</td>
<td>Power in idle state</td>
<td>830 mW</td>
</tr>
<tr>
<td>$P_{sleep}$</td>
<td>Power in sleep state - NIC off</td>
<td>43 mW</td>
</tr>
<tr>
<td>$P_{idle\rightarrow sleep}$</td>
<td>Power during transition from idle state to sleep state</td>
<td>3 mW</td>
</tr>
<tr>
<td>$P_{sleep\rightarrow idle}$</td>
<td>Power during transition from sleep state to idle state</td>
<td>7 mW</td>
</tr>
<tr>
<td>$T_{idle\rightarrow sleep}$</td>
<td>Time for transition from idle state to sleep state</td>
<td>2 ms</td>
</tr>
<tr>
<td>$T_{sleep\rightarrow idle}$</td>
<td>Time for transition from sleep state to idle state</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

coordinates since a node is assumed to be a non-coordinator unless announced. As our results show, this phenomenon does not compromise large energy savings.

V. RESULTS

Simulations were conducted using the ns-2 network simulator [1] version 2.33. We reproduce a simulation environment very similar to what we observed in the Santa Margherita Ecological Reserve (SMER), a part of HPWREN. The child CH nodes form an intra-SMER multi-hop network to deliver data from the sensors to the HPWREN backbone with 802.11b radios. We simulated a 120-node network in square regions of different sizes: 1250m x 1250m, 1000m x 1000m, 750m x 750m, and 500m x 500m. Twenty nodes send and receive traffic. Each of these nodes, if not otherwise specified, generate CBR traffic to another node, sending 128 byte packets. In our experiments, each sender sends three packets per second, similar to typical sensing scenarios, for a total of 60 Kbps of traffic. To make sure that each CBR flow goes through multiple hops before reaching the destination, 10 source and destination nodes are placed randomly in two 50 meters-wide, full-height strips at opposite sides of the simulation area. The initial position of the remaining 100 nodes is chosen using uniform distribution over the entire simulated region. This setup, and the Tx, Rx and Idle state power values are the same as SPAN algorithm [5], which we compare our backbone creation algorithm to. The rest of the simulation parameters are shown in Table 1. To test our solution, we implemented a greedy geographic forwarding protocol. The routing layer knows which nodes are coordinators and which are currently active. Given this information, a node that has data to send, first attempts to forward the packets to the coordinator that is closest to the destination. If such coordinator doesn’t exist, then it tries to find a forwarder among the scheduled non-coordinator nodes. If a forwarder is not found, then the node drops the packet. In our simulations, the module GOD of ns-2 provides the knowledge of the position of each node.

A. Energy

Energy consumption is strictly related to the number of nodes active in the network. As described in Section 3, the $S$ parameter determines the number of non-coordinator nodes that are active. Figure 5 shows that for each simulation area, as the $S$ parameter increases so does the energy consumption. Less energy is consumed by the dense topologies (Figure 5). For denser networks, fewer coordinators are needed to keep all the nodes connected. Since scheduling limits the number of active nodes in a subnetwork, then the overall number of active nodes decreases as the subnetworks become denser. In summary, the number of coordinators and scheduled nodes increases as the density of the network decreases.

Another factor that affects the energy consumptions (summarized in Figure 5) is the number of hops a data packet needs to traverse from source to destination. The more the number of hops, the more the nodes involved in receiving and transmitting data. These operations are power hungry (Table 1). Predictably, the larger the network area (and so the longer the distance from sources to destinations), the more the hops required. In fact, for topologies of increasing sizes 500m x 500m, 750m x 750m, 1000m x 1000m, and 1250m x 1250m, we recorded an increasing average number of hops that is 2.9, 4.5, 6.4, and 8.2 respectively. In summary, as the network size and the number of hops increases, so does the energy consumption.

Figure 6 shows the effect of slot size (100ms, 200ms, 300ms) on energy consumption. We show only the results for the 1000mx1000m topology, as we got similar results for the other scenarios. As expected, larger slot sizes lead to bigger power savings. Shorter slots cause more frequent state transitions and thus greater energy consumption.

We finally compare our results with SPAN [5]. For the case of the 1000mx1000m topology, the energy savings shown in [5] (page 9 Figure 8) are 50%. These are the energy...
savings compared to the situation where all the nodes are active (no nodes are in PSM) and can participate in packet forwarding. For the same topology and $S = 2$, our solution achieves similar savings, 53.9%. Low values of $S$ maximize the energy savings; we evaluate the impact on latency and compare with SPAN in the next section. The main difference with SPAN is that it relies on a set of modifications to the 802.11 PSM in order to reduce power. In contrast, our solution, while achieving similar energy savings, doesn’t require any modifications of the MAC layer. In this way we achieve easy deployability and low cost since legacy devices can be used and expensive MAC modifications can be avoided.

B. Latency

Figure 7 shows the results on average packet delay for different topologies. We find that in general the delay is higher for larger topologies. This is expected because as the size of the network increases, the average number of hops a packet must go through to reach the destination also increases. In the worst case of the largest topology, packet loss is below 4% and decreases as the density of nodes increases.

The impact of the $S$ parameter on the delay is more evident on the largest simulation area, 1250mx1250m. As the density of the nodes increases and the average number of hops to reach the destination increases, the scheduled nodes become very useful in helping the coordinators to forward the packets. Compared to SPAN [5], we achieve better latency results. From [5] we see that in the case of the 1000mx1000m topology, SPAN’s average packet latency is 40.5 ms with 6.1 average hops. With the $S$ parameter set to 2 (number for which we obtain the same energy savings as SPAN), our solution decreases latency by 20.7% with 6.4 average number of hops. This is the result of two main factors. First, by reducing the number of active nodes in the network, our scheduling algorithm reduces contention, thus reducing the MAC layer delay [2][10]. Secondly, those packets that cannot be routed by a coordinator go through one or more scheduled nodes as shown in Figure 8. A scheduled node, compared to a node in PSM as in SPAN case, is capable of much faster forwarding.

VI. CONCLUSION

We presented an energy efficient mechanism for scheduling and routing in heterogeneous wireless sensor networks such as the HPWREN. Our distributed scheduling algorithm allows a large portion of nodes to switch off the NIC thus saving energy. Scheduling is combined with the creation of a backbone of nodes in charge for delivering data to the proper destinations. By requiring no modifications of the MAC layer, our solution can be easily and quickly deployed on existing networks such as the HPWREN where legacy devices don’t need to be replaced nor firmware or drivers modified. Saving energy also lowers the cost of network maintenance by avoiding frequent and expensive replacement of batteries. Compared to SPAN [5], we achieve as good power savings (up to 53%) while delivering packets with 20% lower delay.

ACKNOWLEDGMENT

This work has been supported by HPWREN project, NSF award numbers 0087344, 0426879, 0932403 and 0729029, CNS, Sun Microsystems, UC Micro and Qualcomm.

REFERENCES

[10] Regini, E., Lim, D. and Rosing, T. S. Scheduling Above MAC to Maximize battery Lifetime and Throughput in WLANs, IASTED 2008