*Journal of
Low Power Electronics*
Vol. 7, 1–18, 2011

Resource Management in Heterogeneous Wireless Sensor Networks

Edoardo Regini, Daeseob Lim, and Tajana Šimunić Rosing*

Department of CSE, University of California, San Diego, 9500 Gillman Dr. MS 0404, La Jolla, CA 92093

(Received: 5 October 2010; Accepted: 25 February 2011)

Heterogeneous wireless sensor networks such as High Performance Wireless Research and Education Network (HPWREN) have environmental sensors located in remote and hard-to-reach locations far from the main high-bandwidth data links. 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. Minimizing energy consumption is critical to maintaining operational data links. This paper presents a solution that includes scheduling and routing algorithms and achieves up to 60% energy savings per battery operated node with 20% lower latency when compared to existing techniques. Our TDMA based scheduling algorithm limits the number of active nodes and allows a large portion of nodes to sleep thus saving energy. Since the algorithm is completely distributed and hence minimum (at join time) control packet exchange is required, nodes in sleep state can switch off the wireless network interface thus minimizing power consumption. Furthermore, results show that by limiting the number of active nodes, contention in the channel decreases and hence aggregate throughput increases up to 10%. Scheduling is combined with a dynamic creation of a backbone of nodes in charge of providing connectivity to the network and delivering data to the proper destinations. This mechanism sits on top of the unmodified MAC layer so that legacy network devices can be used, and expensive hardware/software modifications are avoided.

Keywords: Low-Power, Scheduling, Routing.

1. INTRODUCTION

Heterogeneous wireless sensor networks consist of nodes with different communication, computation and power capabilities. They typically include a large number of resource-constrained sensor nodes used for data measurements and fewer resource-rich wireless devices that can be used for data gathering, analysis, and data relaying. Similar to typical wireless networks, heterogeneous wireless sensor networks suffer from a limited battery lifetime, excessive contention between wireless nodes, and insufficient network throughput capacities. In addition, heterogeneous WSN applications have a variety of quality of service (QoS) requirements depending on the nature of the application itself.

The differences among nodes typically lead to a multilevel hierarchical organization of the network in which nodes with increasing capabilities are layered on top of sensor networks. To make the sensed data available/accessible from remote locations, traffic from the sensor networks are gathered at cluster heads. A wireless network of these nodes delivers the data to a backbone of routers with high-bandwidth connections that provide long distance coverage. Therefore, this hierarchical configuration provides wide coverage in the field and successful transfer of data traffic coming from the sensors.

For the purpose of our work we examine the case of a heterogeneous wireless sensor network called HPWREN: High Performance Wireless Research and Educational Network.[10] HPWREN is a collaborative cyber infrastructure for environmental research, education, and first responder activities deployed in southern California (covering nearly 20,000 square miles). Figure 1 shows only the fast wireless backbone links in HPWREN. 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

*Author to whom correspondence should be addressed.
Email: tajana@ucsd.edu

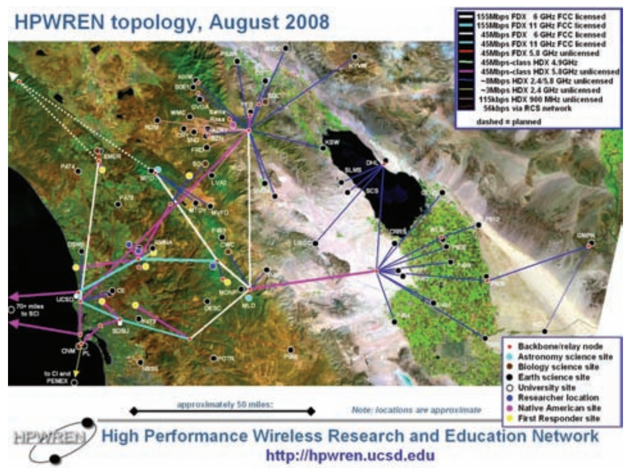
HPWREN topology, August 2008
155Mbps FDX 6 GHz FCC licensed
155Mbps FDX 11 GHz FCC licensed
45Mbps FDX 6 GHz FCC licensed
45Mbps FDX 11 GHz FCC licensed
45Mbps FDX 5.8 GHz unlicensed
45Mbps-class HDX 4.9GHz
45Mbps-class HDX 5.8GHz unlicensed
~8Mbps HDX 2.4/5.8 GHz unlicensed
~3Mbps HDX 2.4 GHz unlicensed
115kbps HDX 900 MHz unlicensed
56kbps via RCS network
dashed = planned
Backbone/relay node
Astronomy science site
Biology science site
Earth science site
University site
Researcher location
Native American site
First Responder site
approximately 50 miles
Note: locations are approximate
High Performance Wireless Research and Education Network
http://hpwren.ucsd.edu

Fig. 1. FGS-based layer structure.

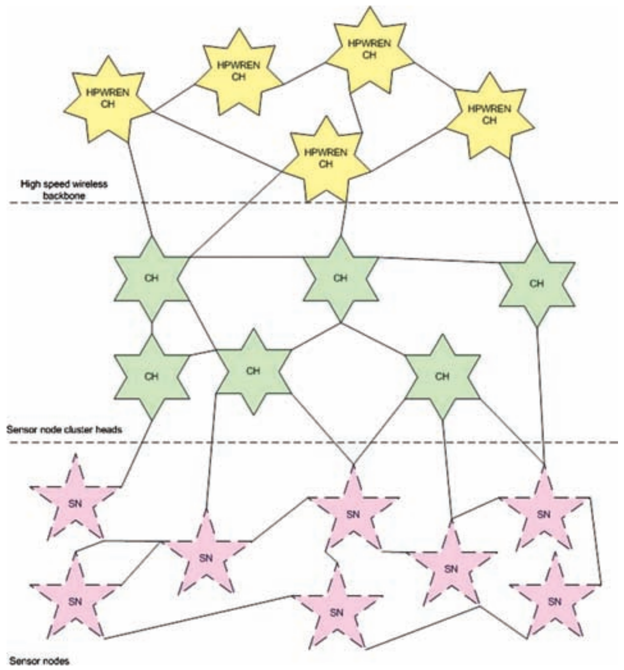

Fig. 2. HPWREN: three layer structure.

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 the 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.

A heterogeneous wireless sensor network such as HPWREN can be described with a three-layer structure shown in Figure 2. The top layer represents the wireless mesh backbone of HPWREN shown in Figure 1. 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 are accessible to line power. In this layer, policy based routing and a level of QoS is provided by the routers. The bottom layer contains sensors. HPWREN offers a large variety of sensors with different characteristics and resource requirements that span, for instance, from low bandwidth but tight real-time traffic deadlines of seismic sensor nodes, to long battery lifetime requirements of small and remotely deployed weather stations. At this layer a big issue is the battery lifetime since sensors are typically small devices with very strict power constraints. 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 2 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 also frequently 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 their battery lifetime to ensure timely delivery of the sensor data. For example, the 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.

Commercial wireless LAN (WLAN) such as IEEE 802.11 is a good candidate for the relay network which connects the cluster head nodes with the routers. 802.11 Wireless LANs (WLANs) are today widely used because of their convenience, cost efficiency and easy deployability. We study the case where WLAN technology is employed in large-scale wireless networks such as HPWREN where nodes are connected in ad hoc mode. Given the lack of a network infrastructure, when a source and a destination node that are far away from each other want to communicate, data is required to be routed through multiple nodes in the network in order to be delivered. Routing of packets is made challenging both by the lack of a network infrastructure and because of limited resources available at the nodes such as energy. Ensuring that the node battery lifetime is long enough for data collection and delivery to happen in a timely manner is of critical importance. Typically, the wireless communication device consumes a large portion of the total energy. In particular the radio power consumption in idle state plays a significant role in the battery lifetime. In the case of a congested network, the issue of communication power is also more important since contention causes many packet collisions and nodes have to perform several packet retransmissions.

This results in severe energy consumption and lower network throughput. Furthermore, because some of the applications have data that urgently need to be delivered (e.g., first responders in HPWREN), there needs to be a method to trade off energy with QoS. The solution described in this paper is designed to address these issues.

The main contributions 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. We focus on optimizing data delivery via wireless network interfaces (NIC) since that accounts for a significant fraction of the overall energy consumption. In contrast to previous work,[3] our algorithm does not require any changes to the MAC and provides better performance in terms of latency. Figure 3 shows the two main components of our solution: the scheduler and the backbone creation and maintenance algorithm. Next we describe these two components and how they combine in a unique solution.

The scheduler uses a TDMA based, ditributed algorithm to limit the number of active nodes in the network. It allows a large portion of nodes to switch off the NIC and save power. In the example in Figure 4, during a generic slot of the TDMA scheme, the *Scheduled-ON* nodes are selected by the distributed scheduler to be active, while the *Sleep-OFF* nodes save power by switching into a sleep state. The scheduling algorithm takes as input a parameter we call *S*. The *S* parameter affects the number of nodes scheduled in a neighborhood of nodes. In fact, at the beginning of each slot, a node assigns exactly *S* tickets to each of its neighbors in two-hop distance. It then generates a sequence of pseudo random numbers for each of them using the unique ID of the nodes as seed. The idea behind this mechanism is that by knowing the ID of their neighbors, different nodes can generate the same sequences of numbers associated with a node. The nodes in the neighborhood then start a competition in which the numbers generated along with the neighbor relation among the nodes determine the way the tickets are decreased. Those nodes that have at least one ticket left at the end of the competition know they are scheduled and thus can stay active for the duration of the current slots. The other nodes switch off their NIC. This competition repeats at the

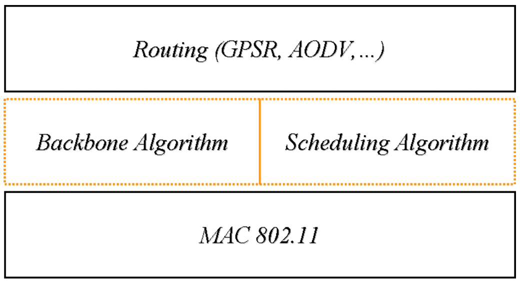

Fig. 3. High level overview of the proposed solution.

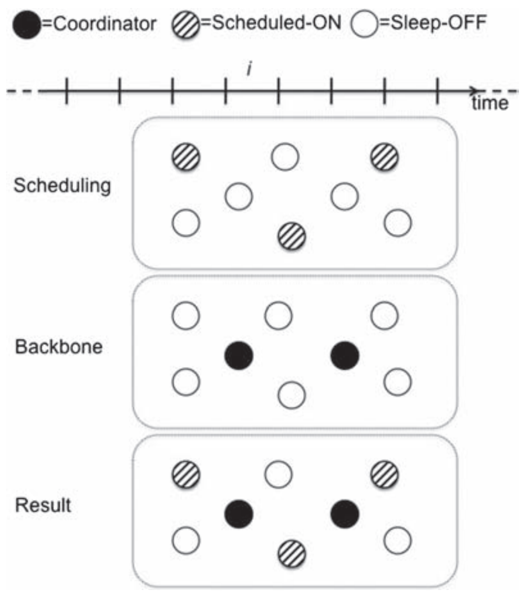

Fig. 4. Overview of the proposed solution.

beginning of each slot. Randomization gives the nodes the same opportunity to be scheduled. If nodes need higher priority, then traffic prioritization at node level is applied using techniques such as Weighted Fair Queuing (WFQ).[6] The algorithm behind the scheduler is described in more details in Section 3.

The backbone algorithm creates a dynamically changing network of a subset of the remaining inactive nodes for data forwarding. In the example in Figure 4, two of the nodes that are not scheduled become part of the backbone (the *Coordinator* nodes). The nodes of the backbone (called *coordinators*) are selected periodically in special communication slots (called *BcastSlot*s). The selection of the coordinators is based on the nodes' remaining battery lifetime and their *utility*. The *utility* is a measure of how many pairs of neighbors the node would connect if it becomes part of the backbone. Nodes volunteer to become coordinators through an announcement message. At the beginning of the *BcastSlot* each node sets up a delay that depends on its remaining battery and *utility*. The nodes that announce itself first become coordinators and are required to stay active until the next *BcastSlot* where the announcement process starts over. Therefore, the delay of the announcement is the key of the backbone creation mechanism. The more battery energy remaining at a node, and the more its *utility*, the less the delay of its announcement is and it thus has a higher probability of becoming a *coordinator*. This mechanism also ensures that the *coordinators* change dynamically as the batteries of the nodes drain. Being a coordinator is an energetically expensive task because coordinators are required to stay active and forward the data coming from the neighbor nodes. The backbone algorithm makes sure that those nodes with more energy in their battery are selected to become part of the backbone. More details about the backbone creation and maintenance algorithm are given in Section 4.

Finally, the information about the active nodes scheduled by the scheduler and the *coordinators* selected by the backbone algorithm are merged as shown in Figure 4 (*Result*). *Result* specifies for each node whether it is active/sleeping and which ones act as *coordinators*. This information is then made available to the routing layer that is now aware of the status of the nodes and can make the proper routing decisions.

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. We implement a greedy geographic algorithm in which a node first attempts to forward a packet to a coordinator that is closest to the destination to test our ideas. If such a coordinator does not 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 solution presented in this paper introduces a new approach in achieving energy savings while maintaining performance in wireless networks. Our low-power scheduling algorithm was designed to be free from any assumption on the network topology and is suitable for multi-hop routing scenarios. Its combination with the forwarding backbone mechanism represents a novel approach based on a unique TDMA scheme where nodes decide to go to sleep in a distributed fashion and synchronize during specific slots to maintain the network connected. In the next section, we outline related work and highlight the differences with our solution.

2. 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.

One approach to improve performance in terms of throughput and/or energy consumption is to revise the MAC layer algorithms. For example PAMAS[5] powers off the wireless NIC during transmission of packets not addressed to the node. Enhanced DCF (EDCF)[16] prioritizes traffic categories with different contention parameters. In contrast the original DCF algorithm cannot give prioritized service to the user. According to the priorities, a wireless node can implement up to eight transmission queues. Each transmission queue has different parameters that decide its backoff time. With this, EDCF gives more chances of channel access to high priority traffic. EDCF is compatible with legacy DCF while providing a differentiated service. Distributed Fair Scheduling (DFS) in Ref. [24] differentiates the backoff interval (BI) according to the packet length and traffic class. As the node with the smallest BI transmits packets first, DFS enables the service differentiation by adjusting BI values. The Opportunistic Auto Rate (OAR) protocol in Ref. [19] exploits the automatically adjusted transmission rate of 802.11. The OAR protocol sends multiple back-to-back data packets when the channel quality is good. To enable this, it changes the information fields in RTS/CTS packets to send more data packets in a reserved transmission time slot. All the algorithms in this category require modifying the existing DCF mechanism or the packet headers. Therefore, it is difficult and expensive to apply the above algorithms to the existing network devices. Generally, altering the MAC layer 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 the MAC layer, which we summarize next.

Overlay MAC layer (OML)[23] adds an additional conceptual layer over the existing 802.11 MAC, thus enabling the use of COTS. OML uses loosely synchronized time clocks to divide the time into 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 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[1] is a rate control mechanism for TCP and UDP traffic which works on the best-effort MAC. SWAN provides service differentiation and sender-based admission control. The strength of the SWAN's model is that it is a distributed mechanism and works with feedback from the network. In fact, it collects the feedback information from the MAC layer or from other network nodes. SWAN improves throughput and achieves good fairness among different types of traffic. However, it does not consider the energy consumption of the wireless nodes. Both Overlay MAC and SWAN assume that the nodes in the network continuously listen to a channel. Reducing energy consumption of nodes is not considered in their work. Combining scheduling and power management is presented in Ref. [8]. The algorithm performs a distributed transmit power control while scheduling wireless nodes in order to eliminate strong interference. However, this mechanism requires a separate contention-free feedback channel for sending information about radio conditions. Furthermore, the scheduling algorithm depends on a central controller node. The TDMA based protocol in Ref. [13] gives a simpler control scheme; a server periodically broadcasts a control packetwhich contains the scheduling information of each client station. A client awakes at a predetermined time to transmit a series of data packets after which it transitions into a power-save mode. Since only one station is activated at any given time, it can complete transmissions during the short interval and stay in its power-save mode for a long time. In contrast, we use a distributed mechanism in which nodes can decide when to transit into a sleep state without exchanging control packets with a central server node or its neighbors. Distributed scheduling is an important characteristic of the scheduling algorithm used in our solution. In fact, centralized scheduling is vulnerable to the failure of the single control node. In previous work, running distributed scheduling algorithms on wireless networks has been discussed.[17, 18, 25] Although these algorithms work in a distributed way, they assume a special framing of radio channels,[17] a separate radio channel,[25] or the traversal of the entire network using a special token.[18]

Our scheduling algorithm is based on the ideas presented in Refs. [14 and 7]. Scheduling is used as a mechanism to save power and to reduce contention. In contrast to Refs. [7, 14], in this work we adapt scheduling to a multi-hop context. In fact, the node-level scheduling algorithm in Refs. [7, 14] is used in networks where nodes are grouped in well defined cells where each node is in one-hop distance with its Base Station (the destination). Instead, we don't make any assumption on the network topology, and source and destination nodes can be at any arbitrary distance (number of hops away from each other). To achieve this, the scheduling algorithm synchronizes with a novel dynamic backbone creation algorithm that enables multi-hop routing of data in an energy efficient and timely manner.

SPAN[3] 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 *BcastSlots* during which the forwarding backbone is created as described in Section 4. Furthermore, SPAN relies on the 802.11 Power Saving Mode (PSM)[2] 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 do not 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 Ref. [22] several power aware routing metrics that increase the lifetime of the nodes and the network are described. Geographic based routing is described in Ref. [9]. In geographic based routing 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 by the current node, candidate node, and the destination. The next hop that is chosen has a higher probability of being 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 and show its efficiency through simulations and measurements. Chapter 4 discusses how to couple our scheduler with a novel dynamic backbone creation and maintenance algorithm. The performance of our solution is evaluated through simulations in Section 5; finally we conclude in Chapter 6.

3. SCHEDULING ALGORITHM

In this section we describe a distributed scheduling technique that enables nodes to save power by spending a large amount of time in the sleep state. Simulation results in Ref. [14] show that the distributed *node-level* scheduling algorithm, by limiting the number of active nodes in the network, achieves considerable power savings and increases throughput. In order to verify this idea we tested the distributed *node-level* scheduling algorithm described in Ref. [14] in a testbed network composed of eight nodes and took measurements on power consumed, packet delay and throughput. In addition we show with our measurements how packet delay can be handled by using node level packet prioritization. In this section we describe the scheduling algorithm and compare the values we measured with the simulation results in Ref. [14]. The purpose of the measurements shown in this section is to verify the effectiveness of the scheduling algorithm that, when combined with the forwarding backbone mechanism presented in Section 4, represents the unique solution that is the subject of this paper.

3.1. The Scheduling Algorithm

When employed in large-scale wireless networks, the 802.11 protocol can see significant reductions in throughput due to contention and interference. Figure 5 shows the results of a simulation that calculate the aggregate throughput while nodes are transmitting with full MAC queues. Simulations where conducted with the *ns-2* simulator[21] version 2.28 with typical 802.11 MAC/PHY settings[14] and RTS/CTS disabled. In this simulation, there is one base station (access point) and several wireless nodes around it. Wireless nodes generate UDP data traffic and send it to the base station. The total amount of generated traffic is set higher than the aggregate throughput, so that MAC layer queues are always full. We change the number of transmitting nodes and measure the aggregate throughput at the base station. We observe that as the number of wireless nodes transmitting data at the same time increases, the throughput falls. This is because in heavy traffic conditions, the chance of nodes to successfully transmit a packet decreases dramatically as nodes spend a lot of time waiting for the channel to become idle.[15] Intuitively, from Figure 5 it is clear that we can get a higher throughput by limiting contention to only a few nodes at a time.

The scheduling algorithm is a TDMA solution that gives opportunities to the nodes to save energy by powering off their 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 margin of error of up to a few milliseconds. Therefore, we can use any lightweight timer synchronization. For instance, the 802.11b TSF[2] is such a synchronization mechanism.

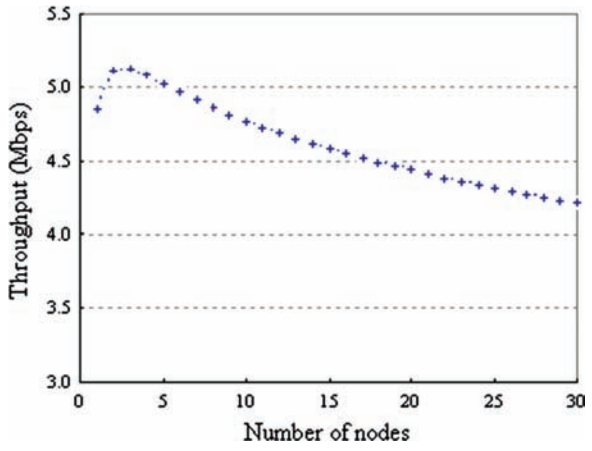

Fig. 5. Throughput drop due to nodes transmitting at full MAC queues.

To describe the scheduling algorithm, 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:

$$AV = \bigcup_{v_i \in V} AV(v_i) \tag{1}$$

Then the basic constraint in the scheduling problem is that the number of neighboring active nodes should not exceed the given parameter S:

$$|AV(vi)| \leq S, \quad \forall vi \in V \tag{2}$$

where $|AV(v_i)|$ is the number of nodes in $AV(v_i)$.

An assignment that maximizes the size of AV is called *maximum assignment*. In terms of the number of active nodes in a given network, maximum assignment is optimal, but it has been shown in Refs. [28 and 29] that the maximum assignment problem for $S = 1$ is *NP*-complete. The demonstration of *NP*-Completeness for $S \geq 1$ is given in the next subsection (Section 3.1.1). Thus, the scheduler implements maximal assignment: nodes are scheduled so that the number of active nodes is maximal; it means that no additional assignment of an active node can meet the constraint in Eq. (2). The maximal assignment problem can be solved in polynomial time for a given network graph $G(V, E)$. From Ref. [29] it is possible to extract an algorithm for maximal assignment for $S = 1$ when the knowledge of the whole network is known. Instead we extend this idea for $S \geq 1$ and restrict the assumption of knowing the whole network topology. Intuitively, a node can decide to be scheduled if the sum of active nodes in its neighborhood is lower than the constant S. Then, a node does not need to know the whole network topology. In fact, a node running the scheduling algorithm needs only the knowledge of its two-hop neighbors (the nodes in one- and two-hop distance). We call this partial network topology a *subnetwork*.

The size of the *subnetwork* is an important factor. Maintaining topology information of a whole network at every node results in a high overhead at runtime whenever a new node joins/leaves the network or when, for any reason, it fails. The larger the size of the network, the more expensive it is to propagate the information of a network topology change to all the nodes. The subnetwork for node v_i, is defined as the subnetwork graph $G_i = (E_i, V_i)$ which is a subset of $G = (V, E)$. All vertices and edges within two hops distance from v_i are added to the subnetwork $G_i = (E_i, V_i)$. An example of building a subnetwork is given in Figures 6(a, b). Figure 6(a) shows a subset of nodes in a

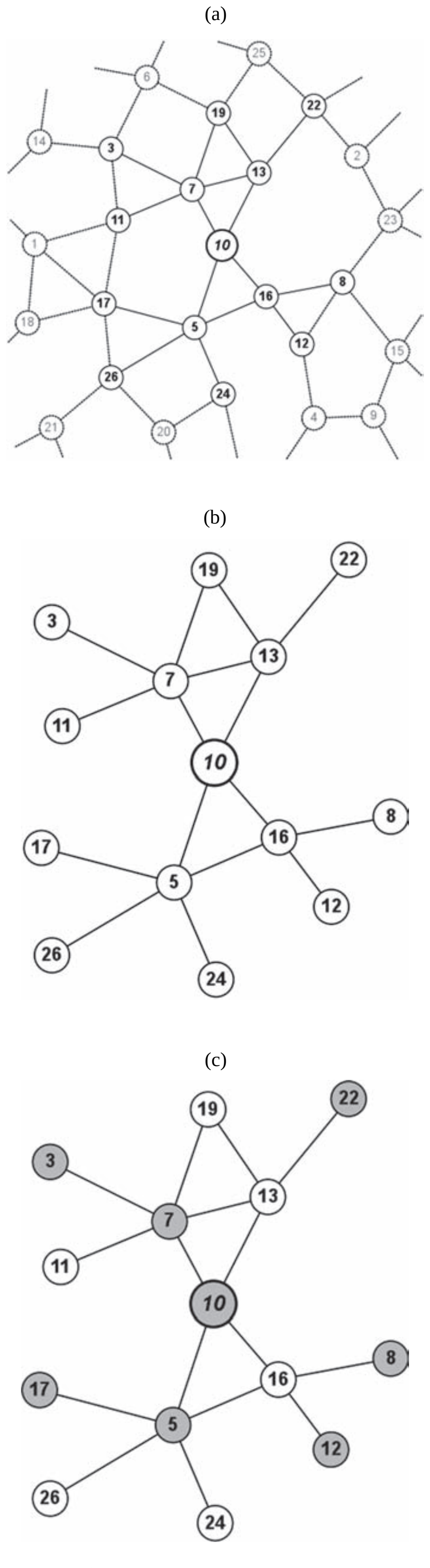

Fig. 6. Example of running the scheduling algorithm on node v_{10}: (a) Node connectivity nearby n_{10}. (b) Two-hop subnetwork for n_{10}. (c) Scheduled nodes.

generic network graph $G = (V, E)$ and their connectivity. Figure 6(b) shows the subnetwork $G_{10} = (E_{10}, V_{10})$.

Algorithm 1 (Pseudo-code of the scheduling algorithm) Give the two-hop distance subnetwork $G_i = (E_i, V_i)$, node v_i:

```
1: Assign $S$ tickets to each node in the subnetwork:
   $tk(v_j) \longleftarrow S$ for $\forall v_j \in V_j$
2: Generate pseudo-random numbers for each node in the subnetwork:
   $rn_j = rand(id_j + slotno),\ \forall v_j \in V_i$
3: Add the nodes into a set of unchecked nodes:
   $V' \longleftarrow V_i$
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
   $tk(v_j) \geq 1,\ \forall v_j \in \{v_i\} \bigcup (N(v_i) \bigcap AV)$
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:
   $AV \longleftarrow AV \bigcup \{v_j\}$,
   $tk(v_j) = tk(v_j) - 1,\ \forall v_j \in \{v_j\} \bigcup N(v_j)$.
7: Remove $v_j$ from the unchecked nodes set:
   $V' \longleftarrow V' - \{v_j\}$
8: if ($V' \neq$ empty) then Go to step 4
   else Return the list of scheduled nodes $AV$.
```

At the beginning of a slot, each node in the network runs the distributed scheduling algorithm. We next describe its steps and give an example of a run of the algorithm at node v_{10} given the subnetwork shown in Figure 6(b). The pseudo-code for the scheduling algorithm is shown 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 the sum of the node IDs and the current sequential slot number as the seed. 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. In step 3, it creates a set of unchecked nodes V' that include all nodes in V_i. In step 4 the algorithm extracts 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, are 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 all of 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 case, 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.

An example of the scheduling algorithm with $S = 3$ running on node v_{10} is shown in Figure 6(c). First, we find v_{10}'s two-hops subnetwork $G_{10} = (E_{10}, V_{10})$ depicted in Figure 6(b). Let's suppose that the node with the lower ID has the higher pseudo-random number value. The scheduling algorithm first schedules v_3, the node with the highest random number. After v_3 is scheduled, the number of tickets of nodes who are neighbors of v_3, v_{14}, v_{11}, v_7, v_6, and v_3 itself are decreased by 1. As a result, we get $tk(v_3) = tk(v_{14}) = tk(v_{11}) = tk(v_7) = tk(v_6) = 2$. In the same way v_5, v_7 and v_8 are scheduled. At this point v_{10} has the lowest random number. Since $tk(v_{10}) = 1$ and the number of tickets of its active neighbors v_7 and v_5 are greater than 0 ($tk(v_5) = tk(v_7) = 1$), node v_{10} is scheduled. The result from running the scheduling algorithm over the subnetwork in Figure 6(b) is shown in Figure 6(c).

3.1.1. Np-Completness of the Maximum Assignment for $\mathbf{S \geq 1}$

Because we consider the scheduling problem for $S > 1$, it is important to show that this problem is NP-complete. If it is NP-complete, maximum scheduling cannot be scalable over a large-size network. It has been reported in Ref. [29] that the decision problem for $S = 1$ is equivalent to the maximum clique problem, which is NP-complete. To show that the maximum assignment problem for $S > 1$ is NP-complete, we extend the idea for $S = 1$ (Ref. [29]) into the case for $S > 1$.

In a given network $G = (V, E)$, we define a corresponding graph $G' = (V', E')$ as the following. First, let $N(v_i)$ be a set of neighbor nodes of a node $v_i \in V$. Define $d(v_i)$ as the degree of a node v_i which is the number of edges at v_i. This means $d(v_i)$ is also the number of neighbor nodes of v_i. In other words, $d(v_i)$ is equal to $|N(v_i)|$. For each node $v_i \in V$, add corresponding subnodes, which belong to corresponding graph G', $v_{i,1}, v_{i,2}, \ldots, v_{i,S_{s-1}(v_i)}$ to V', where $S_{s-1}(v_i)$ is the number of subnodes in V' for a node $v_i \in V$. A subnode in V' represents one of the cases where some of v_i's neighbor nodes are scheduled together with v_i. We explain it below in detail.

$S_{s-1}(v_i)$, the number of subnodes in V' corresponding to v_i in V, is the number of ways to choose $s-1$ nodes from $N(v_i)$. For example, when $N(v_1) = \{v_2, v_3, v_4, v_5\}$ and $s = 3$, $S_2(v_1)$ is $\binom{|N(v_1)|}{3-1} = \binom{4}{2} = 6$. In other words, $S_{s-1}(v_i) = \binom{d(v_i)}{s-1}$. However, in the case of $d(v_i) < s-1$, v_i and its all neighbor nodes can be scheduled together without violating Constraint (2). In that case, $S_{s-1}(v_i)$ is 1. To summarize:

Lemma 3.1. *$S_{s-1}(v_i)$ is $\binom{d(v_i)}{s-1}$ if $d(v_i) \geq s-1$, otherwise 1.*

Now, we connect the subnodes in G'. For a subnode $v_{i,u}$ where $1 \leq u \leq S_{s-1}(v_i)$, we define $N'_{s-1}(v_{i,u})$ as a subset of $N(v_i)$ with the size of at most $s-1$. If $u \neq w$, then $N'_{s-1}(v_{i,u}) \neq N'_{s-1}(v_{i,w})$. In other words, $N'_{s-1}(v_{i,u})$ is a set of neighbor nodes of $v_{i,u}$ in G'. Also, $N'_{s-1}(v_{i,u})$ represents neighboring nodes that are scheduled together with v_i. When $N(v_1) = \{v_4, v_5, v_6\}$ and $s = 3$, for example, $N'_2(v_{1,1}) = \{v_4, v_5\}$, $N'_2(v_{1,2}) = \{v_4, v_6\}$, and $N'_2(v_{1,3}) = \{v_5, v_6\}$. This tells us that we have three choices in scheduling v_1 and its neighbors; we can schedule $\{v_1, v_4, v_5\}$, $\{v_1, v_4, v_6\}$, or $\{v_1, v_5, v_6\}$.

In the above, $v_4 \in N'_1(v_{1,1})$ does not mean $(v_{1,1}, v_4) \in E'$, because $v_4 \in V$ but $v_{1,1} \in V'$. Rather, it means that $v_{1,1}$ is linked with a subset of subnodes of v_4. The actual connectivity between subnodes in V' is explained next. We describe the definition of edges in two steps.
(2) For $(vi, vj) \in E, i \neq j$ in G: if $(vi, vj) \in E$ in G, we add edges between subnodes $v_{i,u}$ and $v_{j,w}$ according to the following rule:

Rule 3.1. *For $\forall (v_i, v_j) \in E$ and $i \neq j, E' \leftarrow E' \cup \{(v_{i,u}, v_{j,w})\}$ iff $v_j \in N'_{s-1}(v_{i,u})$ and $v_i \in N'_{s-1}(v_{j,w})$ for all u, w.*

2 For $(vi, vj) \notin E, i \neq j$ in $G : (vi, vj) \notin E$ means that v_i and v_j are not neighbor of each other. In this case all subnodes of v_i can have links to subnodes of v_j. We can do this conversion by adding *Rule 3.2* to the definition of $N'_{S-1}(v_{i,u})$. Subnodes of v_i are then linked to subnodes of v_j by *Rule 3.1*.

Rule 3.2. *For $\forall (v_i, v_j) \in E$ and $i \neq j, N'_{s-1}(v_{i,u}) \leftarrow N'_{s-1}(v_{i,u}) \cup \{v_j\}$ and $N'_{s-1}(v_{j,w}) \leftarrow N'_{s-1}(v_{j,w}) \cup \{v_i\}$ for all u, w.*

Rule 3.2 means that any node v_j in G is added to $N'_{S-1}(v_{i,u})$ if v_j is neither v_i itself nor a neighbor node of v_i in G. Let the maximum clique found in G' be $MC(G') = (MV, ME)$. Then, we can derive the following from $MC(G')$:

Lemma 3.2. *If $v_{i,u} \in MV$, then $v_{i,w} \notin MV$ for $\forall u \neq w$*

Let's suppose that $vi, u \in MV$ and $vi, w \in MV$ for $u \neq w$. Then, there must be $(vi, u, vi, w) \in E'$, because $MC(G')$ is a maximum clique in G'. In a clique, there exist edges between all pairs of nodes. However, by Rules 3.1 and 3.2, there cannot be any edges between subnodes derived from the same node. Therefore, $MC(G')$ includes at most one subnode for each node $vi \in V$.

Lemma 3.3. *If there is a schedulable assignment of nodes in G, there is a corresponding clique in G'.*

When a given assignment of nodes is schedulable, it means $|AV(v_i)| \leq s$ for $\forall v_i \in AV$. It is obvious that $|AV(v_i) - \{v_i\}| < s$ at any active node $v_i \in AV$. By the definition of $N'_{k-1}(v_{i,u})$, there must be at least one subnode $v_{i,u}$ satisfying $AV(v_i) - \{v_i\} \subset N'_{k-1}(v_{i,u})$. It is because subnodes of v_i cover all combinations of having $s-1$ neighbor nodes of v_i's neighbors.

Given above, when two active nodes v_i and v_j satisfy $v_j \in N(v_i)$, it is always true that there exists a subnode $v_{i,u}$ of v_i and a subnode $v_{j,w}$ of v_j satisfying $(v_{i,u}, v_{j,w}) \in E'$. For v_i and v_j, if $v_j \notin N(v_i)$, then every subnode of v_i has edges with every subnode of v_j in G' by *Rule 3.2*. Therefore, whenever a schedulable assignment of nodes is given in G, there exists a corresponding clique in G'.

LEMMA 3.4. *If there is a clique in G', there is a corresponding clique in G.*

By Lemma 3.2, the clique has at most one subnode for each $v_i \in V$. By the definition of Rule 3.1 and of a maximum clique, a subnode $v_{i,u} \in MV$ has at most $s-1$ edges in $MC(G')$ with the subnodes of $N(v_i)$. So, any node $v_i \in V$ corresponding to $v_{i,u} \in MV$ does not violate Constraint (2). The assignment of nodes corresponding to $MC(G')$ is just to schedule the nodes corresponding to MV; for example, a schedule v_i if $v_{i,u} \in MV$ for any subnode $v_{i,u}$ of v_i. Obviously, the conversion from $MC(G')$ into the assignment in G takes $O(|V|)$ time.

LEMMA 3.5. *Conversion of G into G' takes polynomial time.*

Finding $N(v_i)$ for all $v_i \in V$ takes $O(|V| \cdot |E|)$ time. Rule 3.1 takes $O(|V| \cdot |E|)$ time, and Rule 3.2 needs $O(|V| \cdot |E|^2)$ time. Therefore, the conversion from G to G' takes polynomial time.

THEOREM 1. *The maximum scheduling problem is NP-complete.*

By Lemma 3.3 and Lemma 3.4, there is an one-to-one mapping between the schedulable assignments in G and maximum cliques in G'. By Lemma 3.5, the conversion of a given network G to G' takes polynomial time. As shown in Lemma 3.4, it also takes polynomial time to map a maximum clique in G' into the schedulable assignment in G. We know that the maximum clique problem is NP-complete. Therefore, the given scheduling problem is NP-complete.

3.2. Simulation Results

In this section we present a summary of the simulation results of the scheduling algorithm together with a brief analysis and discussion with the goal of comparing them with measurments on a testbed (see Section 3.3). Simulations are run on the *ns2* network simulator using the simulation parameters listed in Table I. Simulation parameters for 802.11b are set for the typical IEEE 802.11b wireless channel.[26,27] The parameters for power in each power mode are from the data sheet of the *Cisco Aironet* wireless LAN adapter[4] and the measurements presented in Ref. [12]. The size of the network is 533 m by 550 m and nodes are deployed with a hexagonal network topology as shown in Figure 7 that approximates the density of the HPWREN SMER subnetwork used to collect data traces. In the next subsection we present the results of scheduling with HPWREN data, followed by analysis of how our scheduler responds to different slot sizes.

Table I. Scheduling: Simulation parameters.

Parameter	Meaning	Value
d_t	Communication range	50 m
d_{CS}	Carrier-sensing range	99 m
P_{idle}	Power of WNIC in idle mode	0.6698 W
P_{TX}	Power of WNIC in transmitting a packet	1.0791 W
P_{sleep}	Power of WNIC in sleep mode	0.0495 W
$P_{idle\_to\_sleep}$	Transition power from idle mode to sleep mode	0.6698 W
$P_{sleep\_to\_idle}$	Transition power from sleep mode to idle mode	0.6698 W
$T_{idle\_to\_sleep}$	Transition time from idle mode to sleep mode	0.4 ms
$T_{sleep\_to\_idle}$	Transition time from sleep mode to idle mode	20 ms
P_t	Transmitted signal power	0.031622777
CP_{thresh}	Collision threshold	10.0
CS_{thresh}	Carrier sense power	3.00923e-10
RX_{thresh}	Receive power threshold	1.179743-9

3.2.1. Results for Data Traffic Collected at HPWREN

The results shown in this section refer to simulations using real data traffic collected at HPWREN where nodes have an average data rate of 136 Kbps. Figures 8 and 9 show a range of scheduling slot sizes, from 0.1 s to 0.5 s. In

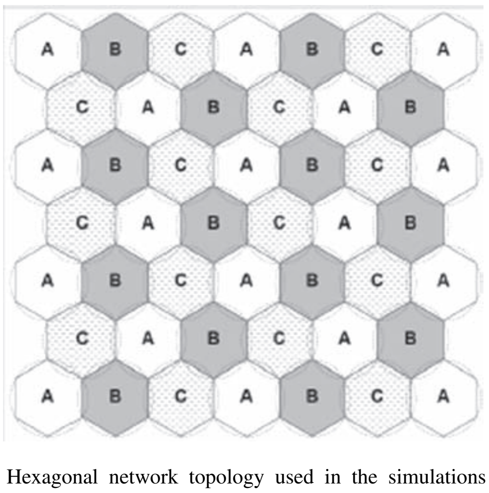

Fig. 7. Hexagonal network topology used in the simulations for the scheduling algorithm.

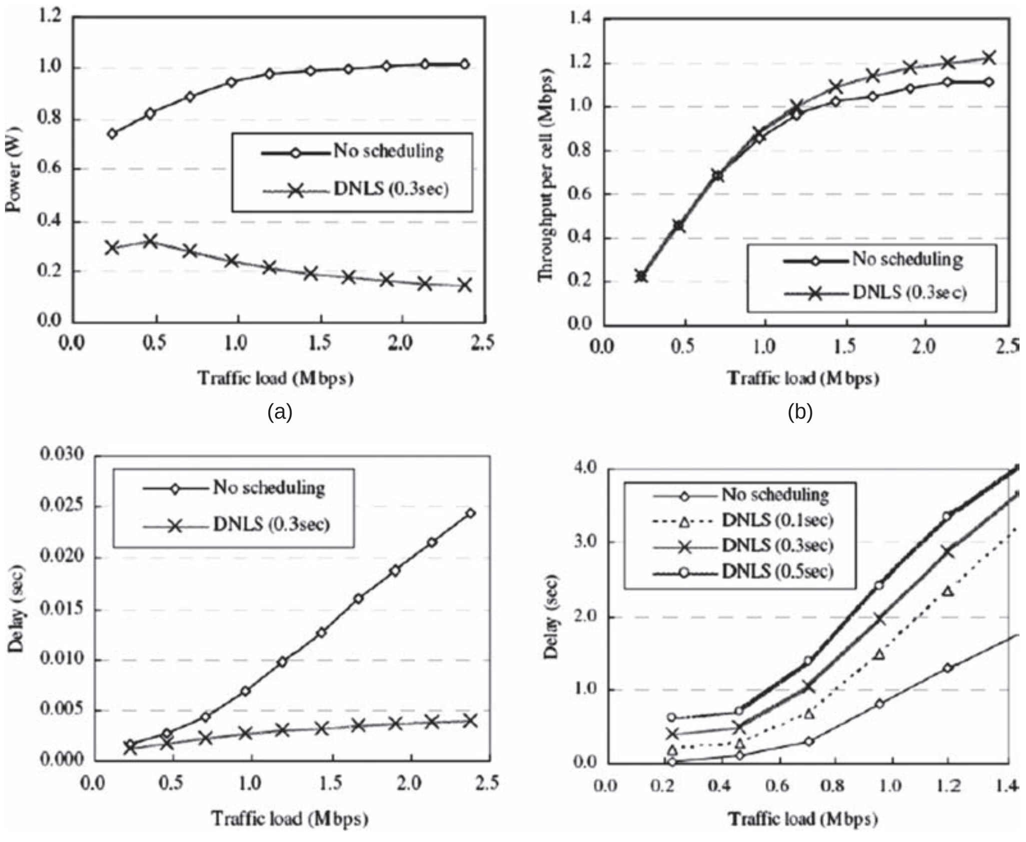

Fig. 8. Node-level scheduling algorithm simulation in hexagonal topology using data from HPWREN. (a) Power. (b) Throughout. (c) MAC delay. (d) Application delay.

Figure 8 we study power consumed (a), throughput (b), MAC delay (c) and application delay (d) as a function of traffic load, while in Figure 9 we present throughput and power as a function of node density. As shown in Figure 8(a), the scheduling algorithm achieves great power savings: up to 85.54% of communication power is saved. Average throughput and MAC layer transmission delay are shown in Figure 8(b and c). We observe that the MAC delay is considerably reduced and throughput is improved by up to 10.31%. The reason for this is that using scheduling on a given network reduces the number of contending nodes in the channel; consequently, the MAC layer delay is reduced. However, because of the TDMA based scheme of the algorithm the application layer delay increases according to the slot size adopted as shown in Figure 8(d). Unscheduled nodes in fact wait in a sleep mode while buffering data from applications. It is unavoidable to experience a certain level of application layer delay in scheduling techniques which use a sleep mode interface device. However, it is possible to efficiently handle the delay by prioritizing traffic, such as with weighted fair queuing[6] as discussed in Section 3.4. Another way to reduce the application layer delay is by minimizing the size of the scheduling slots as shown in the next section.

3.2.2. *Effect of Different Scheduling Slot Sizes*

The results on the average throughput and communication power with different slot sizes are shown in Figure 9. In Figure 9(a), we see that if the scheduling slot is short, nodes switch their modes more frequently. This phenomenon causes more mode transitions. Frequent mode transitions result in the reduction of throughput. The same applies to power consumption. When the wireless interface switches its mode, it consumes at least as much power as it does in the idle mode. Mode transitions also take a certain transition time to wake up and go to sleep. Thus, it is expected that scheduling with the shorter slot size reveals more overhead in energy consumption as shown in Figure 9(b). In order to improve the average throughput and save more communication power it is better to use longer time slots. However, it causes a longer delay as shown in Figure 8(d). This tradeoff is a key issue in determining the scheduling slot size. Experiments such those in Figure 9 can be used to select the most appropriate slot size for a specific network depending on node energy budget and application requirements.

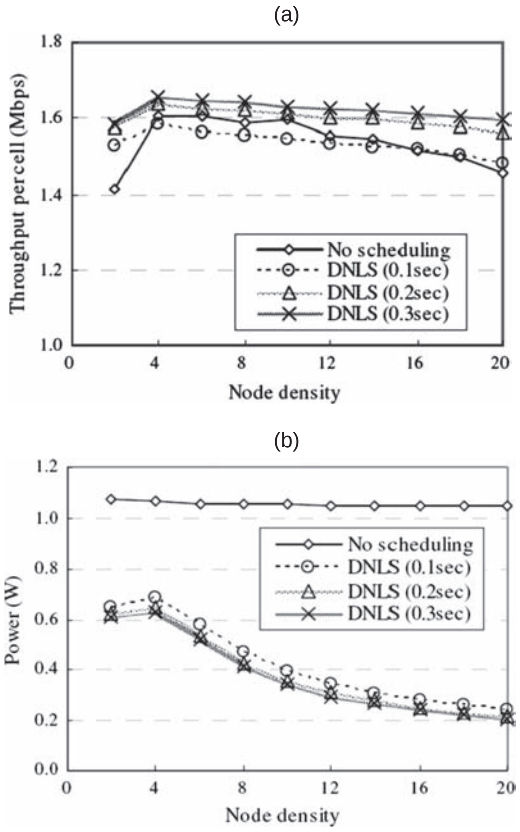

Fig. 9. Node-level scheduler simulation in hexagonal topology with different slot sizes. (a) Power. (b) Throughout.

3.3. Measurements

We evaluate the performance of our scheduling algorithm presented in Section 3 on a measurement setup that well representsa subset of HPWREN network. We present the results of our experiments with respect to throughput and power consumptionin the subsections below and compare them to those obtained in the simulations. We setup a test network consisting of one *parent CH* and 8 *child CHs*. A *parent CH* is a Desktop PC running Linux connected to an Access Point (AP) which is Apple AirPort Base Station. A *child CH* is an *Intel PXA27x* board[11] with a *Cisco Aironet 350 series wireless LANPCMCIA* adapter[4] with similar characteristics to the devices used by *child CHs* in SMER section of HPWREN.

3.3.1. Throughput

Figure 10 shows the specific performance degradation of our test network. In this experiment, each node generates a high rate of CBR traffic in order to keep MAC queues full. The maximum achievable throughput of 5.04 Mbps is obtained when a total of 3 nodes transmit at the same time. As expected this value is slightly lower (1.56%) than the one estimated with the simulations in Section 3.2 because of small interferences still present on campus even during off hours. Figure 10 also shows the throughput bene-

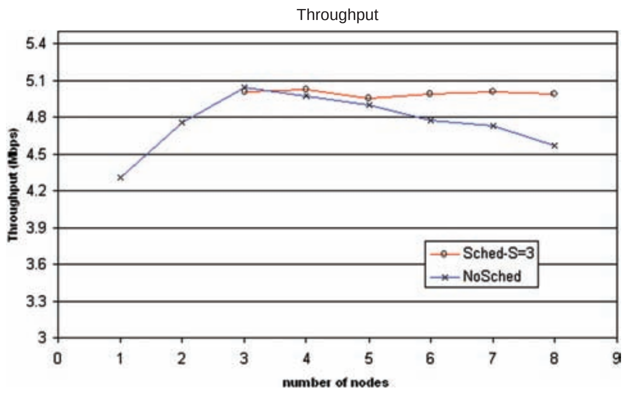

Fig. 10. Measurement results on aggregate throughput.

fits provided by our distributed scheduling algorithm. Our algorithm is independent of the number of nodes in the network and keeps the aggregate throughput very close to the maximum value. In the case of 8 nodes in the network, we measure an increase in throughput of about 9.2% which is very close to what we saw in the simulations in Section 3.2 (9.32%).

3.3.2. Power Consumption

In this section we measure the average power consumed by the wireless network interface (*Cisco Aironet 350 series*[4]). Applying an extender to the PCMCIA slot we measure the current needed by the communication device. We use the National Instrument DAQPAD6070E to sample the voltage fall on a resistor of 0.100 Ω. We acquire voltage samples while running experiments.

We evaluated different methodologies to save power during the inactive slots. The first one is the Power Saving Mode (PSM) provided by the IEEE 802.11.[2] A station in PSM turns off its radio and periodically wakes up to check if the AP has data buffered for it (the default interval is 100 ms).[2] If a station needs to transmit data, it switches on its radio and transmits it. The second technique we evaluated consists of turning off the Tx-power. The wireless cards we used in our experiments support this option. When the Tx-power is off, the station is not able to transmit or receive any data and it loses the connectivity with the AP. The power savings are larger than with the PSM since the station avoids periodically switching on the radio to listen to the AP messages. The delay to turn off the Tx-power is negligible but the wakeup time and time it takes to reach the transmit state again takes up to 450 ms.

Figure 11 compares the power consumption of the network interface card with 802.11 Power Saving Mode (PSM) enabled but without our scheduling algorithm (NO-Sched), with our scheduling algorithmand 802.11 PSM enabled (Sched-Idle), and with our scheduling algorithm and Tx-Power mode enabled (TX-power-OFF). The Tx-power mode turns off the Tx-power of the wireless device during the slots in which a node is not scheduled.

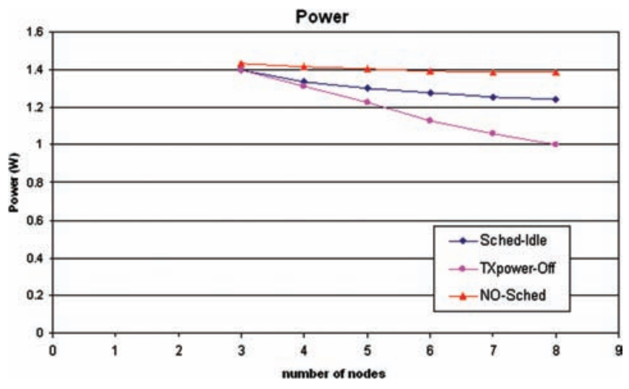

Fig. 11. Measurement results on the power consumption of the NIC.

Figure 12 shows measured node current while running our algorithm using the Tx-power mode. This technique saves about 23% of power when 8 nodes are in the network; with less than 4 nodes there are no measurable benefits. In our implementation we useslots of 300 ms.

Since we have a delay of 450 ms to wake up the NIC, a node needs at least 2 consecutive inactive slots to turn off/on Tx-power. If a node has just one inactive slot, it stays idle. We use an S parameter that is 3. This means that when there are three nodes in the network, a node is always scheduled. When there are 4 nodes, the probability for a node not to be scheduled for 2 consecutive slots is very low as each slot 3 out of 4 nodes get scheduled depending on their pseudo-random numbers. This probability increases with the number of nodes in the network and so enables us to save power.

Using the PSM provides a small advantage. As shown in Figure 11, our scheduling algorithm combined with the PSM can only achievemaximum power savings of 6.1%. In fact when a node transits from an active slot to an inactive one, it is not expected to go to sleep. This is because at the end of an active slot, a node is likely to have packets left to send in its MAC queue. So even if the node is not scheduled, it keeps transmitting until it empties its MAC queue. This has two main consequences; first, the node does not save power; second, the packets it transmits interfere with the data sent by the S active nodes elected by the scheduling algorithm. This increases congestion and causes throughput to fall. Instead with the Tx-power technique, the radio is off at the end of an active period and no packets can be transmitted, resulting in much larger power savings.

3.3.3. *Application Delay*

In this section we compare the average application delay with and without our algorithm for two different scheduling slot sizes: 0.1 s and 0.3 s. We assumeaverage node traffic rate of 136 Kbp sand the $S = 3$ to match the experiments in Section 3.2.1. The choice of the slot size has a large effect on the average packet delay as shown in Figure 13. As the slot size increases, the application delay increases accordingly. As explained in Section 3.2.2, the choice of the slot size is determined by a tradeoff between throughput and delay on one hand and power consumption on the other. This choice depend on node battery budget and application requirements. In the next section we show how traffic prioritization can help mitigate application layer delay for those applications sensitive to delay.

3.4. Traffic Prioritization

The delay introduced by the TDMA scheduling algorithm may represent a problem for those applications that have high priority packets that need to be delivered in a timely manner. Thus, we added a traffic prioritization mechanism so that when a node is not scheduled (thus unable to transmit), it buffers data to different priority queues. Policies such as Ref. [6] and DWRR[20] can be applied to decide the order of the outgoing packets once the node is active and can flush its buffers.

We implemented a WFQ policy at each node. The idea behind WFQ is to serve packets in the order in which they would have finished transmission in the fluid flow system (where traffic is infinitely divisible and a node can serve multiple flows simultaneously). WFQ is a type of packet-by-packet generalized processor sharing (GPS). It emulates

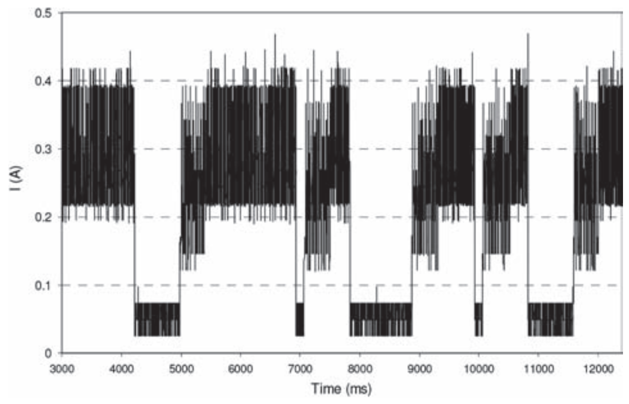

Fig. 12. Sample measurement of TX-Power technique.

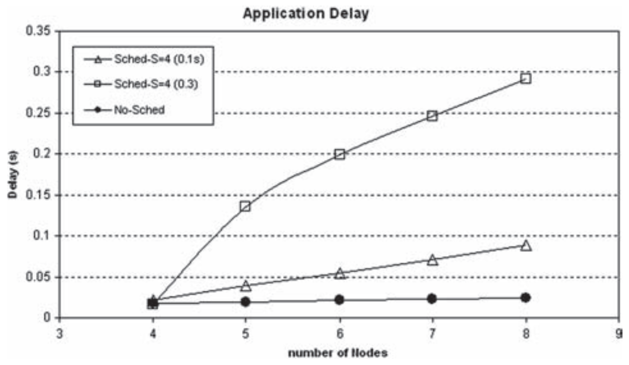

Fig. 13. Measurement results on application layer delay.

GPS by calculating the departure time of a packet (called the *virtual finish time*) and uses this virtual stamp to schedule the packets in the queues. In our implementation, a queue with weight w is associated with each flow with a different priority. The higher the w, the smaller the bandwidth given to the associated flow. A flow can be either *backlogged* (active) or *nonbecklogged* (inactive). A flow is *backlogged* when there is data in its queue, and *nonbacklogged* otherwise.

When a packet k enters a queue i, a sequence number Seq_k (representing its *virtual finish time*) is associated with packet k. Then, the packets in the queues are scheduled to be sent in increasing order of sequence numbers. The sequence numbers are calculated as follows. If packet k arrives while the flow is inactive (*nonbacklogged*), then its sequence number is:

$$Seq_k = roundnumber + (w_i * Size_k) \tag{3}$$

where the *round number* is the number of bytes sent so far and $Size_k$ is the size of packet k in bytes. w_i is the weight associated with queue/flow i. If the packet k arrives while the flow is active (*backlogged*), then the sequence number is:

$$Seq_k = Seq_k - 1 + (w_i * Size_k) \tag{4}$$

An example of running our implementation of WFQ is given in Figure 14. Three priority queues are defined, A, B, and C with weights 300, 600, 1000 respectively. The packet size is shown between parenthesis and the current *round number* is 100. Packets arrive in the order specified in the input queue on the left. When a packet arrives, its priority determines the queue it is assigned. Once it enters the queue, the *sequence number* is computed according to Eqs. (3) and (4). For instance, when packet $A1$ enters queue A that is inactive (*nonbacklogged*), its *sequence number* is: $SeqA1 = roundnumber + (wA * SizeA1) = 100 + (300 * 128) = 38500$. The *sequence numbers* are shown below each packet in the queue. Finally, packets are scheduled in increasing order of *sequence number* as shown in the output queue.

Figure 15 shows the measured application delay of experiments where packets are assigned a random priority

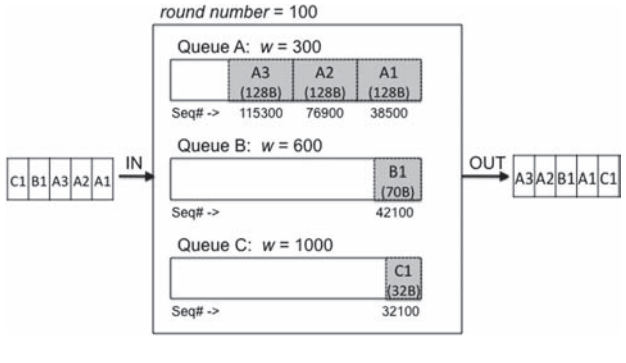

Fig. 14. Example of the WFQ policy implemented at node level.

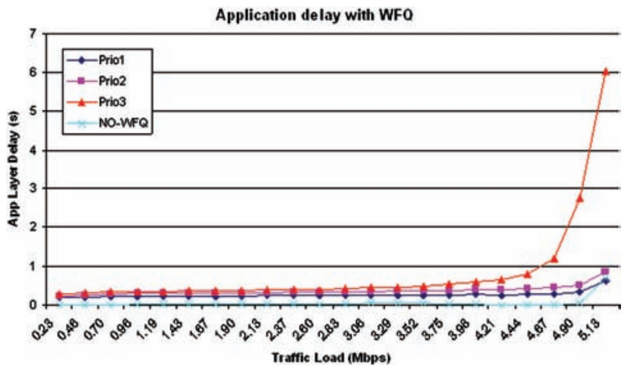

Fig. 15. Measurement results on application delay using prioritization.

and WFQ is applied. We define three priorities: 1, 2, and 3, corresponding to weights 100, 200, and 300 respectively. Priority 1 is then the highest since it is assigned the lowest weight. From Figure 15 we see that when the traffic load is high (above 5 Mbps), the scheduling algorithm results for high priority traffic (Prio 1) outperforms the default 802.11 (NO-WFQ). In fact, in high traffic load, high priority data has a lower application layer delay. Therefore, when the network is overloaded with data, our scheduler can not only save a lot of energy and improve throughput, but it can also provide a more predictable and lower application layer delay for high priority applications.

4. FORWARDING BACKBONE

Minimizing energy is an important challenge in wireless networks. While the scheduling algorithm presented in Section 3 is an efficient solution to save power, it does not consider the routing problem. In fact, by scheduling only a limited number of nodes at each slot, those nodes that are in sleep mode with their NIC off can break the connectivity of the network.

Connectivity in our context is defined as follows: if two nodes are reachable to each other through one or more hops when all the nodes are active, they must also be able to communicate under scheduling. This property must be true at all times (for every slot). In order to maintain connectivity, we combine scheduling with a forwarding backbone that is in charge of maintaining connectivity of the nodes. It is also responsible for the delivery of data throughout the network. The forwarding backbone is composed of a subset of nodes called *coordinators* that are guaranteed to stay active for a predefined period of time and are selected in a way that no node is left disconnected.

Creating such a backbone faces two main challenges related to energy consumption. First, being a *coordinator* is an energy expensive task because the node is required to keep the communication device active. The wireless interface, when active, is a power hungry component that drains batteries very quickly. Furthermore, the backbone is responsible for forwarding the data throughout the network, meaning that *coordinators* spend most of the time in the two most power consuming states that are the receive (PR_X) and transmit (PT_X) states (see Table II). This suggests that to lower the energy consumption, the number of *coordinators* should be minimized.

Second, if the forwarding backbone is defined as a static subset of nodes that become *coordinators*, then those nodes would quickly run out of energy. As a consequence, the network might become permanently disconnected and partitioned. Instead, it is desirable to have a homogeneous distribution of the energy consumption in the network to ensure longer network lifetime and avoid partitioning. Therefore, the task of being a *coordinator* should be fairly assigned among nodes and should dynamically change over time to ensure even distribution of energy in the network.

To achieve these goals we developed the forwarding backbone algorithm whose key component is the coordinator election process described in the next section.

The backbone algorithm creates a dynamically changing network of a subset of remaining active nodes to ensure reliable data forwarding and to maintain connectivity. In Chapter 1 we saw an example of how the scheduling and the forwarding backbone algorithms are combined (Fig. 4). They share the same TDMA scheme and synchronization. The *coordinators* are selected periodically in special communication slots (called *BcastSlot*s). The selection of the coordinators is based on the nodes' remaining battery lifetime and their *utility*. The *utility* is a measure of how many pairs of neighbors the node would connect if it becomes a part of the backbone. Intuitively, the *utility* is a parameter used to minimize the number of *coordinators* required to keep the nodes connected. Those nodes that connect the most neighbors should be selected as coordinators. Nodes volunteer to become *coordinators* through an announcement message. At the beginning of the *BcastSlot* each node sets up a delay dependent on its remaining battery and *utility*. The node that announces itself first, becomes the coordinator and is required to stay active until the next *BcastSlot* where the announcement process starts over. Therefore, the delay of the announcement is the key to the forwarding backbone creation mechanism. The more battery energy remaining at a node, and the more its *utility*, the less the delay is of the announcement and thus the higher its probability is in that particular node becoming a *coordinator*. This mechanism also ensures that the *coordinators* change dynamically as the battery of the nodes drains. Being a coordinator is an energetically expensive task because coordinators are required to stay active and forward the data coming from the neighbor nodes. The backbone algorithm makes sure that those nodes with more energy in their battery are selected to become part of the backbone. In the next section we describe how the coordinator election process works, formalize the concept of *utility*, and show how we compute the delay at the beginning of the *BcastSlot*.

Table II. Simulation parameters.

Parameter	Meaning	Value
PT_x	Transmission power	1400 mW
PR_x	Receive power	1000 mW
P_{Idle}	Power in idle state	830 mW
P_{sleep}	Power in sleep state—NIC off	43 mW
$P_{\text{idle\_to\_sleep}}$	Power during transition from idle state to sleep state	3 mW
$P_{\text{sleep\_to\_idle}}$	Power during transition from sleep state to idle state	7 mW
$T_{\text{idle\_to\_sleep}}$	Time for transition from idle state to sleep state	2 ms
$T_{\text{sleep\_to\_idle}}$	Time for transition from sleep state to idle state	10 ms

4.1. The Coordinator Election Process

A node volunteers to become a coordinator during the periodic BcastSlot (Fig. 16) by sending an announcement message. The announcement message contains information about the sender and its neighbors. It includes the list of the node's two-hop neighbors (as defined in Section 3), specifying which ones are the *coordinators* and which ones are not (the *status of a node*). If two neighbors of a non-coordinator node cannot reach each other directly or via a maximum of two coordinator hops, then such a node is eligible to become a coordinator.

Given the network graph $G = (E, V)$, node $vi \in V$ has a two-hop *subnetwork* $G_i = (E_i, V_i)$, let B_i be the the set of *coordinators* in the subnetwork of node v_i, $Bi \subseteq Vi$. Let $P_{j,k}$ be the set of paths of length 2 and 3 from nodes (v_j, v_k) where $vj, vk \in N(i) - \{vi\}$ and $vj \neq vk$. Formally, the coordinator eligibility rule states that node v_i is eligible to become a coordinator if:

$$\forall v_j, v_k \in N(i) - \{v_i\}, \quad v_j \neq u_k, \quad \nexists p \in P_{jk} | p_i \in B, \quad \forall p_i \; in \; p = (p_j, \dots, p_k) \tag{5}$$

The nodes "compete" during the *BcastSlot* by carefully timing their announcements for becoming a coordinator. This delay is computed as a function of the residual energy currently available at the node and its *utility*. The *utility*, formally defined below, is the number of additional pairs

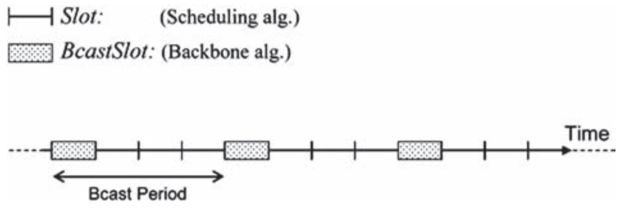

Fig. 16. Broadcastslot and broadcast period.

of nodes C_i among the neighbors N_i that would be connected if node i became a coordinator.

$$\text{utility}_i = \frac{C_i}{\binom{N_i}{2}} \tag{6}$$

Let E_{ri} denote the amount of energy at a node that still remains and E_{mi} the maximum amount of energy available at the same node. We define the delay of the announcement message as:

$$\text{delay}_i = \left(\left(1 - \frac{E_{ri}}{E_{mi}}\right) + (1 - \text{utility}_i)\right) \times N_i \tag{7}$$

The delay is normalized to the duration of the slot. As more energy is available at a node, its *utility* is higher, then the delay to sending an announcement message is lower, and it gives the node a higher chance of becoming a coordinator.

At the beginning of a *BcastSlot* a node checks if the eligibility rule holds. If it does so, it sets the delay for the announcement message according to Eq. (7). While a node waits for the delay to expire, it might receive other announcement messages from its neighbors which have set a lower delay, and possibly changing the knowledge of the *status* of its neighbors (again, *status* specifies whether a node is a coordinator or not). Therefore, just before broadcasting the announcement message, a node must check if the coordinator eligibility rule still holds. Then the header of the announcement message is filled out with the information regarding the node's neighbors with their status and the status of the node itself. Finally, the announcement message is broadcast. At the end of the *BcastSlot*, those nodes that are coordinators must stay awake until the next *BcastSlot* and forward the data coming from its neighbors.

4.2. Combining Scheduling and Forwarding Backbone

The scheduling and forwarding backbone algorithms run on the same TDMA scheme. As shown in Figure 16, time is divided into fixed-size slots. Periodically, during the *BcastSlot* all the nodes are required to be active. In the *BcastSlot*, all the nodes run the backbone algorithm to create the set of *coordinators* that build the data forwarding backbone. At the end of the *BcastSlot*, those nodes that are*coordinators* must remain active until the next *BcastSlot* when the backbone algorithm is run again and a new set of coordinators are selected according to the mechanism described in the previous section. Through the announcement process happening during the *BcastSlot*, nodes also become aware of their neighbors. Thus, the cases of new nodes joining or leaving the network (including node failures), will be detected at most in a *Bcast Period*.

In every slot other than the *BcastSlot*, those nodes that are not coordinators run the distributed scheduling algorithm described in Section 3 to save power. The output of the scheduling algorithm is the set of active/sleeping nodes. This information, together with the information about the coordinator nodes, is made available to the routing layer to make the proper routing decision. For example, in our experiments we use a geographic forwarding routing protocol in which a node first attempts to forward the packets to the *coordinator* that is closest to the destination. If such a *coordinator* does not exist, it tries to find a forwarder among the active non-*coordinator* nodes. If again a forwarder is not found, then the node drops the packet.

5. RESULTS—COMBINED SCHEDULING AND ROUTING

In this section we evaluate the performance of our solution that combines scheduling with the forwarding backbone mechanism. The main purpose of our solution is to save energy. Energy consumption is dependent on the number of active nodes in the network and in Section 5.1.1 this relationship is discussed and quantified. We are also interested on the delay applications should expect when using our solution. In Section 5.1.2, we show the results on application delay under different scenarios and network topologies. We compare our results to SPAN algorithm[3] via simulations. Next we describe the simulation setup and present the results obtained.

5.1. Simulation Setup

Simulations were conducted using the *ns-2* network simulator[21] version 2.33. We reproduce a simulation environment very similar to what we observed in the Santa Margarita 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: 1250 m $\times$ 1250 m, 1000 m $\times$ 1000 m, 750 m $\times$ 750 m, and 500 m $\times$ 500 m. Twenty nodes send and receive traffic. Each of these nodes, if not otherwise specified, generates 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. Packet size and traffic rate are the same as in Ref. [3]. To make sure that each CBR flow goes through multiple hops before reaching their 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 the SPAN algorithm,[3] which we compare our backbone creation algorithm to. The rest of the simulation parameters are shown in Table II.

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 a *coordinator* does not 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.

5.2. Energy Results

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 17 shows that for each simulation area, as the S parameter increases so does the energy consumption. Less energy is consumed by the dense topologies (Fig. 17). 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 consumption (summarized in Fig. 17) is the number of hops a data packet needs to traverse from source to destination. The more number of hops, the more the nodes are involved in receiving and transmitting data. These operations are power hungry (Table II). Predictably, the larger the network area (and so the longer the distance from sources to destinations), the more hops are required. In fact, for topologies of increasing sizes 500 m × 500 m, 750 m × 750 m, 1000 m × 1000 m, and 1250 m × 1250 m, we recorded an increasing average number of hops that are 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 18 shows the effect of slot size (100 ms, 200 ms, 300 ms) on energy consumption. We show only the results for the 1000 m × 1000 m topology, as we got similar results for the other scenarios. As expected, larger slot sizes lead to larger power savings. Shorter slots cause more frequent state transitions and thus greater energy consumption.

We compare our results with SPAN.[3] For the case of the 1000 m × 1000 m topology, the energy savings shown for SPAN in Ref. [3] (page 9 Fig. 8 in SPAN paper) approach ~50% as 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%. 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, does not require any modifications to 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.

5.3. Latency Results

Figure 19 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.

The impact of the S parameter on the delay is more evident on the largest simulation area, 1250 m × 1250 m. 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.

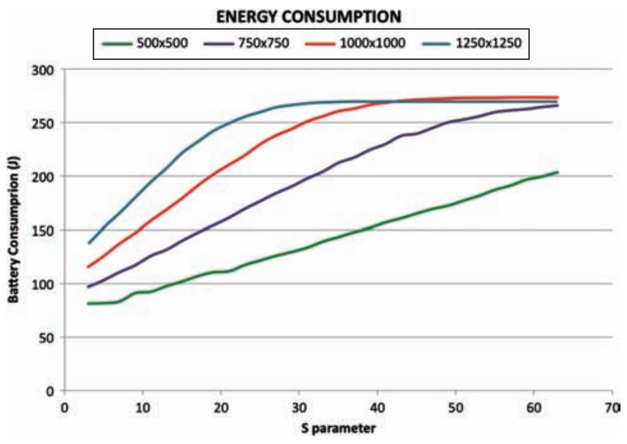

Fig. 17. Average per node battery consumption (J) as a function of the S parameter.

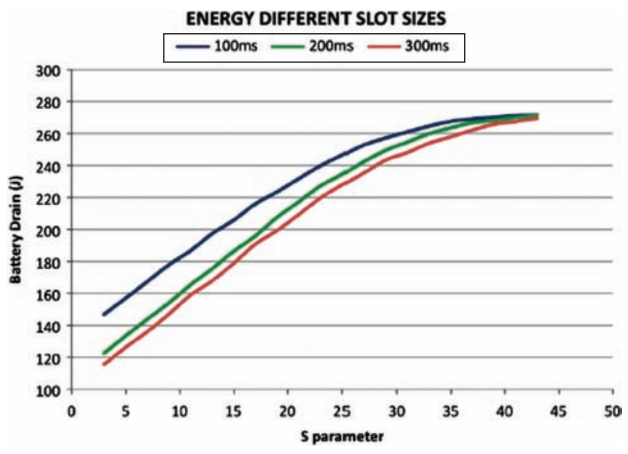

Fig. 18. Average per node battery consumption (J) as a function of S.

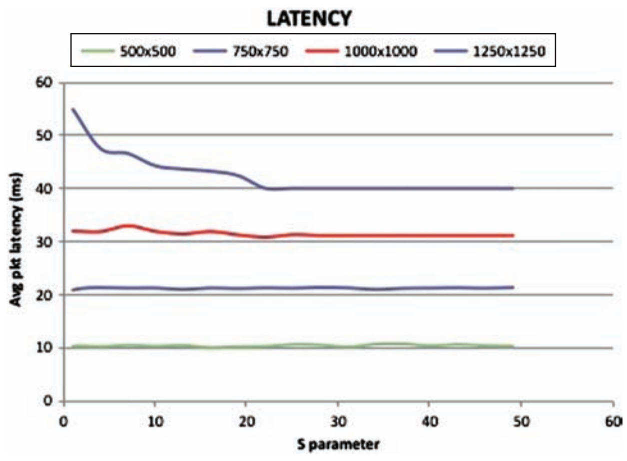

Fig. 19. Average latency (ms) as a function of S and area.

Compared to SPAN,[3] we achieve better latency results. From Ref. [3] we see that in the case of the 1000 m × 1000 m topology, SPAN's average packet latency is 40.5 ms with 6.1 average hops. With the S parameter set to 2 (the number for which we obtain the same energy savings as SPAN), our solution decreases in latency by 20.7% with an average number of hops of 6.4. 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.[14, 7] Secondly, those packets that cannot be routed by a coordinator go through one or more scheduled nodes as shown in Figure 20. A scheduled node, compared to a node in PSM as in the SPAN case, is capable of much faster forwarding.

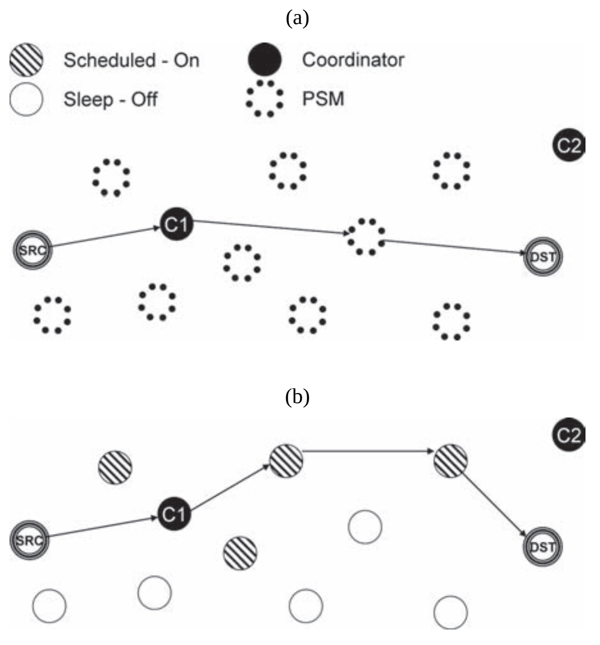

Fig. 20. Forwarding through nodes in PSM versus active nodes. (a) Forwarding through a node in PSM. (b) Forwarding through active nodes.

6. CONCLUSION

We presented an energy efficient mechanism for scheduling and routing in heterogeneous wireless sensor networks such as HPWREN. This network offer the challenge of ensuring that node battery lifetime is long enough for data collection and delivery to happen in a timely manner while using COTS to ensure low cost and ease of deployment. We present a distributed scheduling algorithm that allows a large portion of nodes to switch off the NIC thus saving energy. Scheduling is combined with the creation of a routing backbone of nodes in charge of delivering data to the proper destinations. Since being part of the backbone requires the node to stay awake continuously for a certain amount of time, the nodes of the backbone are dynamically selected so that those nodes that have more energy are more likely to become part of the backbone.By requiring no modifications to the MAC layer, our solution can be easily and quickly deployed on existing networks such as HPWREN where neither legacy devices 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 previous work,[3] the solution presented in this paper achieves great power savings (up to 53%) while delivering packets with a 20% lower delay.

References

1. G.-S. Ahn, A. T. Campbell, A. T. Campbell, A. Veres, and L. Hsiang Sun, Swan: Service differentiation in stateless wireless ad hoc networks. *In Proc. IEEE Infocom.* 2002, 457 (**2002**).

2. I.-S. S. Board, Wireless LAN Medium Access Control (MAC) and Phisical Layer (PHY) Specifications (**2003**).

3. B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, In 7th ACM MOBICOM, Rome, Italy (**2001**).

4. Cisco, Cisco Aironet 802.11a/b/g Cardbus Wireless Lan Adapter Data Sheet, http://www.cisco.com (**2007**).

5. R. A. Corporation, C. S. Raghavendra, and S. Singh, Pamas—Power aware multi-access protocol with signalling for ad hoc networks. *In ACM Communications Review* 28, 5 (**1999**).

6. A. Demers, S. Keshav, and S. Shenker, Analysis and simulation of a fair queueing algorithm, In SIGCOMM '89: Symposium proceedings on Communications Architectures and Protocols, ACM, New York, NY, USA (**1989**), pp. 1–12.

7. T. S. R. Edoardo Regini and D. Lim, Scheduling above mac to maximize battery lifetime and throughput in wlans. *In IASTED* (**2008**).

8. T. ElBatt and A. Ephremides, Joint Scheduling and Power Control For Wireless Ad Hoc Networks (**2002**).

9. I. T. Haque, C. Assi, and J. W. Atwood, Randomized energy aware routing algorithms in mobile ad hoc networks. *In MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM, New York, NY, USA (**2005**), pp. 71–78.

10. HPWREN, The High Performance Wireless Research and Educational Network, http://www.hpwren.ucsd.edu/ (**2009**).

11. Intel, Intel pxa27x Processor Developer's Kit, http://www.intel.com/pressroom/kits/pxa27x/index.htm (**2006**).

12. K. Jamieson, Implementation of A Power-Saving Protocol for Ad Hoc Wireless Networks, Master's Thesis, MIT (**2002**).

13. J. Snow and W.-C. Feng, Implementing a low power tdma protocol over 802.11, *In Proc. IEEE WCNC* (**2005**).

14. D. Lim, Distributed Proxy-Layer Scheduling in Heterogeneous Wireless Sensor Networks, Master's thesis, Univesity of California, San Diego (**2007**).

15. D. Lim, J. Shim, T. S. Rosing, and T. Javidi, Scheduling data delivery in heterogeneous wireless sensor networks. *In ISM '06: Proceedings of the Eighth IEEE International Symposium on Multimedia*, IEEE Computer Society, Washington, DC, USA (**2006**), pp. 575–583.

16. S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, L. Stibor, C. poll Contention, and F. Poll, Ieee 802.11e Wireless Lan for Quality of Service (**2002**).

17. L. Pond and V. Li, A distributed time-slot assignment protocol for mobile multi-hop broadcast packet radio networks. *IEEE MILCOM* (**1989**).

18. R. Ramaswami and K. K. Parhi, Distributed scheduling of broadcasts in a radio network. *In INFOCOM* 497 (**1989**).

19. B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, Opportunistic media access for multirate ad hoc networks, *In MobiCom '02: Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, ACM, New York, NY, USA (**2002**), pp. 24–35.

20. M. Shreedhar and G. Varghese, Efficient fair queueing using deficit round robin. *SIGCOMM Comput. Commun. Rev.* 25, 231 (**1995**).

21. N. Simulator, 2: ns-2. http://www.isi.edu/nsnam/ns.

22. S. Singh, M. Woo, and C. S. Raghavendra, Power-aware routing in mobile ad hoc networks, *In MobiCom'98: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking* ACM, New York, NY, USA (**1998**), pp. 181–190.

23. A. R. I. Stoica, An Overlay Mac Layer for 802.11 networks. *In in MobiSys* 135 (**2005**).

24. N. H. Vaidya, P. Bahl, and S. Gupta, Distributed fair scheduling in a wireless lan, *In MobiCom'00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ACM, New York, NY, USA (**2000**), pp. 167–178.

25. S. Waharte, J. Xiao, and R. Boutaba, Overlay wireless sensor networks for application-adaptive scheduling in wlan. *In HSNMC* 676 (**2004**).

26. W. Xiuchao, Simulate 802.11b channel within ns2. Technical report, National University of Singapore (**2004**).

27. W. Xiuchao and A. L. Ananda, Link characteristics estimation for ieee 802.11 dcf based wlan, *In LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, IEEE Computer Society, Washington, DC, USA (**2004**), pp. 302–309.

28. E. Arikan, Multi-access in packet-radio networks, MS Thesis, MIT.

29. I. Cidon and M. Sidi, Distributed assignment algorithms for multihop packet radio networks. *IEEE Trans. Comput.* 38, 1353–1361.

Edoardo Regini

Edoardo Regini *received a B.S. degree in Computer Science at the University of Urbino, Italy in 2005. In 2006, after a collaboration with the Computer Science and Technology Institute in Urbino, Italy he moved to San Diego, California and continued his research on energy-efficient routing and scheduling algorithms for wireless networks at the University of California San Diego where, in 2009, he received his M.S. degree in Computer Science. He is currently employed at Qualcomm Inc., San Diego working on CPU power management.*

Daeseob Lim

Daeseob Lim *is currently working at SAP labs in Korea. He obtained his MS in CSE in 2006 from UCSD. His thesis topic was "Distributed Proxy-Layer Scheduling in Heterogeneous Wireless Sensor Networks." His interests are in energy efficient system design, embedded computing and wireless sensor networks.*

Tajana Šimunić Rosing

Tajana Šimunić Rosing *is currently an Associate Professor in Computer Science Department at UCSD. Her research interests are energy efficient computing, embedded and wireless systems. Prior to this she was a full time researcher at HP Labs while working part-time at Stanford University. At Stanford she has been involved with leading research of a number of graduate students and has taught graduate level classes. She finished her Ph.D. in 2001 at Stanford University, concurrently with finishing her Masters in Engineering Management. Her Ph.D. topic was Dynamic Management of Power Consumption. Prior to pursuing the Ph.D., she worked as a Senior Design Engineer at Altera Corporation. She obtained the MS in EE from University of Arizona where her thesis topic was high-speed interconnect and driver-receiver circuit design. She has served at a number of Technical Paper Committees, and is currently an Associate Editor of IEEE Transactions on Mobile Computing.*