SCHEDULING ABOVE MAC TO MAXIMIZE BATTERY LIFETIME AND THROUGHPUT IN WLANS

Edoardo Regini, Daeseob Lim, Tajana Simunic Rosing
Dept. of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
{eregini, dalim, tajana}@ucsd.edu

ABSTRACT
Maximizing battery lifetime and throughput is important in today's WLAN networks. In this work, we propose a distributed scheduling algorithm *above* the MAC layer that addresses these problems. We test our ideas on a heterogeneous wireless sensor network we have deployed in southern California – HPWREN. The results show that with our scheduling algorithm it is possible to achieve a throughput improvement of up to 10.31% and maximum power saving of 85.54%.

KEY WORDS
Distributed scheduling, contention, interference, TDMA, queuing network modelling

1. Introduction

WLANs gained popularity for their convenience, cost efficiency and easy deployability. When employed in large-scale multicell wireless networks, the 802.11 protocol can see significant reductions in throughput due to contention and interference. Figure 1 shows the results of a simulation that calculates the aggregate throughput in a single cell scenario while nodes are transmitting at full MAC queues. 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 [21]. Intuitively, from Figure 1 it is clear that we can get a higher throughput by limiting contention to only a few nodes at a time. The algorithm proposed in this work basically uses this property to improve throughput and reduce power consumption. This is achieved by applying a TDMA scheduling algorithm *above* MAC that selects a limited number of active nodes in each slot in order to reduce intra-cell contention and inter-cell interference. Those nodes that are not scheduled in a slot save power by switching off the communication device.

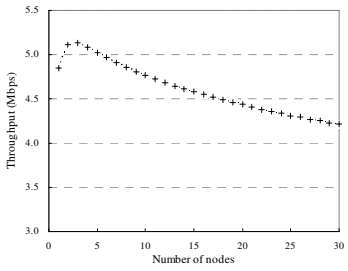


Figure. 1. Aggregate throughput drop in 802.11 network

In a wireless network, such as the HPWREN [2] in southern California, this improvement can help applications use network resources more effectively and achieve longer battery lifetime. The HPWREN is a heterogeneous wireless sensor network (WSN) ideally organized with a three layer structure. At the top layer the parent cluster heads nodes (*parent CHs*) provide long distance high bandwidth connectivity. At the bottom layer there are the sensors deployed in the field. Our scheduling algorithm was designed to fit the needs of the middle layer of the HPWREN that is composed by child cluster head nodes (*child CHs*). The *child CHs* collect the outgoing data from the underlying sensor network and deliver it to the top layer. In this hierarchical structure performance is threatened by contention and interference at the middle layer due to the huge amount of data to be forwarded. Furthermore, the *child CHs*' battery lifetime is continuously shortened by the intensive use of the wireless network interface, typically an 802.11b device. Our algorithm handles efficiently the high traffic load and provide higher throughput while increasing the battery lifetime of the *child CHs*.

The proposed scheduling algorithm *overlays* the MAC layer. The advantage of this choice resides in the immediate deployability and low cost because it doesn't require any modification of the existing MAC layer and legacy network devices can be used. MAC layer changes tend to be expensive as they usually involve design of new hardware, firmware and device drivers.

The rest of the paper is organized as follow. In Section 2 we explore some of the related. Section 3 describes the details of our scheduling algorithm. In Section 4, we evaluate the performance of our algorithm and show the results of simulations and measurements taken on a test network of 8 nodes. Finally we conclude in Section 5.

2. Related Work

Packed-based wireless networks suffer from contention and interference when multiple nodes try to access the channel simultaneously. This results in inefficient channel

utilization. Throughput degradation in wireless networks has been investigated in detail in literature such as in [10] and [7]. It becomes even worse in real networks [5], [16].

One approach to solve the performance degradation is to revise the MAC layer algorithms. The original DCF algorithm cannot give prioritized service to user. Enhanced DCF (EDCF) [12] prioritizes traffic categories with different contention parameters and gives more chance of channel access to high priority traffic. The Distributed Fair Scheduling (DFS) in [15] differentiates the *backoff* interval (BI) according to the packet length and traffic class. The Opportunistic Auto Rate (OAR) protocol in [14] exploits the automatically adjusted transmission rate of 802.11.

The second approach is to apply scheduling *over* the MAC layer. Overlay MAC layer (OML) [13] adds an additional conceptual layer over the existing 802.11 MAC, thus enabling use of off-the-shelf components. 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 [4] is a rate control mechanism for TCP and UDP traffic which works on the best-effort MAC. SWAN improves throughput and achieves good fairness among different 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 mobile nodes is not considered in their work.

Our scheduling algorithm is also above the MAC layer. Each node determines its schedule locally using the information of the nodes within one to two-hops. It makes use of pseudo-random numbers to resolve contention. A similar contention resolution algorithm is presented in [20], though [20] operates within the MAC layer. As in [8] we focus on saving communication power, but we also improve the network capacity in terms of aggregate throughput. We control the channel access of cells on a high level (with the cell-level scheduling) and the channel access of nodes on a lower level (with distributed node-level scheduling). In the next section we describe our scheduling algorithm.

3. Scheduling Algorithm

Our distributed multicell scheduling algorithm uses a TDMA scheme. The cell-level scheduling decides which cells are active in each time slot in order to avoid interferences between neighboring cells. Active cells run the node-level scheduling that schedules the child CHs that are allowed to transmit to avoid excessive contention. If a child CH is not scheduled, it saves power by turning off its communication device and starts buffering the incoming data. The effects of using different time slot sizes are analyzed in the Section 4.2.

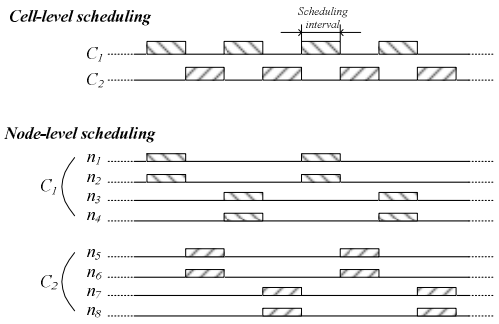


Figure 2. Combined cell-level and node-level scheduling

An example of a schedule produced by our algorithm is shown in Figure 2. In this example, there are two cells C1 and C2, and eight child CHs nodes n1, …, n8. The child CHs n1, n2, n3 and n4 are in Cell C1, and the other are in C2. In this example we use the same size of scheduling interval for cell-level scheduling and node-level scheduling. When C1 is active, the child CHs n1 and n2 run the node-level scheduling algorithm and find that they are scheduled in the next scheduling slot. Subsequently, n1 and n2 wake up from a sleep mode and transmit their data to their parent CH. Since the other child CHs n3 and n4 see that they are not scheduled, they wait for the next schedule in sleep mode. Because C2 is not active in the first scheduling slot, n5, …, n8 wait in sleep mode even if the result of node-level scheduling algorithm says that they are schedulable. The packets of nodes n5, …, n8 will have a delay dependent on the duration of the scheduling interval. We show the results on application layer delay in Section 4.

In our scheduling algorithm, we assume that the timers in all parent and child nodes are loosely synchronized. The 802.11 provides a synchronization mechanism ([19] chapter 7.6) that is accurate enough for our needs. We next provide details of node- and cell-level scheduling algorithms.

3.1 Node-level Scheduling

The goal of the node-level scheduling algorithm is to limit the number of active nodes in a cell in order to achieve and maintain the maximum throughput and allow the inactive nodes to save power by turning off their NIC. Furthermore, it must consider packet priorities and organize data transmissions accordingly. The number of active nodes S that maximizes throughput can be easily estimated through simulations or through basic throughput measurements as shown in Figure 1 and Figure 10(a) respectively.

In order to describe the node-level scheduling algorithm, let 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 other node $v_j \in V$, then $(v_i, v_j) \in E$. The schedule assignment is a

1. Update the *two-hop distance subnetwork* $G_i=(E_i, V_i)$
2. Assign S tickets to each node in the subnetwork:
$$tk(v_j) \leftarrow S \text{ for } \forall v_j \in V_i$$
3. Generate pseudo-random numbers for each node in the *subnetwork*:
$$rn_j = rand(id_j + slotno),\ \forall v_j \in V_i$$
4. Add the nodes into a set of unchecked nodes:
$$V'' \leftarrow V$$
5. Pick a node in the decreasing order of pseudo-random numbers of nodes.
6. Determine if v_j can be scheduled. v_j is schedulable iff
$$tk(v_j) \geq 1 \text{ for } \forall v_j \in \{v_i\} \cup (N(v_i) \cap AV)$$
7. If v_j is schedulable, add it to the assignment of active nodes, $AV \leftarrow AV \cup \{v_j\}$. And decrease the tickets of v_j and all its neighbors by 1, $tk(v_j) = tk(v_j) - 1$ for $\forall v_j \in \{v_j\} \cup N(v_j)$.
8. Remove v_j from , $V'' \leftarrow V'' - \{v_j\}$.
9. If V'' is empty, then continue. Else go to 3.
10. If v_i is schedulable, $v_i \in AV$, then start the data transmission. Use packet prioritization to control delay.
11. If v_i is not schedulable, check if it is eleigible to become a *backup node*.

Figure 3. Pseudo-code node-level scheduling.

set of active nodes *AV*, 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 $AV = \bigcup_{v_i \in V} AV(v_i)$. The basic constraint in the node-level scheduling problem is that the number of neighboring active nodes should not exceed a given constant (in our case *S*); formally

$$\forall v_i \in V, |AV(v_i) < S| \quad (1)$$

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

In our solution 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 equation (1). Each node has only knowledge of its *two-hop* neighbors, thus our solution is scalable. Given a node $v_j \in V$, the *two-hop* distance *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 $E_i \subseteq E$, $V_i \subseteq V$. To keep updated the information about the neighborhood, broadcast messages are generated when a node joins the network or a failure is detected.

Pseudo code of node-level scheduling algorithm is shown in Figure 3. The first step of each node is updating its two hop *subnetwork* according to its latest information (step 1). In the example in Figure 4, node v_1 will find its *subnetwork* to be v_{11}, v_5, v_3, v_{10} and v_8. The number of tickets $tk(v_j)$ of each node is then initialized to *S* (step 2), where *S* is the constant number of active nodes for each slot. In our example *S*=1 and thus v_1 assigns 1 ticket to nodes v_{11}, v_5, v_3, v_{10} and v_8; v_2 assigns 1 ticket to v_8, v_{12},

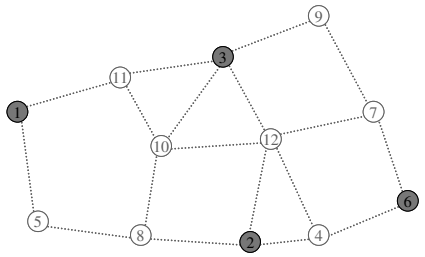


Figure 4. Example of node-level scheduling, $S = 1$

1. v_i runs the DNLS
2. **If** $v_i \in AV(v_i)$
   then exit and sleep
   else $AV \leftarrow AV(v_i)$
3. **If** $|AV| = 0$
   then exit and sleep
   else get $v_{max} \in AV$ where v_{max} is the (active) node with the greatest ID
4. **If** (*id* of v_i) > (*id* of v_j) for all $v_j \in N(v_{max}) - AV(v_{max})$
   then continue
   else $AV \leftarrow AV(v_i) - \{v_{max}\}$ and go to 3.
5. v_i is a *backup node*. Wait for interval *t*.
6. **If** no data is sent from v_{max} during interval *t*
   then v_i elects itself active for the rest of the current slot
   else go to sleep

Figure 5. Pseudo-code *backup* node

v_4, v_5, v_{10}, v_3, v_7 and v_6. In step 3 pseudo-random numbers are generated for each node v_j in the *subnetwork* using as seed the sum of the node ID and the current sequential slot number. For instance, v_1 generates pseudo random numbers 11, 5, 3, 10 and 8 for v_{11}, v_5, v_3, v_{10} and v_8 respectively. In step 4 a node creates a set of unchecked nodes *V''* that includes all nodes in its subnetwork and itself. The node v_j with the smallest random number in the subnetwork is selected in step 5 in order to check if it is schedulable. In our example node v_1 selects itself since it has the smallest random number. v_{12} selects v_2, and so on. Selected nodes are checked for schedulability in Step 6. A node v_j is schedulable only if itself and the active neighbors have at least one ticket. For example v_1 find itself schedulable; same does v_{12} for v_2. If a node v_j is scheduled, it is added to the set of active node *AV* and the tickets of v_i and of all its neighbors are decreased (step 7). For example v_1 adds itself to *AV* and decrease the tickets of v_1, v_{11}, v_5, v_3, v_{10} and v_8. Otherwise v_j is removed from *V''* (step 8). The process repeats until there are nodes in *V''*(step 9). When *V''* is empty a node either starts the transmission of data (step 10), or it checks whether it is eligible to become a *backup node*. For example node v1 stops when it has removed v_{11}, v_5, v_3, v_{10} and v_8 from *V''*and find itself in *AV*. If a node is not scheduled (step 11) it can *become a backup node*. A set of *backup nodes* is selected at every slot to recover from situations when a node does not have the data ready to be sent or for any

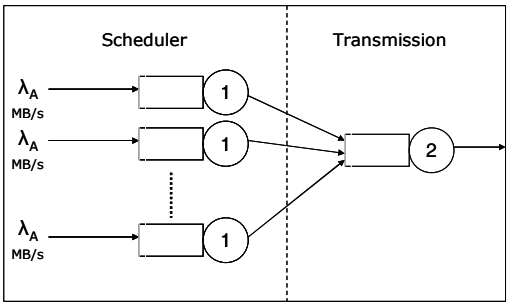


Figure 6. A model of the node-level scheduling algorithm

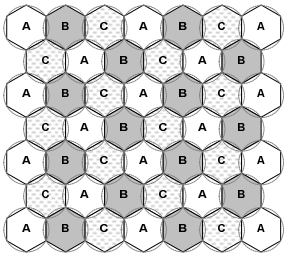


Figure 7. Cell-level scheduling in hexagonal multicell topology

reason it fails to send its data. A *backup node* stays awake to monitor the activity of an active node for a short interval of time. If the *backup node* doesn't overhear any data transmission
from the active node, it elects itself schedulable in the current slot and start transmitting its data. Otherwise it goes to sleep. A node v_i elects himself a *backup node* and decide whether to be active in the current slot in a distributed fashion according to the pseudo code in Figure 5. Due to space limitations we skip the step-by-step explanation.

3.2 A queuing network model for the node-level scheduling

A queuing network model [21] for the node-level scheduling is depicted in Figure 6.
Subsystem (1) (the scheduler) receives *requests* (packets) at rate λ_A and has throughput $X_1 = \lambda_A$. The residence time R_1 is the dalay due to the TDMA scheduling and is:

$$R_1 = P \cdot (0.5 \cdot sSize) + (1-P) \cdot (0.5 \cdot (nN/nS) \cdot sSize) \qquad (2)$$

where nN is the total number of nodes, nS is the number of active nodes in each slot and P is the probability for a node to be scheduled. All nodes have the same probability P to be one of the nS scheduled nodes out of nN, thus $P = (nS / nN)$. Also they have probability $(1-P)$ not to be scheduled. Since in each slot new random numbers are generated, these events are independent and so the probability for a node not to be scheduled after n slots is:

$$(1-P)^n = (1-nS/nN)^n \qquad (3)$$

We measured that in our test network with $N = 8$ and $S = 4$, each node is scheduled after 1.7 inactive slots as an average. All the N nodes are scheduled on average at least once every 2.7 slots with a maximum of 8 slots and a minimum of 2 slots. Using *Little's Law* [21] the average number of *requests* is: $N_1 = X_1 \cdot R_1$.
Subsystem (2) represents the transmission of packets. It has throughput $X_2 = \lambda_A$. *Residence time* R_2 is a characteristic of the 802.11 MAC and the traffic load generated by the active nodes. Using our simulation results and measurements we found that it can be represented with the following exponential function: $F(x) = a \cdot e^{(b \cdot x)}$, where a and b are coefficients with 95% confidence bounds. Then the residence time is

$$R_2 = F(nS \cdot \lambda_A) + const \qquad (4)$$

where $F(nS \cdot \lambda_A)$ is the delay for the nS scheduled nodes; $const$ is the average delay for a packet to traverse the protocol stack from the MAC to the application layer. We can assume this value to be constant and close to zero. Finally, using *Little's Law*, the average number of *requests* (packets) is $N_2 = X_2 \cdot R_2$.

3.3 Cell-level Scheduling

Through simulations we found that even using the technique described in [11] to reduce packet collisions to ~1%, the interference rate is still very high, more than 35%. As a result, our scheduling needs to limit both contention within a cell and interference from neighboring cells. In this work, we propose a simple cell-level scheduling algorithm for the hexagonal topology in Figure 7 since it well approximate the HPWREN's topology. The active cells scheduled at the same time slot are indicated with the same letters (A, B, or C) and with the same color patterns. The scheduling scheme is calculated statically a priori according to the specific network topology.

4. Experimental Evaluation

4.1 Simulation Results

We run simulation on the ns2 network simulator [3]. Simulation parameters for 802.11b are set for the typical IEEE 802.11b wireless channel [17] [18]. The parameters for power in each power mode are from the data sheet of the Cisco Aironet wireless LAN adapter [1] and measurements presented in [9]. The size of the network is 533m by 550m. Hexagonal network topology shown in Figure 7 well approximates the density of HPWREN subnetwork we used to collect data traces.
1) Results for data traffic collected at HPWREN: The scheduling algorithm achieves great power savings as shown in Figure 8(a) where up to 85.54% of communication power is saved. Average throughput and MAC layer transmission delay are shown in Figure 8(b) and Figure 8(c). Throughput is improved by up to 10.31%. Using the 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 of interface device. However, it is possible to handle efficiently the delay by prioritizing traffic, such as with weighted fair queuing [6] as discussed in section 4.2. Another way to reduce the application layer delay is by minimizing the size of the scheduling slots.

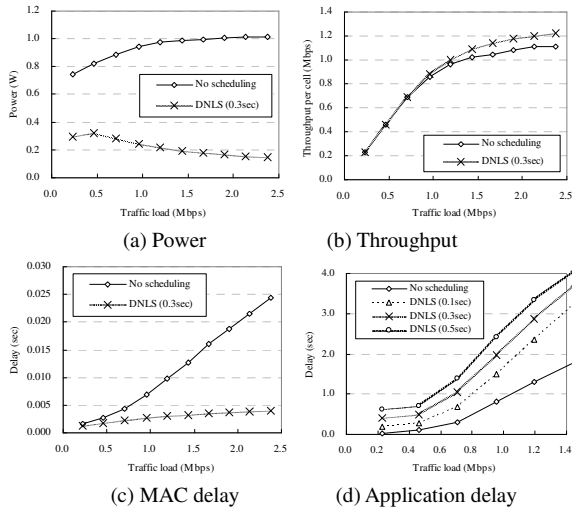
2) Effect of different scheduling slot size: The results on the average throughput and communication power with different slot sizes are shown in Figure 9. In Figure 9 (a), we see that the amount of overhead in scheduling is inversely proportional to the length of the scheduling slot. If the scheduling 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. While the wireless interface switches its mode, it consumes at least as much power as it does in the idle mode. Mode transition also takes a certain transition time to wake up and to go 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 saves more communication power it is better to use longer time slots. Though, it causes a longer delay. This tradeoff is a key issue in determining the scheduling slot size.

4.2 Measurements

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 (*Apple AirPort Base Station*). A *child CH* is an *Intel PXA27x board* [22] with a *Cisco Aironet 350 series* wireless LAN PCMCIA adapter [1] that have similar characteristics to the device used by *child CHs* in HPWREN.

1) Throughput: Figure 10 (a) shows the specific performance degradation of our test network. In this experiment, each node generates high rate 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 simulation because of a small interference still present on campus even during off hours. Figure 10 (a) also shows that as the number of nodes increases, our algorithm keeps the aggregate throughput very close to the maximum value; it is independent of the number of nodes in the network. In the case of 8 nodes, we measure an increase in throughput of about 9.2% which is very close to what we saw in simulations (9.32%).

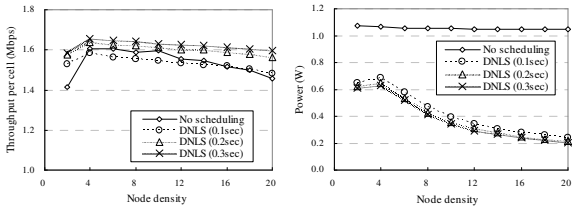
2) Power consumption: Applying an extender to the PCMCIA slot we measure the current needed by the communication device (Cisco Aironet 350 series). We use the National Instrument DAQPAD6070E to sample the voltage fall on a resistor of 0.100 Ω. We acquire voltage



(a) Power (b) Throughput

(c) MAC delay (d) Application delay

Figure 8. Simulations results in hexagonal multi-cell



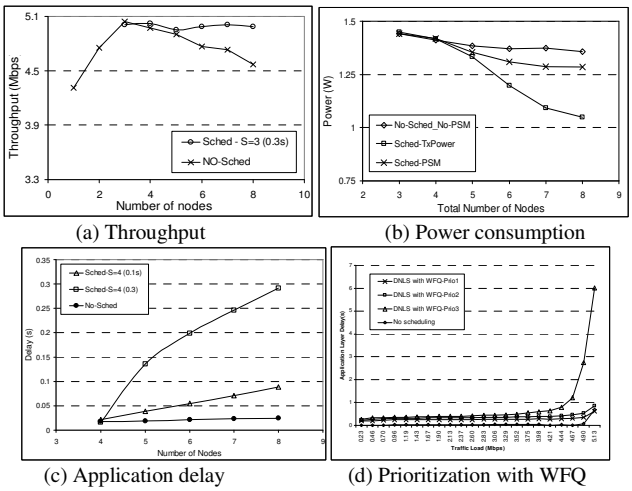
(a) Throughput (b) Power

Figure 9. Simulations on throughput and power for different slot sizes

samples while running experiments. We evaluate two different methodologies to save power during the inactive slots. The first one is the Power Saving Mode (PSM) provided by the 802.11 [19]. The second is turning off the TX-power. Figure 10(b) compares the power consumptions whether our scheduling algorithm is running or not using the two techniques just cited. The power savings with TX-mode are larger than with PSM since the stations avoid periodically turning on the radio to listen to the AP messages. In fact, while PSM saves only 6.1%, the TX-power mode achieves about 23% of power savings when 8 nodes are in the network.

3) Application delay: In Figure 10(c) we compare the average application delay when our algorithm is running versus when it is not. Two different slot sizes are used: 0.1s and 0.3s. The traffic rate and the S parameter used are analogous to the experiment in section 4.1. As expected, the choice of the slot size has a large effect on the average packet delay measured. As the slot size increases, the application delay increases accordingly.

4) Traffic prioritization: When a node is not scheduled, it assigns data to different priority queues. Policies such as WFQ and DWRR are applied. Figure 10(d) shows the measured application delay of experiments where packets are assigned a random priority in the range 1-3 and WFQ is applied. As expected, prioritization provides for an easy way to control the application layer delay. In addition, our scheduling algorithm has a lower application layer delay for high priority data in high traffic load. We



(a) Throughput (b) Power consumption

(c) Application delay (d) Prioritization with WFQ

Figure 10. Measurements on throughput, power consumption and delay

obtained similar results using the model described in 3.2. We found that when increasing the total traffic load (by increasing either λ_A of *Nn*) above ~5Mbps, the scheduling algorithm outperform the standard 802.11. Thus, we can confirm that when the network gets overloaded with data our scheduler can not only save a lot of energy, improve throughput, but can also provide a more predictable and lower application layer delay.

3. Conclusions

In this work we present a distributed scheduling algorithm *above* the MAC layer that is the combination of node-level scheduling (that reduce the problem of contention among nodes) and cell-level scheduling (that lessens the interference between cells). It runs in a TDMA scheme *over* the existing network and MAC layers, using off-the-shelf network components. Saving communication power, while achieving an increase in throughput, is a big accomplishment of our proposed algorithm. Simulations show an improvement in average throughput of up to 10.31 %, with maximum power saving of 85.54 %. Simulation results are then validated by measurements taken by running an implementation of the algorithm on a wireless test network. We show that the algorithm manages application layer delay efficiently by using traffic prioritization and is suitable for high traffic load conditions often found in large scale WSNs.

Acknowledgements

Special thanks to the High-Performance Wireless Research and Education Network (HPWREN) for their support. This work has been supported by the HPWREN, the National Science Foundation, (NSF award number 0426879), CNS, and Sun Microsystems.

References

[1] Cisco Aironet 802.11a/b/g CardBus wireless LAN adapter data sheet, http://www.cisco.com/en/US/products/hw/wireless/ps4555/products_data_sheet09186a00801ebc29.html, Cisco Systems.

[2] High Performance Wireless Research and Education Network (HPWREN), http://hpwren.ucsd.edu/

[3] ns-2 network simulator, http://www.isi.edu/nsnam/ns/

[4] G. Ahn, A. T. Campbell, A. Veres, and L. Sun, "SWAN: Service differentiation in stateless wireless ad hoc networks," in IEEE INFOCOM, 2002.

[5] S. Choi, K. Park, and C. Kim, "On the performance characteristics of WLANs: revisited," ACM SIGMETRICS, 2005.

[6]] A. Demers, "Analysis and simulation of a fair queueing algorithm," ACM SIGCOMM, 1989.

[7] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," IEEE INFOCOM, 2003.

[8] B. Hohlt, L. Doherty, and E. Brewer, "Flexible power scheduling for sensor networks," IPSN, 2004

[9] K. Jamieson, "Implementation of a power-saving protocol for ad hoc wireless networks", Master's thesis, MIT, Feb. 2002.

[10] J. Jun, P. Peddabachagari, M. Sichitiu, "Theoretical maximum throughput of IEEE 802.11 and its applications," IEEE International Symposium on Network Computing and Applications, 2003.

[11] D. Lim, J. Shim, T. S. Rosing, and T. Javidi, "Scheduling data delivery in heterogeneous wireless sensor networks," in IEEE International Symposium on Multimedia (ISM 2006), Dec. 2006.

[12] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor, "IEEE 802.11e wireless LAN for quality of service," European Wireless, 2002.

[13] A. Rao and I. Stoica, "An overlay MAC layer for 802.11 networks," ACM MOBICOM, 2005.

[14] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic media access for multirate ad hoc networks," ACM MOBICOM, September, 2002, Atlanta, Georgia.

[15] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed fair scheduling in a wireless LAN", ACM MOBICOM, 2000.

[16] A. Vasan and A.U. Shankar, "An empirical characterization of instantaneous throughput in 802.11b WLANs," Technical report, Department of Computer Science, University of Maryland, 2002.

[17] W. Xiuchao, "Simulate 802.11b channel within ns2", Technical report, School of Computing, National University of Singapore, 2004.

[18] W. Xiuchao and A. L. Ananda, "Link characteristics estimation for IEEE 802.11 DCF based LAN," IEEE LCN , 2004.

[19] Wireless LAN Medium Access Control (MAC) and Phisical Layer (PHY) Specifications, IEEE-SA Standards Board, June 2003

[20] Lichun Bao, J.J. Garcia-Luna-Aceves , "Channel Access Scheduling in Ad Hoc Networks with Unidirectional Links", 2001

[21] Edward D. Lazowska, John Zahorjan, G. Scott Graham, Kenneth C. Sevcik, "Quantitative System Performance , Computer System Analysis Using Queueing Network Models".