A Code-Based Sleep and Wakeup Scheduling Protocol for Low Duty Cycle Sensor Networks

Neeraj Shrestha, Jong Hoon Youn, and Nitin Sharma

***Abstract*—One of the key challenges in wireless sensor networks is to design an energy efficient communication protocol. In this paper, we present an innovative code-based sleep and wake-up scheduling scheme to minimize energy consumption caused by idle listening. The proposed scheme called SWAP divides time into slots of equal length, and at the beginning of each slot a sensor node enters either an active or power-saving state. The slot scheduling at each node is based on a binary vector, which is constructed using the mathematical properties of finite fields. The set of scheduling vectors generated by SWAP distributes active slots of nodes evenly over entire time frame. This distribution of active slots reduces channel contention and allows better channel utilization. The SWAP scheduling scheme also ensures that any pair of neighboring nodes using the proposed scheduling scheme will be able to communicate each other since their active periods overlap at least once within a cycle of the sleep and wake-up slots. We also design a packet prioritization scheme in SWAP to reduce the packet latency of delay sensitive packets. To evaluate the energy efficiency and performance of the proposed mechanism, we have implemented SWAP in TOSSIM, and compared it with a widely used channel access protocol, BMAC with LPL, in terms of packet delivery ratio, power consumption and delay. The results of our study show that SWAP outperforms BMAC in terms of energy consumption while maintaining comparable packet delivery ratio and latency.**

***Index Terms*—Code-based scheduling, wake-up/sleep scheduling, wireless sensor networks, galois field.**

I. Introduction

A Wireless Sensor Network (WSN) consists of small sensing and computation devices working in a group to detect and monitor environmental changes in plains, forests, oceans, etc. WSN devices are limited in their energy, processing, and communication capabilities. Most of the WSN devices powered with small batteries are deployed in the remote area, and it is not easy to recharge or replace battery once they are deployed at those remote areas. Thus, the design and development of low-energy algorithms and protocols are essential for sensor networks. Especially, the energy consumption of wireless exchange of data between nodes strongly dominates other node functions such as sensing and processing [1], and there have been a number of research studies focusing on the development of energy conserving communication schemes in WSNs [2]–[6].

Manuscript received February 5, 2014; revised March 24, 2014. This work was supported in part by the National Science Foundation under grants CNS-1062995.

Neeraj Shrestha and Nitin Sharma were with the University of Nebraska at Omaha, Omaha, NE 68182 USA (e-mail: nshrestha@unomaha.edu, nsharma@unomaha.edu).

Jong Hoon Youn is with University of Nebraska at Omaha, Omaha, NE 68182 USA (e-mail: jyoun@unomaha.edu).

Many energy-conserving communication protocols in the literature toggle between active and sleep modes in order to reduce energy consumption caused by idle listening. They usually implement either periodic sleep and wake-up cycles or design an asynchronous mechanism that synchronize the wake-up period of a sender and receiver pair [3]. One of well-known schemes based on an asynchronous power conserving approach is BMAC [5], which uses asynchronous sleep technique called Low Power Listening (LPL). In BMAC/LPL, nodes asynchronously but periodically wake up and listen to the channel shortly to see whether there are any activities or not. If no activity is detected, they go back to sleep. In BMAC, a sender wakes up and sends a long preamble before transmitting data packets. The duration of the preamble should be longer than receiver's check interval to capture receiver's attention. The long preamble based approach in BMAC sacrifices sender's energy efficiency for the sake of receiving nodes. These problems can be addressed by developing an efficient sleep and wake-up scheduling method that synchronizes a sender and receiver pair with minimal overhead.

In this paper, we propose a substantially different approach for wake-up scheduling. In the proposed scheme, each sensor node generates a binary scheduling vector based on its node ID, and uses the scheduling vector to determine its sleep and wake-up periods. This approach is independent of network topology and local parameters such as the number of nodes or neighbors. Our approach reduces channel contention significantly by evenly spreading node's active periods over the time. It also minimizes sender's idling energy consumption by allowing a sender to precisely predict the wake-up schedule of the receiving node. The scheduling patterns derived by our method guarantee that the active intervals of any given node always overlap with all neighboring node's active periods within a cycle of sleep and wake-up periods.

The rest of this paper is organized as follows. Section II presents a scheduling vector generation algorithm and the properties of the derived vectors. Section III shows how to use these vectors for sleep and wake-up scheduling and how to exchange messages with neighbors in the proposed scheme. Section IV presents the result of our performance evaluation study using TOSSIM. Finally, we provide a discussion of our conclusions and future works in Section V.

II. Construction of Sleep Scheduling Vectors

This section discusses how a node constructs a binary vector for sleep and wake-up scheduling followed by the discussion about the properties of the constructed scheduling vectors.

DOI: 10.7763/JACN.2014.V2.109

A WSN can be represented as an undirected graph $G=(V, E)$, where V is a set of vertices denoting sensors, and E is a set of edges between vertices. For any two vertices $u, v \in V$, $(u, v) \in E$ if and only if they can hear each other's transmission. In this case, the vertices u and v are *neighbors*. In the proposed scheme, each sensor in V has a Sleep Scheduling Vector (SSV), and uses its SSV to toggle between listen and sleep operative modes. The SSV is defined as follows.

Definition 1: A Sleep Scheduling Vector (SSV) is a binary vector of length n. The SSV of a host $A \in V$, denoted by SSV_A, is $a_1\ a_2\ a_3\ a_4\ ...\ a_{n-2}\ a_{n-1}\ a_n$.

If $a_i = 1$, the i^{th} slot is an active slot

If $a_i = 0$, the i^{th} slot is a power-saving slot, where $1 \leq i \leq n$.

A. Generation of Sleep Schedule Vectors

The generation of SSVs is based on the mathematical properties of finite fields. A Galois field (or finite field), GF (pm) where p is a prime and m is a positive integer, is an algebraic structure with a finite number of elements, where addition, subtraction, multiplication and division are defined [7]. The description of the proposed scheme assumes that the reader is familiar with the fundamental concepts of Galois fields.

Let the elements of GF ($q = p^m$) be denoted by $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{q-2}, \alpha_{q-1}$. First, a node A needs to figure out a polynomial of degree 1 over GF (q) to be used for the construction of the SSV_A. Let the polynomial be $fA(x) = \alpha_i x + \alpha_j$, where α_i and α_j are the elements of GF (q). In order to find a proper i and j, a node performs the following computations. Let s be $NodeIDA \% p^m$, where $NodeIDA$ is a unique node ID of A and % is a modular operator. Then, $i = \lfloor s/p \rfloor$ and $j = s \% p$.

Using the polynomial $f_A(x)$, each sensor generates a SSV as follows:

$$SSV_A = [\phi(f_A(\alpha_0))\ \phi(f_A(\alpha_1))\ \phi(f_A(\alpha_2)) \ldots\ \phi(f_A(\alpha_{q-1})),\ \phi(\alpha_i)],$$

where ϕ is a function which maps a q-ary number to a binary vector of length q:

$\phi(\alpha_0)$ =1000 … 00

$\phi(\alpha_1)$ =0100 … 00

$\phi(\alpha_2)$ =0010 … 00

$\phi(\alpha_{q-1})$ =0000 … 01

While evaluating $f_A(x)$, note that the multiplication and addition operations must be done in GF (q).

```
// Let the elements of GF (q= p^m) be α0, α1, α2, …, αq-2, αq-1.
GENERATE_VECTORS_GF(p, m)
1  q ← p^m
2  for i ← 0 to q - 1
3    for j ← 0 to q - 1
4      for k ← 0 to q
5        if ( k = q )
6          then fx← αi
7          else fx ← (αi * αk) + αj   // * and + are over GF(p^m)
8        for l ← 0 to q
9          if ( fx = αl )
10           then print 1
11           else print 0
```

Fig. 1. An algorithm for generation of SSVs over GF (p^m).

If a node in the network has enough free memory space, the whole set of SSVs can be constructed in advance and stored in its memory before deployment. The pseudocode shown in Fig. 1 takes p and m as inputs and produces a whole set of scheduling vectors. A set of SSVs generated by this algorithm with p=2 and m=2 is shown in Table I. Note that it is not necessary for a node to generate all SSVs. Since a node only needs to know the SSVs of its neighbors, it can selectively generate a subset of SSVs as needed. In order to generate the SSVs of neighboring nodes, a node only needs to know neighbor's node IDs. The node ID will be used to find a proper i and j using modular operations.

TABLE I: SLEEP SCHEDULING VECTORS WITH P=2 AND M=2

i	j	$f(x)$	$\phi(f(\alpha_0))$	$\phi(f(\alpha_1))$	$\phi(f(\alpha_2))$	$\phi(f(\alpha_3))$	$\phi(\alpha_i)$
			k=0	k=1	k=2	k=3	k=i
0	0	$\alpha_0 x+\alpha_0$	1000	1000	1000	1000	1000
0	1	$\alpha_0 x+\alpha_1$	0100	0100	0100	0100	1000
0	2	$\alpha_0 x+\alpha_2$	0010	0010	0010	0010	1000
0	3	$\alpha_0 x+\alpha_3$	0001	0001	0001	0001	1000
1	0	$\alpha_1 x+\alpha_0$	1000	0100	0010	0001	0100
1	1	$\alpha_1 x+\alpha_1$	0100	1000	0001	0010	0100
1	2	$\alpha_1 x+\alpha_2$	0010	0001	1000	0100	0100
1	3	$\alpha_1 x+\alpha_3$	0001	0010	0100	1000	0100
2	0	$\alpha_2 x+\alpha_0$	1000	0010	0001	0100	0010
2	1	$\alpha_2 x+\alpha_1$	0100	0001	0010	1000	0010
2	2	$\alpha_2 x+\alpha_2$	0010	1000	0100	0001	0010
2	3	$\alpha_2 x+\alpha_3$	0001	0100	1000	0010	0010
3	0	$\alpha_3 x+\alpha_0$	1000	0001	0100	0010	0001
3	1	$\alpha_3 x+\alpha_1$	0100	0010	1000	0001	0001
3	2	$\alpha_3 x+\alpha_2$	0010	0100	0001	1000	0001
3	3	$\alpha_3 x+\alpha_3$	0001	1000	0010	0100	0001

B. Properties of Binary Scheduling Vectors

In this subsection, we discuss the properties of SSVs generated by the algorithm shown in Fig. 1. The properties of SSVs are mostly depends upon the inputs parameters p and m. The total number of SSVs is $p^m \times p^m = p^{2m}$, and the length of a SSV is $p^m \times (p^m + 1)$. For example, the SSVs shown in Table I are based on GF(2^2), and have 16 SSVs and the length of the SSVs is $2^2 \times (2^2+1) = 4 \times 5 = 20$.

In order to discuss other properties of SSVs, let us define a few terms first.

Definition 2: The *weight* of a binary vector $X = (x_1, x_2, ..., x_n)$ where $x_i \in \{0,1\}$, denoted by $\omega(X)$, is the number of 1's in X.

Definition 3: The duty cycle is a ratio of active time to the total cycle time. The duty cycle of a node A where $SSV_A = a_1\ a_2\ a_3\ a_4\ ...\ a_{n-2}\ a_{n-1}\ a_n$ is $\omega(SSV_A) / n$.

Note that $\omega(SSV_A)$ is $p^m + 1$ since, at line 4 in Fig. 1, k goes from 0 to q, and for each k, there is exactly one 1. Thus the duty cycle of SSV_A for all $A \in$ V is $(p^m + 1)/ (p^m * (p^m + 1)) = 1/ p^m$. For example, the duty cycle of the SSVs in Table I is 5/20 = 1/4 = 25%.

Definition 4: For two scheduling vectors, $SSV_A = a_1\ a_2\ a_3\ a_4 ...\ a_{n-2}\ a_{n-1}\ a_n$ and $SSV_B = b_1\ b_2\ b_3\ b_4\ ...\ b_{n-2}\ b_{n-1}\ b_n$, a column c in which both a_c and b_c are 1, for $1 \leq c \leq n$, is called a common active slot (CAS) of SSV_A and SSV_B.

One of the key characteristics of SSVs generated by the proposed scheme is that, for any given pair of SSV_A and SSV_B, there is a column i where $a_i, = b_i = 1$. This makes sure that any pair of nodes using a vector constructed by SWAP will be able to communicate each other since their active

periods overlap within a duration of the sleep and wake-up cycle. The flowing lemma and theorem prove this property.

Lemma 1. If $f(x)$ is a non-zero polynomial of degree d over any Galois field F, then the equation $f(x) = 0$ has exactly d solutions (zeros) over F.

Proof: in [7]

Theorem 1. For any given pair of SSVs constructed by the algorithm in Fig. 1, there exists at least one CAS.

Proof: Let us assume that the polynomial $f_A(x) = \alpha_i x + \alpha_j$ and $f_B(x) = \beta_i x + \beta_j$ where α_i, α_j, β_i and β_j are the element of GF(p^m), are used to construct SSV_A and SSV_B respectively. In order to have an overlap of their active slots, $f_A(x_0) = f_B(x_0)$ for some $x_0 \in GF(q)$. Thus, in order to find x_0, we can find the solutions of the difference polynomial $f_A(x_0) - f_B(x_0) = 0$. By Lemma 1, the number of overlaps of any two SSVs is exactly the same as the degree of their difference polynomial. If $\alpha_i \neq \beta_i$, the degree of the difference polynomial is 1, and so there will be exactly one CAS. If $\alpha_i = \beta_i$, at least the last active slot of SSV_A and SSV_B must overlap since $\phi(\alpha_i) = \phi(\beta_i)$. Therefore, there exists at least one CAS for any pair of SSVs generated by the proposed algorithm.

Another important property of the set of SSVs generated is that there is exactly p^m 1's at any single column in the set of SSVs. For example, in the SSVs in Table I, there are only four 1's in any given column. This is an attractive property of the scheduling vectors since it guarantees that there will be exactly 4 active nodes in the networks at any given time. This attribute of the constructed scheduling vectors reduces the number of collisions in network considerably by distributing active nodes over the time period.

III. Pattern Based Sleep and Wakeup Scheduling Protocol

In this section, we discuss how a node communicates with power saving neighbors using the SSVs constructed in the previous section. The proposed protocol called a Sleep and WAke-up Protocol (SWAP) is designed for a wireless sensor networks with particularly low duty cycle.

A. Assumptions

Each node uses an omni-directional antenna for communication, and the network operates in half-duplex mode, which means that a host cannot transmit and receive simultaneously.

We assume that all sensors are synchronized in time. Synchronization can be achieved by using a Global Positioning System (GPS) card with up to 100 ns resolution [8], or a global time synchronization protocol in the literature [9]–[11]. We also assume that a node in the network has a unique ID, and is randomly deployed in the sensing field.

B. Packet Sending and Receiving

After constructing a SSV, a node schedules its sleep and active periods based on the SSV. If $a_i = 1$, the i^{th} slot is an active slot where the node must stay awake. If $a_i = 0$, the i^{th} slot is a power-saving slot where the node can sleep to conserve energy if there is no pending urgent packet to be transmitted (see Fig. 2).

The node follows the assigned SSV from left to right and cycles through it when it reaches at the end of the SSV. A complete sleep-active cycle is called a frame. A data packet will be transmitted based on the sender's and receiver's SSV. Therefore, two nodes can communicate only during their CAS in which their active periods overlap. For example as shown in Fig. 3, both Node 1 and Node 16 become active at the beginning of the 5th slot, and they can exchange a message in this slot.

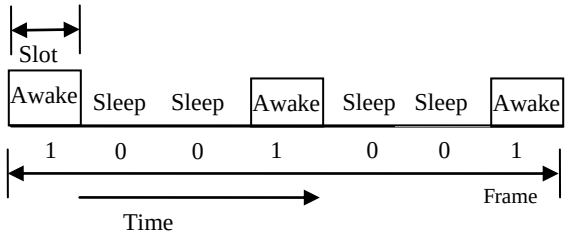

Fig. 2. Sleep and active periods in a frame.

If a node has any pending packet to be transmitted, it first waits for its active slot. After being active, it scans the destination node ID of the outgoing packets, and checks whether there is any active destination nodes at the current slot. If so, it transmits a packet to one of the active neighbors.

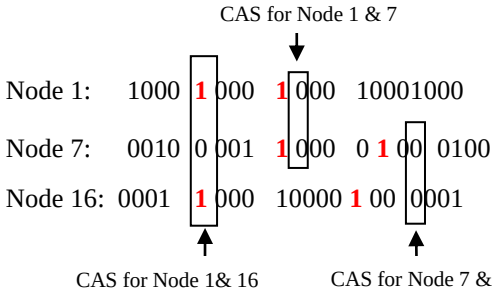

Fig. 3. Send/receive in common active slot.

C. Buffering and Priority Packets

The latency of the original SWAP protocol is much higher than BMAC/LPL. This is because the SWAP source nodes send packets only at one of CASs where both sender and receiver are active. Thus, the packet should be waited in the SWAP transmission queue longer. To address the latency issue, we introduce a data prioritization scheme. The delay sensitive data packets and urgent packets are marked with higher priority at the application layer. The data packets along with their priority level are received by the SWAP scheduling layer. The priority packets can be transmitted during any active slots of the destination node. In other words, the high priority packets are transmitted as soon as the destination node becomes active. In this way, the delay of high priority packets can be significantly deduced.

Suppose Node 1 has high priority data packet to transmit to Node 2 then, Node 1 transmits the packet as soon as Node 2 becomes active regardless of its own state. For example in pattern shown in Fig. 3, if Node 7 has a normal packet destined to Node 16, then it has to wait till the fourteenth slot for both nodes to get active. But in case of high priority packets Node 7 can transmit packet to Node 16 at any active slots of Node 16; Node 16 is active in the fourth, fifth, ninth, fourteenth and twentieth slots, and so Node 7 can use any nearest active slot among them to send high priority packets to Node 16. In this way, high priority packets will experience

less buffering time than normal packets.

D. MAC layer

The proposed scheduling scheme is independent from a MAC layer. In other words, the SWAP layer can be implemented over any MAC protocol with minor adjustments. The MAC protocol selected for our simulation study is a primitive version of the Carrier Sense Multiple Access (CSMA) protocol to optimize the operations. We customize some CSMA parameters such as the number of retransmissions, preamble length, and lower and upper bound of the initial back-off range.

IV. Performance Evaluation

We have implemented the SWAP scheme in a TOSSIM-2 simulation environment [12], [13], and compared our protocol with BMAC-LPL [5]. The current version of TOSSIM does not support the CC2420 radio stack used by the LPL library. We have modified the default BMAC-LPL implementation of MICAz and simulated the protocol in TOSSIM [13]. We have added an interface to simulate the radio sleep and active modes for the CC2420 radio stack. In order to measure energy consumption, we have also added several counters to measure the number of active slots and sleep slots as well as the number of transmitted and received packets over the duration of each simulation.

The experiments are carried out with a randomly deployed network where six hundred nodes are randomly scattered in a randomly generated topology. In the simulation scenarios, we boot all nodes at the same time, so that they are initially synchronized. A SSV is assigned to each node based on its node ID. Packets are generated locally at each node with a random inter-arrival time. The default queue and buffer size is 10 for all the experiments. The duration of a slot in the scheduling pattern is 16 milliseconds.

A. Simulation Scenario

In the first set of simulation, we compare the energy consumption and packet delivery ratio of SWAP with BMAC/LPL protocol with varying pattern (duty cycle). To measure the energy consumption of a single node we take the average of total energy consumed by the network. The energy consumption is measured in millijoules (mJ), and the amount of energy consumed is measured for a simulation runtime, which is calculated in milliseconds (ms). Packet delivery ratio is the ratio of the number of data packets successfully delivered to the recipients to the number of data packets originated from the sources. The corresponding duty cycle in percentage for different patterns in SWAP is given in Table II. For this set of experiment, 150 nodes are randomly chosen among 600 nodes, and they generate packets to random destinations. Application data packets are generated every $r \times 512$ms, where r is a random number ranging from 1 to 60.

TABLE II: Pattern Duty Cycle

SWAP Patterns: GF(p^m)	Duty Cycle (%)
GF(3)	33.33
GF(5)	20.00
GF(7)	14.28
GF(11)	9.09
GF(16)	6.25
GF(23)	4.34

B. Energy Consumption

Fig. 4 shows the average energy consumption per node in SWAP and BMAC/LPL. As we expected, SWAP performs much better than BMAC/LPL since SWAP handles the idle listening more efficiently by predicting the receiver's wake-up slots. Furthermore, in SWAP, the active slots spread more evenly over the time, and so SWAP reduces the network contention. However, a BMAC/LPL sender wastes energy to transmit a long preamble before data exchanges to capture its target node. At a high duty cycle (smaller pattern), BMAC/LPL has very high energy consumption compared to SWAP. This is because, in BMAC/LPL, all non-target receivers within the transmission range of the sender also wake up during the long preamble and listen to the channel till the end of the preamble.

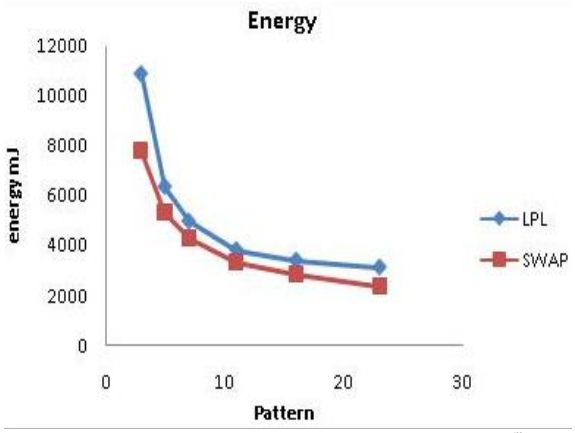

Fig. 4. Energy consumption with varying pattern (GF(p^m)).

C. Packet Delivery Ratio

Fig. 5 shows that the impact of varying duty cycle on the packet delivery ratio is not significant for both SWAP and BMAC/LPL. The packet delivery ratio remains comparable with varying scheduling patterns. According to the results of our experimental study, the packet delivery ratio drops as the traffic density increases. This is because, in case of high traffic load, there is a greater chance for collisions to occur. This could raise the chance for packet drop.

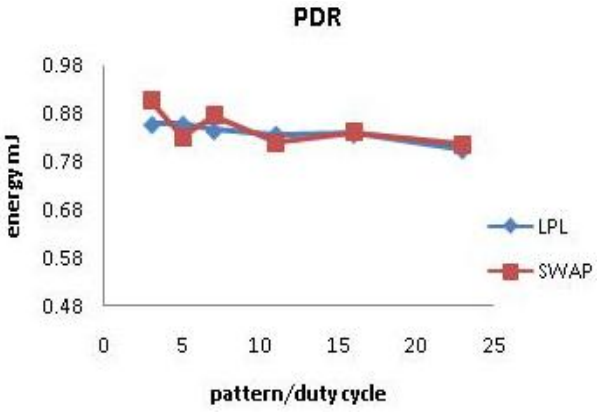

Fig. 5. Paceket Delivery Ratio with varying pattern (GF(p^m)).

D. Latency

In this study, we assume that 10 percent of SWAP packets are high priority and measure the latency of the high priority packets separately. As you can see in Fig. 6, the delay of

prioritized packets (SWAP_PRI) are comparable with that of BMAC/LPL, although the delay for normal packets (SWAP) is much higher than BMAC/LPL. This is because the SWAP sender can transmit high priority packets in any slot where the destination is active regardless of its own slot state. Based on the result of this simulation scenario, we conclude that SWAP can deliver high-priority packets as fast as BMAC.

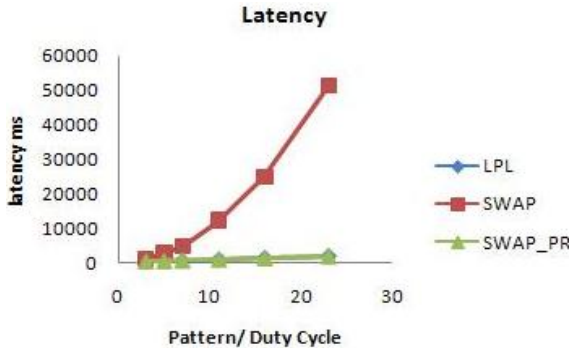

Fig. 6. Latency with varying pattern.

V. Conclusion and Future Work

This paper presents a pattern based sleep scheduling scheme for low-duty cycle wireless sensor networks. Using the property of Galois field, we propose an algorithm to generate a set of binary scheduling vectors with a wide range of duty cycles. These vectors are used to toggle radio modules between sleep and active modes to reduce idle listening energy waste. One of potential issues of our scheme is the latency of packet delivery in a low duty cycle sensor networks. In the proposed scheme, a sender should defer the transmission of a packet until the beginning of a common active slot between itself and the target receiver, the latency of the packet gets accumulated at each hop along the path between the source and receiver. In order to address the latency issue of delay sensitive packets, the SWAP scheme can be integrated with the proposed prioritization method which can guarantee faster delivery for high priority packets.

The simulation results show that our scheme performs significantly better than BMAC/LPL in terms of energy efficiency while maintaining packet delivery ratio at least comparable to BMAC/LPL. Our experimental results also show that the SWAP scheme in conjunction with the proposed prioritization method can provide reliable communication with acceptable latency for high priority packets while conserving energy for low-power sensor nodes in a low-duty sensor network. As a future work, we are planning to extend the proposed scheduling scheme to asynchronous sensor networks.

References

[1] K. Chakrabarty and S. S. Iyengar, *Scalable Infrastructure for Distributed Sensor Networks*, Springer, 2005.
[2] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," *ACM Embedded Networked Sensor Systems*, pp. 303–320, 2007.
[3] G. Halkes, T. V. Dam, and K. Langendoen. "Comparing energy-saving mac protocols for wireless sensor networks," *ACM Mobile Networks and Apps*, vol. 10, no. 5, pp. 783–791, 2005.
[4] B. Luis, O. Rodolfo, P. Miguel, M. Mario, and P. Paulo, "A wireless sensor mac protocol for bursty data traffic," *IEEE Personal, Indoor and Mobile Radio Communications*, pp. 1-5, 2007.
[5] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," *ACM Embedded Networked Sensor Systems*, pp. 95–107, 2004.
[6] W. Ye, J. Heidemann, and D. Estrin, "An energy efficient Mac protocol for wireless sensor networks," *IEEE Computer and Communications Societies*, pp. 1567–1576, 2002.
[7] P. B. Garrett, *Abstract Algebra*, Chapman & Hall/CRC, 2007.
[8] B. W. Parkingson and J. J. Spilker, *Global Positioning System: Theory and Applications* vol. 1, Washington, DC, American Institute of Aeronautics and Astronautics, 1996.
[9] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482-1493, October 1991.
[10] K. Romer, P. Blum, and L. Meier, *Time Synchronization and Calibration in Wireless Sensor Networks*, October, 2005.
[11] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. IEEE Wireless Communications and Networking Conference*, 2003, pp. 1266–1273.
[12] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," *ACM Embedded Networked Sensor Systems*, pp. 126-137, 2003.
[13] TinyOS 2.x documentation for TOSSIM. [Online]. Available: http://docs.tinyos.net/index.php/TOSSIM

Neeraj Shrestha was a graduate student at the Department of Computer Science at the University of Nebraska, Omaha. He received his master's degree in computer science from the University of Nebraska - Omaha in 2010. His research interests include wireless communications, media access control protocols, wireless sensor networks, and energy-efficient communication.

Jong Hoon Youn is an associate progessor at the Department of Computer Science, University of Nebraksa, Omaha. He received the BS degree in computer science from Kyunggi University, Rep. of Korea in 1997, and the MS and PhD degrees in computer science from Oregon State University, United States in 1999 and 2002 respectively. His primary research interests are wireless communication and mobile computing. He has published more than 50 papers in technical journals and conferences.

Nitin Sharma was a graduate student at the Department of Computer Science at the University of Nebraska, Omaha. He received his master's degree in computer science from the University of Nebraska, Omaha in 2010. He is now working at Network Generation Networking as a network programmer in New York, United States.