Implementation of the Wireless Autonomous Spanning Tree Protocol on Mote-Class Devices

Kamini Garg, Daniele Puccinelli, and Silvia Giordano
Networking Lab
University of Applied Sciences of Southern Switzerland
CH-6928 Manno
Email:{kamini.garg,daniele.puccinelli,silvia.giordano}@supsi.ch

***Abstract*—The Wireless Autonomous Spanning Tree Protocol combines medium access control and routing to streamline energy-efficient communication in Wireless Body Area Networks. As these networks generally contain low-end resource-constrained devices, a thorough evaluation on real hardware is essential to the validation of any protocol. We present our implementation of the Wireless Autonomous Spanning Tree Protocol on mote-class devices, and highlight the challenges of taming the vagaries of low-power wireless that do not arise in simulation-based evaluations. We study the performance of the protocol in several dimensions, with a special emphasis on energy-efficiency. Our comprehensive set of experimental results indicates that the protocol can achieve low duty cycles if used jointly with a low-power link layer.**

***Keywords*-Wireless Body Area Networks; Wireless Autonomous Spanning Tree Protocol; Low Power Listening, Medium Access; Routing**

I. Introduction

A Wireless Body Area Network (WBAN) is a sensor network whose nodes are either attached or implanted into the human body. As for sensor networks in general, energy-efficiency is of paramount importance for WBANs. Because the radio notoriously accounts for the lion's share of the overall energy consumption, WBAN protocols must streamline communication. Due to their energy constraints, WBAN nodes are forced to employ low-power radios whose limited transmit power often precludes the option of forming a simple star network topology and require multihop communication as a matter of course.

In this context, channel access and routing decisions cannot be made in a vacuum and must account for the conditions of the wireless medium. While most existing solutions address the MAC and the network layer separately, the Wireless Autonomous Spanning Tree Protocol (WASP) [1] offers a unified framework for the coordination of medium access and multihop routing tailored to the relatively small network size of WBANs. With the WASP, WBAN nodes self-organize in a tree topology where parents set up a medium access schedule for children. In the original WASP work, the existence of a static tree topology is taken for granted, and the protocol is mainly evaluated through simulation. In practice, however, static connectivity cannot be expected in low-power wireless networks, because time-varying link dynamics affect even networks of stationary nodes. In a WBAN, nodes are generally stationary with respect to one another, but the network as a whole moves with the person and its link dynamics are affected accordingly. In this paper we tackle the challenging task of taming the vagaries of low-power wireless to implement the WASP on mote-class devices and evaluate it experimentally in various dimensions. Because the paramount goal of the WASP is energy-efficiency, we use the duty-cycle of our nodes as our primary figure of merit, and investigate the interplay of the WASP with the standard link-layer duty-cycling technique known as Low Power Listening (LPL) [2].

II. Related Work

In the sensor network literature, communication protocols can be mainly categorized as contention-based protocols, such as B-MAC [2], Wise-MAC [3], and X-MAC [4], and slotted protocols, such as S-MAC [5]. B-MAC is a CSMA-based technique that leverages asynchronous LPL, the standard technique for link-layer duty-cycling that enables nodes to periodically put their radios into sleep mode while maintaining the illusion of an always-on link. LPL dampens the idle listening problem by shifting the energy cost of communication from the receiver to the transmitter, which needs to match the preamble of its outgoing packets to the sleep interval of the receiver (long preamble). Wise-MAC gets transmitters to learn the wake-up schedule of their intended receivers and shortens LPL's long preamble through synchronization, while X-MAC sticks to asynchronous LPL but reduces the energy impact of the preamble. S-MAC uses a periodic listen/sleep cycle and ensures the schedule synchronization of neighboring nodes. Tree-based collection routing is a basic primitive for sensor networks employed by several protocols, such as MintRoute [6], the Collection Tree Protocol (CTP) [7], and Arbutus [8]. Cross-layer protocols that merge medium access and routing have been proposed to meet the specific needs of WBANs, whose network size is typically rather small (less than 20 nodes). The Wireless Autonomous Spanning Tree Protocol (WASP) [1] combines slotted medium access control and tree-based col-

 ISBN: 978-1-61208-103-8

lection routing to streamline energy-efficient communication in WBANs.

III. Protocol Description

A. Overview of the WASP Protocol

As described in [1], the WASP protocol is a slotted cross-layer protocol that uses a multi-level spanning tree for the coordination of medium access and multihop routing. The WASP employs a slotted notion of time, which is viewed as a succession of WASP-cycles (sets of time-slots). Moreover, the WASP presupposes the pre-existence of a spanning tree rooted at the sink. It further assumes that every node in the tree has exactly one parent as well as a complete knowledge of its neighborhood (*i.e.*, its parent, its siblings, and its children), and that the sink has a complete knowledge of the whole tree. Each node provides channel access information to its children by broadcasting a node-specific message called WASP-scheme. The WASP-scheme serves to regulate medium access from parent to children: a parent uses a WASP-scheme to tell its children when to access the medium and when to sleep. The sink initiates the process by broadcasting its own WASP-scheme. Upon reception of the WASP-scheme from the sink, its children broadcast their own WASP-scheme, which they derive from the sink‘s WASP-scheme. This process continues until all nodes in the tree have learned the correct timing for channel access using WASP-schemes. A node whose many children cannot be accommodated within the medium access time-slots allocated through its parent's WASP-scheme can use its own WASP-scheme to request additional time-slots for its children to use in future WASP-cycles.

Table I
Format of the WASP-Scheme for the Sink

SinkID	ChildIDs	SP	TFS	CS

As shown in Table I, the format of the sink's WASP-scheme comprises the SinkID, the ChildIDs, the Silent Period (SP), the Total number of Forwarding Slots (TFS), and the Contention Slot (CS). The SinkID and ChildIDs are the addresses assigned, respectively, to the sink and its child nodes. The WASP-scheme of the sink indicates the timeline of one WASP-cycle that includes a slot for the sink to broadcast its WASP-scheme, a slot for each of its children to send out their own WASP-schemes, a radio sleep mode period (whose slot count is the SP), a period during which the sink receives data forwarded by its children (the forwarding slots, whose total count is the TFS), and a special slot, the CS, in which new nodes may join the network using CSMA-CA. Let S denote the sink and Λ_m denote the m^{th} level in the spanning tree, with $\Lambda_0 \triangleq \{S\}$. Also, let T_i denote the set of nodes in the subtree of node i. The SP of the sink can be computed as

$$\mathrm{SP}_S = \max_{i \in \Lambda_1} |T_i|. \quad (1)$$

The TFS needed by the sink is given by the sum of all forwarding slots required by each node in Λ_1. During the forwarding slots, the nodes in Λ_1 forward their received data to the sink, and the number of forwarding slots required by each node in Λ_1 is equal to the total number of nodes in its sub-tree.

Table II
Format of the WASP-Scheme for a Node in Λ_m ($m \geq 1$)

NodeID	SP	ChildIDs	TFS	CS	DATA

Every node derives its own WASP-scheme based on the parent's WASP-scheme and therefore learns about its role in each time-slot of the WASP-cycle. While a sink's WASP-scheme is a dedicated control packet, the WASP-schemes of all other nodes may include data. As shown in Table II, the format of the WASP-scheme for nodes other than the sink contains the NodeID (the address of the node), the SP, the ChildIDs, the TFS, the CS, and the data itself. For any node other than the sink, the SP is used to tell its children in which time-slots they can go into sleep mode. As explained in [1], for a node $j \in \Lambda_1$, the length of the SP is equal to the slot number of the start of the SP of the sink S minus the slot number of its first occurrence in the WASP-scheme of S, minus 1. For a node $i \in \Lambda_m$ ($m > 1$), the length of the SP is equal to the slot number of the CS minus the slot number of its first occurrence in its parent's WASP-scheme. The number of forwarding slots for each node can be computed based on the requirements of the children and may vary accordingly over different WASP-cycles. For any node in any particular WASP-cycle, the TFS is given by the sum of the data packets received from its children. The length of a WASP-cycle depends on the length of the sink's WASP-scheme. The total number of WASP-cycles needed to send data from all the nodes to the sink depends on the depth the spanning tree. Data up to Λ_2 can be sent only in one WASP-cycle while for the further level nodes more WASP-cycles are required.

The original work on WASP uses analysis and simulation to evaluate the protocol, and takes for granted the tree formation process as well as the distribution of connectivity information across the tree. In practice, however, tree topologies [7][8] are never static and are continuously subjected to real-life link dynamics: parents, siblings, and children may and do change. To enable an implementation of WASP on Berkeley motes, we approximate the static tree that WASP presupposes by constructing a stable tree, *i.e.*, a tree that is solely constituted by links with high noise margins, or, equivalently, a high Received Signal Strength (RSS). We

 ISBN: 978-1-61208-103-8

will refer to links whose RSS exceeds a cutoff threshold Θ as highly reliable links. As long as the noise margins of its links are sufficiently high to withstand the link dynamics, our empirical evidence indicates that a stable tree can be expected to remain static with high probability.

B. Generation of a Stable Tree

We obtain a robust connectivity graph by blacklisting all links other than the highly reliable ones. A stable tree is obtained by using a randomized subset of the links in the robust connectivity graph. The stable tree formation process initiates from the sink and continues across the network until all the nodes learn about their local connectivity. In accordance with the WASP's requirements, every node has to have a highly reliable link to its single parent, and the nodes that share a parent are also required to have a highly reliable link to each other. Let $R_{i,j}$ denote the RSS measured at j when i transmits. The connectivity matrix between nodes i and j can be defined as

$$\mathrm{C}_{ij} = \mathbf{1}_{R_{i,j} \geq \Theta}. \quad (2)$$

By way of a sink-initiated connectivity discovery sweep, each node in Λ_m ($m > 0$) selects a unique parent and implicitly assigns itself to a given level Λ_m based on the availability of a highly reliable link to a unique parent and highly reliable links to one or more siblings (nodes that share the same parent). The basic condition for the assignment of a node j (with k as its parent) to level m is

$$\mathrm{C}_{jk}\mathrm{C}_{kj} = \mathbf{1}, k \in \Lambda_{m-1} \quad (3)$$

If multiple nodes satisfy (3), then we examine the cross-links between the nodes to find all pairs (i, j) such that

$$\mathrm{C}_{ji}\mathrm{C}_{ij} = \mathbf{1}, i \in \Lambda_m, j \in \Lambda_m \quad (4)$$

After arbitrarily selecting one pair (i, j) that satisfies (4), we check whether other nodes that satisfy (3) also have highly reliable links to both i and j (according to (4)). All such nodes are added to Λ_m. If no pair (i, j) exists that satisfies (4), one single node that satisfies (3) is selected. Once the stable tree has been set up, every node learns the structure of its subtree and sends it to its parent. Subsequently this structure is propagated until the sink receives it. At the end of this process the sink learns the structure of the whole tree and determines the number of slots in its SP in its own WASP-scheme accordingly.

Depending on the connectivity properties of the network, it may not be possible for the stable tree to span all the nodes for a given value of the cutoff Θ. Network partitioning is a well-known side effect of blacklisting [9], but it is not likely to occur at the levels of node density that are typical of WBANs.

C. Example

Let us illustrate the tree formation process with a sample network of ten nodes. Nodes are labeled with numbers ranging from 0 to 9, and the sink is node 0. The connectivity matrix of our sample network is displayed in Table III.

In the matrix highly reliable links are represented with a 1 and all other links with a 0. For our sample network the sink has highly reliable links to nodes 1, 2, 3, 4, 5 and 9, as shown in Figure 1. All these nodes satisfy (3) and are eligible to become children of the sink. We further check the cross-links and find the pairs of nodes satisfying equation (4). For our sample network such pairs are (1,3), (1,4), (1,9), (2,4), (2,5), (3,9), (4,5), (4,9) and (5,9). We arbitrarily choose pair (1,3) and further check for the remaining nodes satisfying (3) and (4). If we consider node 4, we find that it satisfies (3) with the sink as its parent; node 4 also satisfies (4) with node 1 but does not satisfy (4) with node 3, and therefore it is not elected to join the pair (1,3) as a sibling. Likewise, we check for all the eligible nodes and, as an outcome of this process, we find that node 9 is the only remaining node that satisfies both (3) and (4). Therefore nodes 1, 3, and 9 will belong to Λ_1. For our sample network the structure of the stable tree up to level 1 is given in Figure 2.

Table III
CONNECTIVITY MATRIX OF HIGH CONNECTIVITY NODES

NodeID	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	1	1	1	1	1	1	0	0	0	1
1	1	1	0	1	1	0	1	1	1	1
2	1	0	1	0	1	1	0	0	1	0
3	1	1	0	1	0	0	1	0	1	1
4	1	1	1	0	1	1	1	0	0	1
5	0	0	1	0	1	1	0	0	0	1
6	0	1	0	1	0	0	1	0	0	1
7	0	1	0	0	0	0	0	1	1	1
8	1	1	1	1	0	0	1	1	1	1
9	1	1	0	1	1	1	0	1	1	1

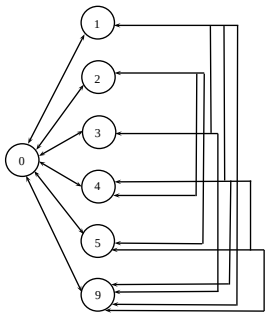

Figure 1 : Connectivity Graph of the Sink's Neighborhood

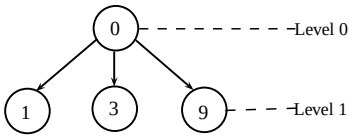

Figure 2: The Stable Tree Formation Up to Level 1

Nodes at Λ_1 start identifying their own children by using the same procedure as the sink. Let us assume that node 1 begins the child formation, followed by node 3 and 9. Highly reliable links to node 1 are 0, 3, 4, 6, 7, 8 and 9. Node 0 is the parent of 2 while nodes 3 and 9 are the siblings. Therefore the nodes that are eligible to become the children of node 1 are 4, 6, 7 and 8. Further we find that only pair (7,8) satisfies (4) and therefore belong to Λ_2. The procedure continues at nodes 3 and 9 and the outcome is shown in Figure 3.

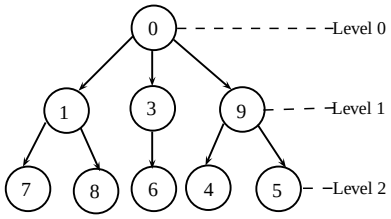

Figure 3 : The Stable Tree Formation Up to Level 2

Afterwards, Λ_2 nodes will also perform the same procedure for the formation of Λ_3. Node 2 will be the only node in Λ_3. For our sample network the final stable tree (comprising of three levels) is displayed in Figure 4.

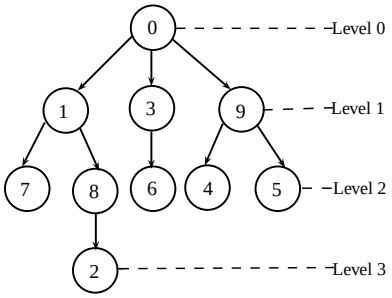

Figure 4 : The Final Stable Tree of Level 3

As soon as the complete tree is built, the sink learns the structure of the whole tree. After the tree formation is complete, the sink constructs its WASP-scheme and broadcasts it. To generate its WASP-scheme, the sink first calculates the SP and the TFS. In our example, $\Lambda_1 = \{1, 3, 9\}$, and from (1) $\mathrm{SP}_S = 4$. The TFS of the sink is given by the sum of all the forwarding slots needed by each child in Λ_1 and is therefore equal to 6. Table IV shows the SP and TFS for each node in two WASP-cycles.

Table IV
SP AND TFS VALUES IN TWO WASP-CYCLES

	WASP-cycle 1		WASP-cycle 2	
NodeID	**SP**	**TFS**	**SP**	**TFS**
0	4	6	4	6
1	2	0	2	1
2	1	0	1	0
3	1	0	1	0
4	2	0	2	0
5	1	0	1	0
6	1	0	1	0
7	2	0	3	0
8	1	0	2	0
9	0	0	0	0

IV. PROTOCOL IMPLEMENTATION AND EXPERIMENTAL RESULTS

We have implemented the WASP using MICAz motes and the TinyOS operating system. MICAz is a widely used research platform built around the CC2420 transceiver, which employs the 802.15.4 physical layer. We evaluated our implementation on an indoor testbed of ten MICAz motes. The testbed consists of a sink acting as the coordinator and nine other nodes that inject their data into the network so that it can be delivered to the sink. The sink in turn forwards everything to a base station node connected to a Crossbow MIB600 gateway that exports the data for offline processing. The 5ms slot length used as a simulation parameter in [1] is simply not workable with our hardware. To simplify the implementation we relax the slot length to one second, thereby eliminating the need for a tight time synchronization technique. Our own experimental results suggest that the RSS threshold $\Theta = -60$dBm is more than sufficient for the link dynamics that are typical of WBANs (in general, it is a rather conservative calibration).

Since the ultimate goal of WASP is energy efficiency, a paramount figure of merit is the duty-cycle, which we measure with online software estimation [10]. In our implementation, we compare WASP's own duty-cycling and the joint action of WASP's duty-cycling with LPL [2]. We use the standard TinyOS implementation of LPL, integrated in a link layer called BoX-MAC [11].

Each of our experiments was run for the duration of one hour. We explored two different LPL settings and ran experiments with sleep intervals of 100ms and 150ms for each node. In general, the LPL sleep interval must be significantly smaller than the duration of a WASP slot. Figure 5(a), 5(b) and 5(c) show the stable tree obtained by applying our tree formation algorithm in three different experiments (respectively with no LPL, with LPL at 100ms, and with LPL at 150ms). Although we employed a similar network setup in all experiments, the tree formation process (selection of a subset of the highly reliable links in the robust communication graph) is randomized, leaving us with

 ISBN: 978-1-61208-103-8

no control over the specific tree layout for each individual experiment. In general, because of the requirements that the WASP imposes on the tree topology, routes may be set up using more hops than needed based on connectivity; sacrificing a low hop count is certainly a drawback of the WASP approach [12].

In our experiments we measure the duty cycle and the Packet Delivery Ratio (PDR) of each node in the network. The PDR for a given node other than the sink is defined as the total number of data packets delivered to the sink over the total number of data packets injected by the node into the network. The PDR for the sink is the end-to-end delivery ratio computed as the total number of delivered packets over the total number of injected packets by all nodes. We group nodes based on their Λ_m ($m \geq 0$) membership and compare the duty cycle and the PDR of the nodes residing in the same level. We also measure the Control Overhead for the network, defined as the ratio of the number of dedicated control packets over the total number of transmitted packet (control and data packets). Tables V, VI, and VII show the duty cycle and PDR for all the experiments.

In Tables V, VI, and VII, we observe that by imposing LPL on the WASP protocol we obtain a sharp drop in the duty cycle of each node as compared to WASP's own application-based duty cycling. The sharp decrease in the duty cycle allows a drastic reduction of the overall energy consumption needed for network communication and boosts the energy-efficiency of the WASP protocol. We observed from Tables V, VI, and VII that the duty cycle for the sink without using LPL is 61.35%, which drops sharply to 9.48% with a 100ms LPL and 7.51% with a 150ms LPL. The drop in the duty cycle for the sink is around 52% with either LPL setting. Our results show that WASP can peacefully coexist with a low-power link layer and greatly benefit from it, because its baseline duty-cycling is not sufficiently aggressive to ensure significant energy savings. Nodes in Λ_1 are responsible for forwarding the data of the lower levels, and therefore they consume the largest amount of energy as they have to keep their radio on for the longest time. Our results show that both nodes that are very active (like Λ_1 nodes) and nodes that are see relatively little action (like the leaf nodes) can greatly benefit from the joint action of WASP and LPL.

LPL only takes a small toll on the reliability of the protocol. For our experiments the end-to-end PDR is 100% without LPL, 99.83% with 100ms LPL, and 99.82% with 150ms LPL. We obtain a better PDR performance compared to [1] where packet loss rate is 30%, but this is largely a byproduct of our relaxed time-slotting as well as the overly pessimistic channel model employed in the WASP simulations in [1].

We also measure the Control Overhead for the network. In our implementation the control packets are the packets broadcast for the network set up and the control WASP-scheme broadcast by the sink node. The Control Overhead is about 6% in all of our setups, suggesting that neither the tree layout nor the presence/absence of LPL has a considerable impact on the overhead. We display the total number of control and data packets transmitted by the WASP protocol (without LPL) over time in Figure 6.

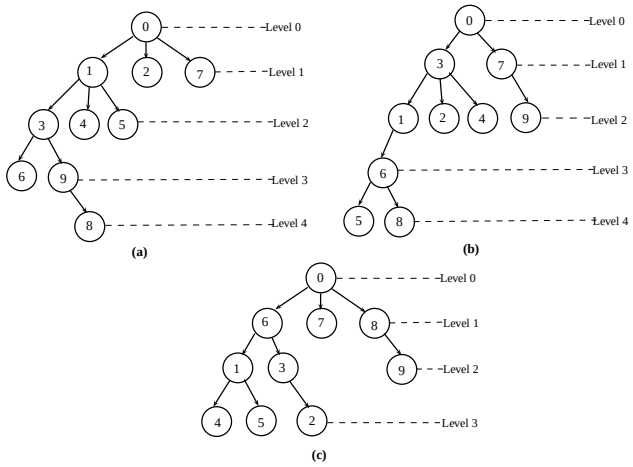

Figure 5: (a) Tree obtained with WASP-No LPL (b) Tree obtained with WASP-LPL-100ms (c) Tree obtained with WASP-LPL-150ms

Table V
DUTY CYCLE AND PDR VALUES FOR THE WASP-NO LPL

Level Λ_m ($m \geq 0$)	NodeID	Duty Cycle (in %)	PDR (in %)
Λ_0	0	61.35	100
Λ_1	1	91.71	100
	2	22.76	100
	7	28.29	100
Λ_2	3	72.15	100
	4	31.04	100
	5	36.54	100
Λ_3	6	42.03	99.49
	9	61.25	99.49
Λ_4	8	86.08	99.49

Table VI
DUTY CYCLE AND PDR VALUES FOR THE WASP-LPL AT 100MS

Level Λ_m ($m \geq 0$)	NodeID	Duty Cycle (in %)	PDR (in %)
Λ_0	0	9.48	99.83
Λ_1	3	15.64	100
	7	14.46	100
Λ_2	1	11.29	100
	2	3.63	100
	4	4.10	100
	9	4.11	99.49
Λ_3	6	10.33	99.66
Λ_4	5	7.91	99.49
Λ_4	8	9.91	99.49

 ISBN: 978-1-61208-103-8

Table VII
DUTY CYCLE AND PDR VALUES FOR THE WASP-LPL AT 150MS

Level Λ_m ($m \geq 0$)	NodeID	Duty Cycle (in %)	PDR (in %)
Λ_0	0	7.51	99.82
Λ_1	6	14.05	100
	7	2.71	100
	8	12.13	100
Λ_2	1	8.80	100
	3	7.38	100
	9	4.49	100
Λ_3	4	4.73	99.47
	5	5.49	100
	2	5.83	99.47

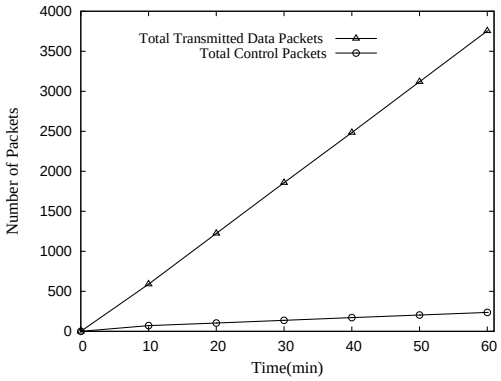

Figure 6: Total Control and Transmitted Data Packets for the WASP over Time

V. CONCLUSIONS

We successfully implemented the Wireless Autonomous Spanning Tree Protocol (WASP) on mote-class devices and evaluated its performance on 10-node testbed. We chose the WASP because of its promising cross-layer design approach that combines and coordinates medium access and multihop routing. The WASP presupposes the existence of a static tree structure to be superimposed to the network, but in practice low-power wireless connectivity is highly dynamic, even for networks of (almost) stationary nodes like WBANs. To obtain a stable tree topology out of inherently unstable low-power links, we adopted a blacklisting-based approach to build a tree of highly reliable links. While this approach is indispensable to approximate the static tree topology that the WASP presupposes, in practice it may lead to routes consisting of too many short hops, with the consequent loss of the benefits of long-hop routing [12]. The approximation of static connectivity also has scalability issues, and the lack of overlap between logical and physical (broadcast) connectivity may thwart WASP's slotted medium access with omnipresent hidden node effects. From the standpoint of energy-efficiency, while WASP's baseline duty-cycling is insufficient to meet the lifetime demands of WBANs, our results show that the WASP can be effectively complemented by a standard link-layer duty-cycling technique, which reduces the mean node duty-cycle from over 50% to about 8%, and the standard deviation from 25% to 4%.

REFERENCES

[1] B. Braem, B. Latré, I. Moerman, C. Blondia, and P. Demeester. The Wireless Autonomous Spanning tree Protocol for multi hop wireless body area networks. In *The 3rd Ann. Int. Conf. on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS'06)*, San Jose, CA, USA, July 2006.

[2] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Baltimore, MD, USA, November 2004.

[3] A. El-Hoiydi and J.-D. Decotignie. Wisemac, An ultra low power MAC protocol for multi-hop wireless sensor networks. In *First Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS'04)*, Turku, Finland, July 2004.

[4] M. Buettner, G. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *4th ACM Conference on Embedded Networked Sensor Systems (SenSys'06)*, Boulder, CO, USA, November 2006.

[5] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *21th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, pages 1567–1576, New York, NY, USA, June 2002.

[6] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, Los Angeles, CA, November 2003.

[7] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*, Berkeley, CA, November 2009.

[8] D. Puccinelli and M. Haenggi. Reliable Data Delivery in Large-Scale Low-Power Sensor Networks. *ACM Transactions on Sensor Networks*, July. 2010.

[9] O. Gnawali, M. Yarvis, J. Heidemann, and R. Govindan. Interaction of Retransmission, Blacklisting, and Routing Metrics for Reliability in Sensor Network Routing. In *1st IEEE Conf. on Sensor and Ad Hoc Comm. and Networks (SECON'04)*, Santa Clara, CA, USA, October 2004.

[10] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *4th Workshop on Emb. Networked Sensors (Emnets IV)*, Cork, Ireland, June 2007.

[11] D. Moss and P. Levis. BoX-MACs: Exploiting Physical and Link Layer. Technical Report 08-00, Stanford University, 2008.

[12] M. Haenggi and D. Puccinelli. Routing in Ad Hoc Networks: A Case for Long Hops. *IEEE Communications Magazine*, 43:93–101, October 2005.

 ISBN: 978-1-61208-103-8