Techniques for Increasing Network Functionality while Remaining within Legal Maximum TX Duty Cycle Requirements

Eoin O'Connell, Victor Cionca, Brendan O'Flynn
Tyndall National Institute
University College Cork
eoin.oconnell@tyndall.ie
Ireland

***Abstract*—In this paper, a technique is presented which allows users of the license free ISM bands to increase functionality of the wireless network while remaining within the maximum legally allowed transmit duty cycle requirements. We show through analytical modeling and empirical evaluation, that traditional MAC and routing techniques result in a significant increase in the overall TX duty cycle when sensor sample frequencies are increased. Specifically, we focus on the 868MHz ISM band where the maximum allowed legal TX duty cycle must be $< 1\%$. Two simple techniques are presented that significantly reduce the TX duty cycle. The result of this technique is a reduction in power consumption and more importantly, a method to increase network functionality while remaining within the legal requirements. We show that these techniques allow for sensor readings to be collected far more frequently in multi-hop, receiver duty cycled, wireless sensor networks.**

***Keywords*—868MHz limitations, Duty Cycle, Low Power, WSN, Multi-Hop**

I. INTRODUCTION

Considering that the license free bands can be occupied by multiple co-existing wireless networks, transmit power levels and maximum transmit duty cycle restrictions are placed on the users. These sanctions are placed by the ISM band regulatory authorities [1]. Examples of such systems which can be found in residential homes, would be wireless smoke alarm systems and wireless burglar alarm systems. These restrictions are put in place to maximize fairness and reduce the probability of interference/ collisions. Operation in the license free $868-868.6$ MHz ISM band requires maximum TX duty cycles of $<1\%$ (Class 2) and maximum TX power levels of approximately +14dBm. Other nearby bands require $<0.1\%$ (Class 1). This specific duty cycle requirement of $<1\%$ translates to a maximum of 36 seconds of TX activity within one hour and a maximum length of 3.6 seconds for any single transmission. (See Table I).

These restrictions place limitations on the functionality of the network, specifically, they limit the rate at which sensor readings can be reported to the network sink. Increasing the sensor reporting rate increases the TX duty cycle. The rate at which sensor readings must be reported from each node in the network, is a largely application based requirement. Depending on the application, these regulatory restrictions may impose a limit on the desired sensor report rate and the network engineer will need to be satisfied with a lower sampling rate. To overcome the issues described above, two techniques are implemented to reduce the overall TX duty cycle of the multi-hop network as a whole. The first technique and the one that has the largest impact on reducing the TX duty cycle is a neighbor schedule learning system. Using this layer 2 MAC protocol optimization, nodes learn when neighboring nodes are expected to wake up and choose the most energy efficient time to begin contacting them. The result of this is a drastic reduction in TX duty cycle.

TABLE I: Frequency Bands For Non-Specific Short Range Devices in Europe

Frequency Band	Max ERP	TX Duty Cycle	Bandwidth
$868-868.6$MHz	14dBm	$< 1\%$	No Limits
$868.7-869.2$MHz	14dBm	$< 0.1\%$	No Limits

The second technique is a layer 3 routing optimization, whereby nodes piggyback their sensor readings into packets from upstream nodes within the multi-hop network. When a packet is received which requires forwarding towards the network sink, this triggers the recipient of the packet to carry out a sensor reading. This optimization results in fewer transmissions and more optimal utilization of the maximum payload length of the physical layer itself.

A. TX on time during transmissions

The length of time for which the radio is in TX mode during each transmission to a duty cycled neighbor, varies greatly from MAC protocol to protocol. To guarantee reliable communication, non-acknowledge based MAC protocols such as B-MAC and BoX-MAC1 are required to leave the radio transceiver in TX mode for the full duration of the receive check interval (T_W) [8], [6]. The receive check interval is the time interval between receive check operations in duty cycled WSN nodes. Conversely, acknowledge based protocols such as X-MAC and BoX-MAC2 stop transmitting as soon as an ACK is received [2], [6]. On average, this ACK packet will be received half way through the receive check interval. Interestingly, the total length of time for which both of these protocols spend in TX mode during transmissions to duty cycled neighbors, is approximately equal to $\frac{T_W}{4}$. On average, the overall time the radio is active for during transmissions is

 ISBN: 978-1-61208-296-7

indeed $\frac{T_W}{2}$. During this time period of $\frac{T_W}{2}$, the radio spends a share of its time in TX mode sending packets and the rest in RX mode listening for ACK (acknowledge) packets. In the case of BoX-MAC2, the proportions of time spent in RX and TX will vary as the payload length changes. X-MAC on the other hand spends fixed lengths of time in both TX and RX during the contacting phase due to its RTS (Request To Send)/ CTS (Clear To Send) system.

The goal of this work is to provide a simple technique to lower TX duty cycle, allowing users of the license free bands to increase network functionality.In Section II, a detailed description is given into the functionality of both of these techniques. In Section III, modeled results are presented showing the potential savings in TX duty cycle that can be achieved. In Section IV, the experimental setup is explained, empirical test data is presented and compared against our simulated results. In Section IV-D, we discuss these important results and their implications. Finally a brief summary of related work is presented in Section V and in Section VI we conclude and suggest a few topics for further expanding this work.

II. DESIGN

The MAC protocol used for testing is structurally very similar to X-MAC. It consists of an RTS/ CTS preamble wakeup stream which contains source and destination address information. A node which receives an RTS packet addressed to itself, responds with a CTS (containing source and destination address) packet to the sender. Upon reception of the CTS packet at the sender, it proceeds by sending the payload data and waits for a payload acknowledge message. The duration of one send RTS packet and listen for CTS cycle is 2.4ms, one cycle consists of 1.2ms of TX time and 1.2ms of RX time listening for a CTS packet.

A. Reducing TX On Time through Neighbor Schedule Learning

To reduce the energy required to send data packets, a mechanism to learn the wakeup schedules of neighboring nodes is implemented. Learning schedules involves no additional synchronization packets. Each node in the network maintains its own wakeup schedule, and simply guarantees that each periodic receive check will occur at an integer multiple of the network wide receive check interval.

The times at which nodes wake up are random in terms of when the application firmware begins running, but fixed in terms of when they are allowed to perform receive checks. The firmware on each node may begin operating at different times, but the interval between receive checks will remain to be an integer multiple of global receive check period for the duration of the nodes lifetime (excluding drift). Nodes must obey this wakeup schedule independent of their own tasks.

With wakeup scheduling now handled by the aforementioned mechanisms, each node in the network can learn the time offset between its wakeup schedule and those of its neighbors. During the very first interaction between two nodes, the sender stores the number of RTS/CTS cycles it required before the destination responded. This number of *attempts* is stored in a neighbor specific data structure and is proportional to the time offset between their respective wakeup schedules.

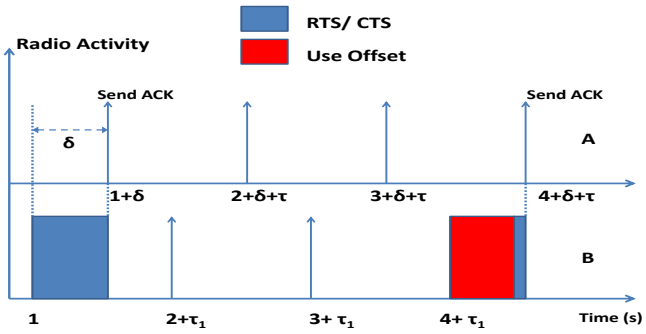


Fig. 1: Vertical arrows indicate when each node wakes at integer multiples of the receive check interval (1 second here). Node B learns the time offset to Node A and applies this learned offset (δ) during next transmission. Graph also illustrates how receive checks will drift (τ,τ_1)

One additional byte is required to store the time offset between wakeup schedules, hence the system is scalable.

During the next transmission to this particular neighboring node, the sender now knows reasonably accurately, when the destination will wakeup. The sender recalls the time offset value from the neighbor specific data structure, and delays accordingly so as to begin performing RTS/CTS cycles a few ms before the destination is expected to wakeup. This time offset is updated during every encounter to account for oscillator drift. The offset learning process is illustrated in Figure 1.In Figure 1, node B first learns the time offset between its wakeup schedule and that of node A, during the next transmission it uses the learned time offset and delays accordingly before contact the destination. In our implementation, the sender begins attempting to contact the destination 9.6ms (T_{SYNC} = 4 RTS/ CTS cycles) before it is expected to wakeup.

B. Piggybacking Data Messages

The primary task of each node in the WSN is to report periodic sensor readings to the network's sink. To reach the network sink, nodes at the outer edges of the deployment may have to route through several nodes, depending on the density and RF environment of the deployment.

The underlying idea of our piggybacking optimization is as follows: nodes that happen to lie in a path that has neighboring nodes generating or forwarding data packets will piggyback their sensor readings into messages which are being forwarded. This process is described graphically in Figure 2. Traditionally, this is done differently and each individual node generates its own periodic data messages.

To accommodate piggybacking, the payload is partitioned into different blocks. Each node which forwards the packet, adds its sensor readings to the payload in a specific position. The position is dependent on the hop number. The leaf node that generated the packet, adds its sensed data to position 0. The next node to interact with and forward the packet adds its sensed data to position 1. In our implementation and application, each node adds a total of 5 bytes to the payload and the length of the variable data packet. Additionally, packets

 ISBN: 978-1-61208-296-7

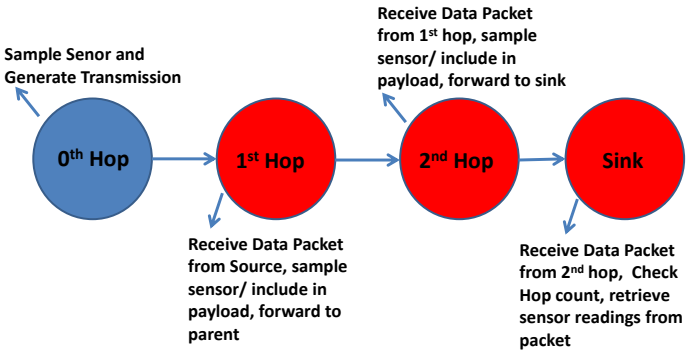


Fig. 2: Data piggybacking. Each node along the path intelligently adds their sensor readings to the payload of incoming messages. Reduces extra wasteful transmissions

contain a routing header. Forwarding nodes increment a *Hop Count* value, and add the ID and RSSI of the last hop to the routing header. Packets that originate from nodes which have a direct RF path to the network sink are short. Payload lengths grow linearly as the hop count increases. At the bit rate of the physical layer in use, each byte adds a total of 80μS to the length of the payload.

When the network sink receives a unique sensed data packet, it first examines the hop count of the packet in the routing header. Depending on this value, it knows how many nodes have interacted with the packet and included sensor readings. It is also aware of the position in the payload where to find sensor readings from the Nth hop.

III. Simulation

To estimate the TX duty cycle across different data send intervals with and without optimizations, the system was modeled in Matlab. The model calculates the overall TX duty cycle resulting in sending data packets at different rates,when one child node is attached and assumes sending to a duty cycled neighbor. The algorithm used is shown in Algorithm 1. Each transmission incurs a total TX time of T_{SYNC}ms worth of RTS/ CTS attempts, i.e., 4.8ms of TX time, plus the length of the payload. A payload length of 24 bytes is 2.4ms at the bit rate of the physical layer. The simulated experimental length was 2 days. The TX duty cycle is hence the total time spent

Algorithm 1 Model of TX on time during transmissions, neighbor learning enabled

```
Initialize Variables
repeat
   Time In TX = 0.0048s + Payload Length
   Accumulated_Time += Send Interval
until Accumulated_Time > 2Days
Duty Cycle = Time In TX / 2 Days
```

in TX mode divided by the length of the test. The results are shown in Figure 3. This graph includes 4 curves, two for neighbor learning enabled and two for neighbor learning disabled. The two curves where neighbor learning is enabled, include piggybacking enabled and disabled. The same applies for the other 2 curves, where neighbor learning is disabled.

TABLE II: Simulated duty cycle values when changing the data send intervals, NL=Neighbor Learn, PB=Piggybacking, None=No optimizations

Interval	NL+PB	NL	PB	None
1s	0.72%	1.44%	25.4%	50.8%
6s	0.12%	0.24%	4.233%	8.47%
20s	0.036%	0.072%	1.27%	2.54%
40s	0.018%	0.036%	0.635%	1.27%

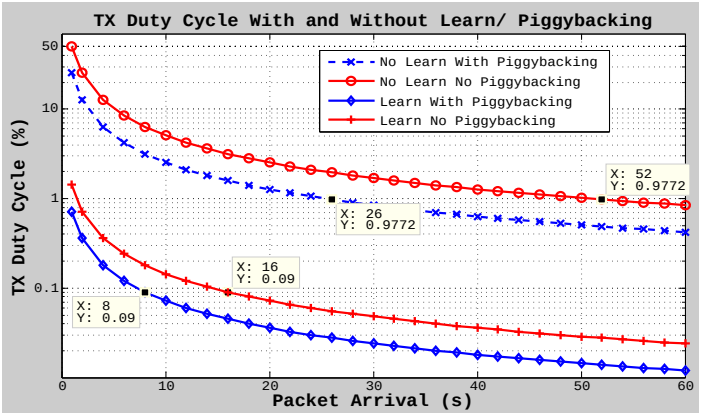


Fig. 3: Results of simulation showing the reduction in TX duty cycle due to Neighbor Learning and Piggybacking

In the case of neighbor learning and piggybacking being enabled, the simulation predicts a 70 times reduction in TX duty cycle compared to both being disabled. With both optimizations, nodes are capable of achieving 0.72% TX duty cycle forwarding data packets every 1 second. With both optimizations disabled this time increases to 52 seconds.The results are summarized in Table II. For each transmission, the node without neighbor learning spends on average half of the receive check interval performing RTS/ CTS, in this scenario that is 0.5s. Of that 0.5s half is spent in TX mode, or 0.25s (as previously discussed in Section II).

In the case of only the piggybacking optimization being enabled, there are two unique scenarios. Firstly, the node under test has piggybacking disabled and it forwards data packets for its child node and also generates its own additional data packets.The simulation shows it is only able to provide $< 1\%$ TX duty cycle when packets are being generated every 52 seconds. When piggybacking alone is enabled, the nodes can report back sensor readings every 26 seconds while remaining $< 1\%$ TX duty cycle.

IV. Empirical Evaluation

A. Testbed

The testbed used for this experiment consists of custom devices, comprised of a PIC24F microcontroller and an SX1211 868MHz radio transceiver. The radio is operated at 100kbps data rate and a sleep current of 1μA is achieved for the platform. The hardware platform is pictured in Figure 4.

 ISBN: 978-1-61208-296-7

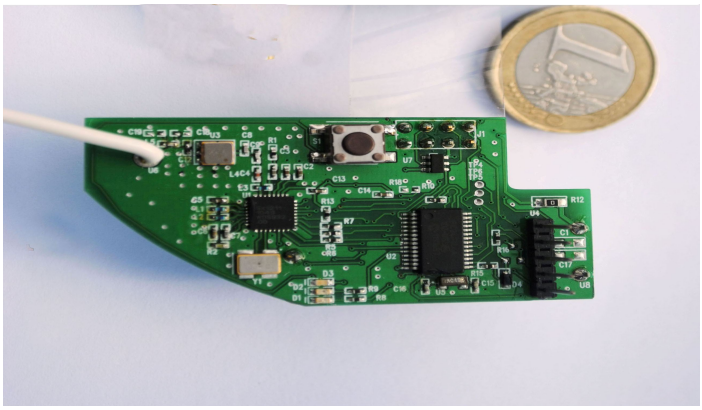


Fig. 4: Testbed WSN node, SX1211 868MHz radio and PIC24F microcontroller

B. Measuring TX on time

To be able to accurately estimate the overall TX duty cycle of deployed nodes, a counter is implemented in software which keeps track of the total amount of time spent in TX mode. This counter is included in each transmission and allows the network sink to keep track of how much time each node spends in TX mode.

The basic unit of time for this counter is 1ms, each increment of the 32-bit counter is equal to 1ms and it overflows after 60 days of constant TX on time. 1ms is the length of an RTS packet and is hence a convenient unit of time. Each transmission which takes place, results in the counter being incremented by the number of RTS/ CTS attempts required to contact the destination, plus the length of the payload transmission. This technique enables very accurate monitoring of the time spent in TX mode at each node. An example of this counter working is a transmission which requires 100 RTS/ CTS attempts to contact the destination and a payload length of 2ms. This transmission would results in the counter being incremented by 102, 100 for the RTS transmissions and 2 for the length of the payload.

The network sink which always listens, sends all received data packets to a PC over USB, where the results are logged. Using the technique described above, the TX duty cycle can be easily calculated for each node using Equation 1.

$$\text{Duty Cycle}(\%) = 100 \times \frac{\text{Packets} \times 0.0011}{\text{Length of Test}} \qquad (1)$$

A screen-shot of the printout from the network sink is shown in Table III. Here nodes 34 and 35 are forwarding packets through node 12 every 5 seconds.The TX duty cycle counter counts the total time spent in TX during transmissions, as can be seen in Table III, the difference between the two consecutive transmissions is 5 counter units or (5*1ms) 5ms. Also included is the RSSI of each hop, -54dBm from nodes 34 and 35 to node 12, and -44dBm from node 12 to the network sink. Most importantly from Table III the packet counter feature can be seen, node 35 requires 5 packets to send its status message (1395-1390).

TABLE III: Log at Network Sink

Hop	ID	Pkts	RSSI	ID	Pkts	RSSI	Time
R1	35	1390	-64	12	7183	-44	7:1:23:45
R1	34	1316	-54	12	7186	-44	7:1:23:48
R1	35	1395	-64	12	7189	-44	7:1:23:50
R1	34	1321	-54	12	7192	-44	7:1:23:53

C. Experimental Setup

To validate our modeled results presented in Section III a small network was deployed using the node pictured in Figure 4. The node's firmware also contains layer 3 routing protocols and these may cause fractional overheads in terms of TX duty cycle. An example of one of these overheads is a periodic probe message to check if the network sink is in range. The network sink was configured to be in an always listening state. This reduces the TX duty cycle of nodes which can communicate with the network sink because they only ever need 1 RTS/ CTS cycle before the payload data can be sent. All nodes were programmed to perform receive checks once a second and forward data packets at the same rate. The sensor sample interval and hence packet generation intervals were chosen to be 1, 2, 5, 10, 20, 30, 40 and 60 seconds.

The first experiment was devised to measure only the difference between nodes with neighbor learning enabled and disabled. For this experiment which was conducted, a total of 17 nodes were deployed in a large office and a maximum of 2 hops was observed. After having deployed the 17 nodes, it was observed that 11 of the nodes were able to communicate directly with the network sink. The remaining 6 were forced to route their messages through the 11 nodes which had a path to the network sink. The nodes of interest were the 6 nodes which did not have a direct RF path to the network sink because they were required to send to duty cycled neighbors. 3 of them were programmed with neighbor learning enabled and 3 without. Tests were conducted for 24 hours and all results were logged on a PC which was connected to the network sink. Of the 6 nodes under test, their TX duty cycles were calculated using Equation 1. 'Packets' represents the software packet counter shown in Table III and explained in Section IV-B.

The second experiment was devised to test the piggybacking and neighbor learning technique combined and a slightly different topology was required. The reason being that in the first experiment the leaf nodes were the only nodes sending to duty cycled neighbors. A controlled 3 hop topology was required because the nodes under test needed to be parent nodes, and still forward to duty cycled neighbors. The topology is depicted in Figure 5 and the nodes under test are the yellow nodes (2^{nd} row from right).

D. Results

The first experiment was devised to solely measure the efficiency of our implemented neighbor schedule learning optimization. The TX duty cycle of a total of 6 leaf nodes was measured over a 1 day period (3 with neighbor learning, 3 without). The results are summarized in Table IV. Each test (i.e., each send interval) was carried out a total of 3 times and the maximum,minimum and overall average TX

 ISBN: 978-1-61208-296-7

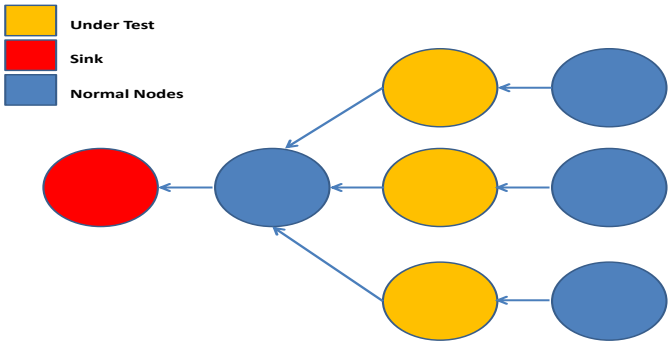


Fig. 5: Topology for 2^{nd} Experiment, Yellow nodes are the nodes under test

duty cycles are listed in Table IV. In Figure 6 , simulated and tested results are compared. The empirically tested values are in extremely close agreement with the modeled results. The maximum variation in the predicted vs the measured duty cycle is 0.044% when sending packets every 40 seconds. Minute variations can be observed between the predicted and tested results, these variations can be explained by layer 3 routing protocol overheads. These variations can be explained by the overheads mentioned in Section IV-C.

The second experiment was designed to measure the efficiency of our piggybacking and neighbor learning algorithm combined. The TX duty cycle of 3 parent nodes each with 1 child node a piece, was measured. The experiment was carried out using a number of different configurations. Different combinations of piggybacking/ neighbor learning enabled and disabled were used. The results are presented in Figure 6. In the case of piggybacking only being disabled, the parent node under test forwards packets for its child and generates its own packet during each data send interval. In the experiment where piggybacking was enabled, the parent node sends only one packet during each data send interval. The difference in TX duty cycle is intuitively approximately 50%, as the workload of the parent is reduced by a factor of 2 when piggybacking is enabled. The slight increase in the payload length due to piggybacking is insignificant when compared to the amount of time spent in TX mode during the RTS/ CTS phase of the transmission (each byte adds only 80μs compared to 1ms for an RTS/ CTS cycle).

In the case of neighbor learning and piggybacking being enabled we observe a 70 times reduction in TX duty cycle, 0.7% forwarding every second (6b), when both are disabled this forwarding rate must be increased to almost 50 seconds to achieve $< 1\%$ TX duty cycle (6a). With both optimizations enabled, parents with a single attached child node can forward packets every 8 seconds to comply with the $< 0.1\%$ TX duty cycle restrictions (6b), with 2 child nodes attached this figure increases to 16 seconds.

From Figure 6a, there is very close correlation between experimental and predicted results. With only piggybacking enabled the data send interval can be 26 seconds while still remaining below 1%, with it disabled, the data send interval must be 52 seconds to comply with the 1% TX duty cycle restrictions.

TABLE IV: Simulated Duty Cycle Values vs Tested, Experiment 1. Minimum, Maximum and Average values included

Interval	Simulated	Test (Avg)	Test (Min)	Test (Max)
1s	0.72%	0.74%	0.71%	0.76%
2s	0.36%	0.37%	0.33%	0.39%
10s	0.072%	0.079%	0.076%	0.078%
20s	0.036%	0.039%	0.036%	0.042%
30s	0.024%	0.026%	0.025%	0.028%

V. Related Work

In terms of the two optimizations and their novelty, there are some similar concepts to be found in the literature. Protocols such as WiseMAC [4] and Hui and Culler's techniques in [5], use similar neighbor schedule learning systems. The differences between WiseMAC and this work are the following: WiseMAC uses an uninterrupted preamble wakeup sequence that doest not contain address information. This work uses an RTS/CTS system which will stop as soon as the destination responds (zero over-listening). WiseMAC requires periodic exchange of scheduling information, this work does not. WiseMAC relies on a Layer 1 receive check, this work uses Layer 2 (i.e., lower power in dense networks due to less overhearing but a slight increase in receive check energy). WiseMAC is also for infrastructure networks and it only considers down-link traffic (parent to children).

In [5], the authors briefly mention and describe their neighbor schedule learning system. They improve upon WiseMAC's preamble only wakeup stream by adding some address information into the wakeup stream. Receive checks are layer 1 based and nodes which overhear can quickly decide the packet is not destined for them because of the embedded address information. But still overhearing does occur, unlike this work. Schedule information is exchanged by including extra data in acknowledge messages, in this work no extra data is transferred to provide neighbor schedule learning.

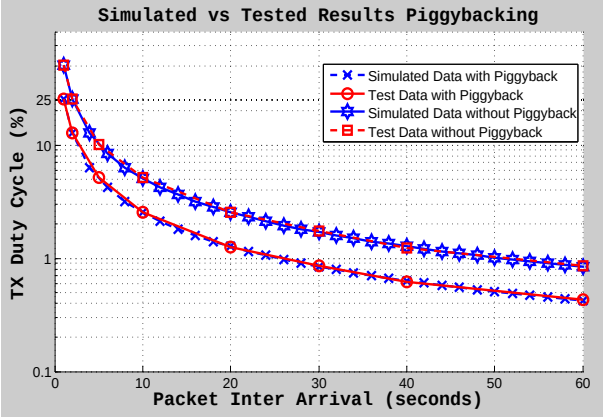
Ye et al. in SCP-MAC [10], present a MAC protocol where all nodes are scheduled to wake during the same time window. Transmissions now take place within this window, resulting in a reduced TX duty cycle compared to standard duty cycled MAC techniques. SCP-MAC requires additional scheduling packets to be transmitted and this has an impact on the overall TX duty cycle, it also suffers from high receive check energy and high latency.

The authors in [4], [5] and [10] present results on the reduction in power consumption of their techniques but fail to investigate the potential reduction in TX duty cycle. This work leverages some of these concepts and simplifies/ optimizes them and applies them to industrial applications.

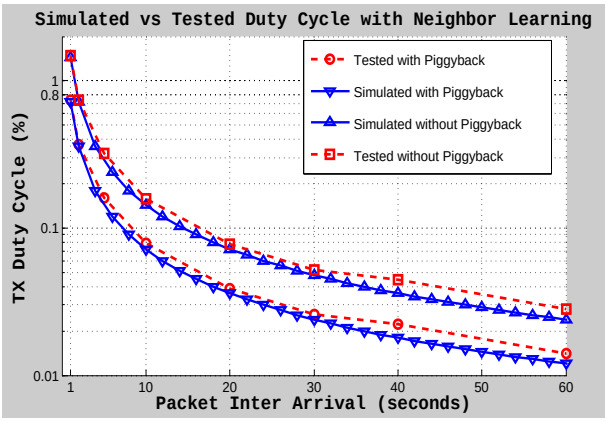
VI. Conclusion & Future Work

In this paper, two techniques to reduce TX duty cycle are described, simulated and empirically evaluated. WSNs which must report real time sensor readings are considered and data aggregation techniques such as in [9] are not considered. Our techniques improve functionality of WSN deployments by allowing users of license free bands to increase network activity, while still remaining within the legal maximum TX

 ISBN: 978-1-61208-296-7



(a) Results of simulation showing the reduction in TX duty cycle due to Piggybacking alone. Upper two curves have piggybacking disabled and lower two have it enabled. Excellent correlation between predicted and tested results. In general piggybacking reduces TX duty cycle by a factor of 2



(b) Neighbor Learning Simulated vs Test Data, excellent correlation between simulated and tested results, maximum error of 0.044% when packets are being sent every 40 seconds. Upper two curves have piggybacking disabled and lower two have it enabled

Fig. 6: Empirical Test Results vs Simulated Results

duty cycle requirements. Using both of the aforementioned techniques, it is demonstrated that sparse (non-dense) networks can provide sensor readings 70 times more frequently to comply with the $< 0.1\%$ TX duty cycle requirements of some of the ISM bands.

Specifically, it is shown that using both techniques and having only 1 dependent child, sensor readings can be forwarded every 8 seconds while still complying, and every 16 seconds when 2 child nodes are attached. The result is a far more active network which is able to provide more frequent sensor readings to the end user. Another important by-product of the drastic reduction in TX duty cycle is the reduction in power consumption and increased lifetime of the battery powered network.

There are a few distinct areas where our work can be improved upon and developed further. For applications where latency is not an issue, large savings in TX duty cycle could be made if nodes which are under a large workload (i.e., multiple child nodes) could queue received data packets and transmit them all in one burst. This would prevent the TX duty cycle from increasing with workload, the disadvantages of this approach would be that sensor readings would no longer be real-time and payload lengths would increase drastically. The performance of the developed techniques in large scale multi-hop deployments is also of interest. The authors would also like to compare their work against other cross layer approaches such as Dozer and Koala [3], [7].

ACKNOWLEDGMENT

The authors would like to acknowledge and thank the support of EI Electronics and IRCSET (Irish Research Council for Science Enterprise and Trade).

REFERENCES

[1] ETSI EN300 220-1, 2012.

[2] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006.

[3] Nicolas Burri, Pascal von Rickenbach, and Roger Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proceedings of the 6th international conference on Information processing in sensor networks*. ACM, 2007.

[4] Amre El-Hoiydi and Jean-Dominique Decotignie. Wisemac: An ultra low power mac protocol for multi-hop wireless sensor networks. In SotirisE. Nikoletseas and JosD.P. Rolim, editors, *Algorithmic Aspects of Wireless Sensor Networks*, volume 3121 of *Lecture Notes in Computer Science*, pages 18–31. Springer Berlin Heidelberg, 2004.

[5] Jonathan W. Hui and David E. Culler. Ip is dead, long live ip for wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 15–28, Raleigh, NC, USA, 2008. ACM.

[6] D. Moss and P. Levis. Exploiting physical and link layer boundaries in low-power networking. Technical report, Stanford, 2008.

[7] Razvan Musaloiu-E., Chieh-Jan Mike Liang, and Andreas Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*. IEEE Computer Society, 2008.

[8] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 95–107, New York, NY, USA, 2004. ACM.

[9] S. Sivaranjani, S. Radhakrishnan, and C. Thangaraj. Adaptive delay and energy aware data aggregation technique in wireless sensor networks. In VinuV Das and Yogesh Chaba, editors, *Communications in Computer and Information Science*, volume 296, pages 41–49–. Springer Berlin Heidelberg, 2013.

[10] Wei Ye, Fabio Silva, and John Heidemann. Ultra-low duty cycle mac with scheduled channel polling. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 321–334, Boulder, Colorado, USA, 2006. ACM.

 ISBN: 978-1-61208-296-7