An Adaptive Sleep-Time Management Model for Wireless Sensor Networks

Eyuel D. Ayele, Jianjun Wen, Zeeshan Ansar and Waltenegus Dargie

Chair for Computer Networks, Faculty of Computer Science
Technical University of Dresden
01062 Dresden, Germany
Email:{eyuel.ayele, jianjun.wen, zeeshan.ansar, waltenegus.dargie}@tu-dresden.de

***Abstract*—The energy consumption of a wireless sensor network affects its lifetime which in turn affects the scope and usefulness of the network. Most existing or proposed MAC protocols enable nodes to specify a duty cycle, so that they can sleep much of the time to save energy. However, only very few models exist to determine the appropriate time and duration of a sleep phase. Existing approaches rely on pre-calculated sleep durations or are difficult to implement on real platforms. We propose a runtime and adaptive model to estimate the sleep time and duration of wireless sensor nodes. Our model takes the statistics of incoming and outgoing packets at a relay node which is then supplied to a general queueing model. The model is lightweight and can be fitted into any existing MAC protocol. We have implemented our model for TelosB platform and TinyOS environment. We integrated our model with two existing protocols (TinyOS LPL MAC and XMAC) and compared the performance of these protocols with and without our model. The performance evaluation results show that the energy consumption of a relay node reduced by 11.4 – 64.8%. The overall throughput of the network increased by up to 24%. Moreover, our model readily responded to changes in packet traffic rate while at the same time increasing the packet transmission reliability by 64.5 – 67.4% for different traffic scenarios.**

***Index Terms*—Adaptive sleep time, duty cycle, energy efficiency, MAC Protocol, queueing Model, runtime management, wireless sensor networks**

I. Introduction

Most wireless sensor networks should operate for a long time, unattended and without frequent replacement or recharging of batteries. This can be achieved only if they consume a small amount of energy at a time. Fortunately, the interesting phenomena the networks should monitor occur infrequently and nodes can turn off their radio and set their processor on a low-power mode whenever they are idle. For this reason, almost all existing or proposed medium access control protocols enable applications to specify the duty cycle of nodes, the portion of time a node stays active. It is defined as:

$$D = \frac{t_a}{t_a + t_s} \times 100\% \tag{1}$$

where t_a and t_s are the active and sleep durations, respectively, and $t_a + t_s = T$ is the period of the duty cycle. Alternatively, it can be sufficient to define only t_s, so that once the nodes are awake, they can stay as long as their activity lasts, but they should sleep for a specified amount of time before they are ready for their next round of activity.

Whereas the need to define a duty cycle is recognized early on, so far very few models exist to determine this essential parameter. For example, in the first well-documented field deployment for habitat monitoring, a duty cycle of 1.7% is specified (i.e., nodes sleep 98.3% of the time) but no justification is given for this particular value [1]. The first MAC protocol that introduced the use of duty cycle for wireless sensor networks does not address how it should be determined [2]. Likwise, TMAC [3] introduces the concept of adaptive duty cycle but ignores its realisation. In some cases, experimental observations and knowledge of applications requirements are used to determine the values of a duty cycle [4], [5].

More recently, different approaches have been proposed to determine the duty cycle of wireless sensor networks. Wang et al. [6] pre-compute different sleep times for different application data rates and store these values in a table. The MAC protocol keeps record of these sleep times and chooses the value that is suitable for the data rate of the application (to satisfy the application's requirement). Similarly, Zimmerling et al. [7] compute the optimal duty cycle of a wireless sensor network by taking different application-layer constraints (energy, reliability and per-hop latency) into account.

In this paper, we propose an adaptive model for computing the optimal sleep duration of a node in a wireless sensor network. Our approach is modular, meaning it is totally independent of the MAC and application layers and separates the concern of managing duty cycle from computing the duty cycle. Consequently, it can be integrated into any kind of MAC protocol. It is also dynamic, in that it autonomously adapts the sleep duration of individual nodes to changes in their surrounding, because the model relies on the statistics pertaining to the packet reception and transmission of individual nodes.

The contributions of the paper can be summarized as follows:

1) A Modular framework: Our model effectively separates the functionality of controlling the runtime sleep duration from the MAC and the application layers. The algorithm we propose keeps track of the changes in packet traffic statistics to determine the optimal sleep

duration.

2) A distributed approach: Since the packet reception and transmission statistics of individual nodes can be different, a global duty cycle management strategy may not be efficient. Moreover, centralized approaches are often costly and slow to react to changes. Therefore, our approach relies only on local information and enables individual nodes to determine their own sleep schedule.
3) A lightweight analytic model: We propose a G/G/1/K queueing model to determine runtime parameters. The desirable characteristic of this model is that it is able to capture the statistical changes of the network traffic without the need to specify any MAC or application layer constraints. Furthermore, it does not make any assumption about packet arrival and transmission statistics.
4) Prototyping: We have implemented and tested our model on real sensor platforms (TelosB and TinyOS) and compared the performance of two types of protocols (XMAC [4] and TinyOS LPL MAC [8]) with and without our model. In all the experiments we conducted, our approach has impressive results.

The rest of this paper is organized as follows: In Section II, we present related work and highlight the merits and drawbacks of existing or proposed approaches. In Section III, we introduce and discuss the theoretical aspects of our approach in detail and in Section IV, we discuss its implementation. The algorithm that computes and manages sleep time is also presented in this section. In Section V, we present the validation and evaluation of our protocol and, finally, in Section VI we give concluding remarks and outline future work.

II. Related Work

Almost all existing or proposed MAC protocols for wireless sensor networks enable nodes to sleep much of the time. The prevailing focus in the design of the first generation of protocols has been on enabling the nodes to effectively communicate with each other in spite of the fact that they may have different sleeping and waking times. Hence, sleep schedule synchronization and asynchronous communication mechanisms have been proposed [4], [5], [9], [10], [11], [12]. However, the research community gradually realized that determining the appropriate sleep and active durations was of equal importance, since these two parameters influence the quality of data that can be extracted from the networks and the lifetime of the networks. The longer nodes sleep, the longer becomes their lifetime, but the associated packet transmission latency also becomes longer, since nodes have to wait longer until their neighbours are ready to communicate with them. On the other hand, short sleep durations enable faster packet delivery, but reduce network lifetime.

Zimmerling et al. [7] propose a framework for adapting the MAC layer parameters at runtime, so that appropriate trade-offs between the lifetime of the network and the requirements of applications can be made. The framework employs the Glossy flooding algorithm [13] to collect information pertaining to the state of the network in real time and the optimal MAC parameters are determined outside of the network using this information. The optimality criteria is fulfilling specified application requirements (network lifetime, end-to-end reliability and latency). Then, the framework disseminates the parameters to all nodes in the network. The limitation of the framework is its introduction of a large communication overhead when collecting state information and disseminating MAC parameters. To reduce such overhead, Challen et al. [14] propose a decentralized framework by arguing that local changes in the state of the network often affect local nodes only and, therefore, it is sufficient to optimise MAC parameters locally. Unlike the framework of Zimmerling et al. the framework of Challen et al. focuses on minimizing the energy consumption of the network and does not consider other constraints. Each node maintains a vector of combined energy load and calculates the optimal trade-off between energy and network utility.

Wang et al. [6] propose an off-line approach which pre-calculates sleep times for different data rates required by applications. Each data rate corresponds to a specific activity duration and a specific energy consumption profile for receiving and transmitting packets. Then (*sleep time*, *average energy*) pairs are stored in a table based on which each node locally decides its sleep time at runtime. The merit of this approach is the small runtime overhead required for determining the appropriate sleep time. The limitation is the poor usability of pre-calculated sleep times in a network which experiences a high degree of network dynamics.

Byun et al. [15] propose a queueing model in a feedback control system to dynamically adjust the sleep time of a node. When the the queue size is constrained to a predetermined value, it produces a sleep time that achieves high energy efficiency and low delay. The model is analytic and simulation results confirm to its usefulness, however, it is too complex to be useful in practical settings. Likewise, Vigorito et al. [16] propose to control theoretic model to determine the optimal duty-cycle in a wireless sensor network that harvests energy from its environment. Thus, nodes harvest energy while sleeping and consume energy while they are active. The relationship between the harvested and consumed energy is expressed as a linear quadratic tracking problem. The objective of the model is to minimise the error between anticipated harvest and actual consumption, so that the system is always at an energy neutral operation. The duty cycle that achieves this state is the node's duty cycle for the next phase

Li et al. [17] propose a distributed algorithm to control the sleep duration of nodes by employing convex-optimization. The aim is to achieve energy-fairness while allowing nodes to determine their own sleep schedule. Nodes adjust their sleep time locally by exchanging current sleep interval, energy consumption rate with their neighbours. The algorithm is self-adaptive for different traffic loads and adjusts the upper bound on the energy consumption rate. However, the paper does not reveal the overhead of exchanging information.

Generally, the proposed approaches for adaptively determining the duty cycle of nodes rely either on centralized

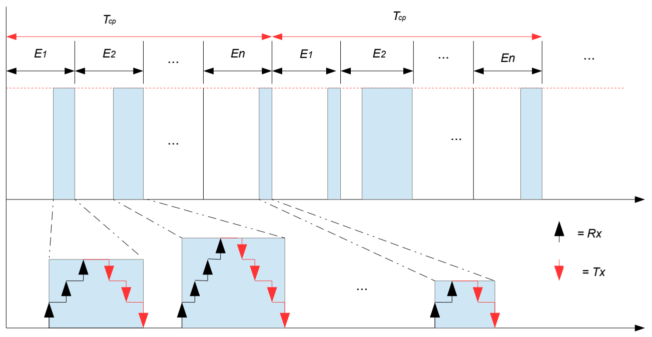


Figure 1: Packet reception and transmission pattern at a relay node. Black arrows show the number of packets received in succession whereas red arrows represent transmitted packets.

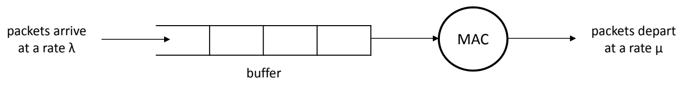


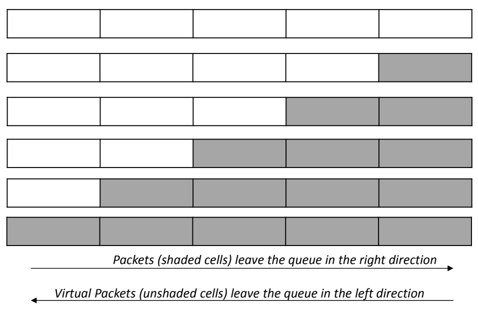
Figure 2: A general queue process to model the packet arrival and departure pattern at a relay node.

coordination or pre-calculated empirical values to adapt the sleep time. Therefore, their applicability is limited to small-scale networks where the topology of the networks does not change significantly over time. In contrast, our approach is scalable, distributed, and lightweight; it is capable of coping with significant changes in the network topology. By considering the local packet traffic statistics, individual nodes are able to determine their optimal sleep duration and to announce it to their immediate neighbours only, which directly are affected by their sleep schedule. Our model autonomously adapts the sleep time to network traffic changes at runtime, without imposing application or MAC layer constraints, unlike the approaches proposed in, for example, [6] and [7].

III. Theoretical Concept

Packets bound to be transmitted arrive at the MAC layer either from the network layer or directly from the application or the sensing layer. Generally, the time between the arrival of two packets (inter-arrival time) is a random variable, since it cannot be known in advance except in a probabilistic sense. Similarly, the duration between the time a MAC protocol begins transmitting a packet and the arrival of an acknowledgement of successful transmission is a random variable because no packet can be transmitted instantly. This time depends on several factors, such as the back-off time chosen by the MAC protocol to avoid collision and the time it takes the MAC protocol to win the channel through contention (which, in turn, depends on the data generation rate of the neighbour nodes and the network density).

Fig. 1 illustrates the activity pattern of a relay node. As can be seen, the number of packets that arrive at the MAC layer in succession and the time required to transmit them are variable. Likewise, the duration of the idle time between two activity phases is variable. If the expected idle time can be estimated,


Packets (shaded cells) leave the queue in the right direction
Virtual Packets (unshaded cells) leave the queue in the left direction
Figure 3: Creation of a queue of length 5. As packets arrive at the queue, a queue is created. At the same time, the unoccupied places in the queue form a virtual queue in the opposite direction.

then it is possible to let the node sleep for that duration to save energy.

By considering the MAC buffer as a queue of fixed capacity (k packets) and the MAC protocol as a server for transmitting the packets in the buffer (queue), then it is possible to adaptively estimate the expected idle time of a relay node using a general queueing model, $G/G/1/k$ (see Fig. 2). In this model, each packet arrives and is processed with different arrival and service rates λ_i and μ_i, where the index i is associated with the $i-th$ packet. The mean packet arrival and service rates are depicted by λ and μ, respectively. According to Little's Law [18], the expected number of packets in the queue is expressed as:

$$L = \lambda W \tag{2}$$

where W is the expected waiting time, which is a function of the mean service rate μ. If the number of packets in the buffer at an arbitrary time is denoted by N and the number of packets in the buffer by the time a particular packet A arrives at the queue by N_A, the queue length probability distributions can be expressed as:

$$P_n = P\{N = n\}, 0 \leq n \leq K \tag{3}$$

$$P_n^A = P\{N_A = n\}, 0 \leq n \leq K \tag{4}$$

Note that P_0 is the probability that the queue is empty, signifying the probability of a node experiencing an idle time.

Kim and Chae [19] propose two complementary expressions to determine P_0. In the first expression, they consider a normal queueing formation whereas in the second, they consider the mirror image of the normal. As can be seen in Fig. 3, the white cells represent empty cells in the buffer and the shaded ones indicate occupied cells in the buffer. So, it is possible to imagine of "two" queues, which are complementary to each other. The one is formed by the arrival of packets while the other is formed by their departure (we call it a virtual queue). In the beginning, the former is empty while the latter is full.

The reason for considering "two" queues is that they can yield two different expressions for P_0 which can then be combined so that P_0 can be expressed in terms of $\lambda_i, \mu_i, \lambda$, and μ.

Hence, according to Kim and Chae, using the queue formed by real packets, the probability that there are n number of packets in the queue is expressed as:

$$P_n = \lambda(P^A_{n-1}\mu^{-1}_{n-1} - P^A_n\mu^{-1}_n + P^A_n\mu^{-1}), 1 \leq n \leq K-1 \quad (5)$$

And the probability that the buffer is empty is expressed as:

$$P_0 = 1 - \rho\left(1 - P^A_K\right) \quad (6)$$

where $\rho = \frac{\lambda}{\mu}$ is the traffic density. Similarly, using the virtual queue,

$$P_n = \lambda\left(P^A_n\lambda^{-1}_n - P^A_{n-1}\lambda^{-1}_{n-1} + P^A_{n-1}\lambda^{-1}\right), \quad 1 \leq n \leq K-1 \quad (7)$$

And,

$$P_0 = \lambda P^A_0\lambda^{-1}_0 \quad (8)$$

We formed simultaneous equations using Equations 5, 6, 7, and 8 to get the following expression:

$$P^A_n = P^A_{n-1}\left(\frac{\mu^{-1}_{n-1} + \lambda^{-1}_{n-1} - \lambda^{-1}}{\lambda^{-1}_n + \mu^{-1}_n - \mu^{-1}}\right) \quad (9)$$

Equation 9 can be recursively calculated to derive an expression for P^A_0 in Equation 8. As a result, the probability that the buffer is idle can be expressed as:

$$P_0 = \lambda\lambda^{-1}_0\left(\frac{1-\rho}{\lambda\lambda^{-1}_0 - \rho\Psi}\right) \quad (10)$$

where:

$$\Psi = \frac{\beta_0\beta_1\cdots\beta_{K-1}}{\gamma_1\gamma_0\cdots\gamma_K} \quad (11)$$

$$\beta_n = \mu^{-1}_n + \lambda^{-1}_n - \lambda^{-1} \quad (12)$$

$$\gamma_n = \lambda^{-1}_n + \mu^{-1}_n - \mu^{-1} \quad (13)$$

IV. Framework

We implemented the general queueing model to manage the sleep time of nodes in a wireless sensor network. Our framework is modular in that it can be plugged into any MAC protocol. We plugged it into two existing protocols without significant modifications to the existing codes.

The framework has three components. The first component is responsible for monitoring and recording the arrival and departure time of incoming and outgoing packets from which it estimates the packet arrival and service rates. In the second component resides our idle time estimation algorithm, which is the implementation of Equation 10. The third component is a runtime sleep management component which is responsible for managing and disseminating the sleep schedule of the node. Figure 4 displays the architecture of our framework. We have depicted the framework outside of the MAC protocol to demonstrate its modularity and that its implementation does not significantly affect the realisation of the MAC protocol.

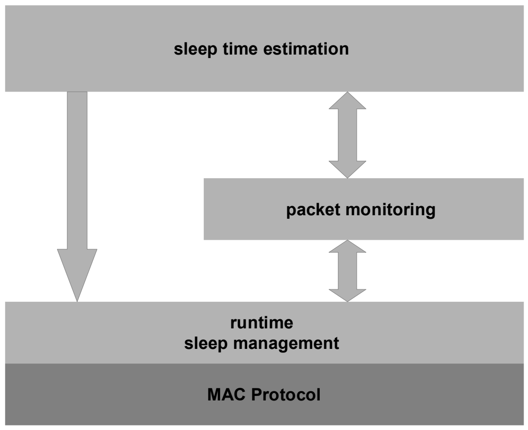


Figure 4: Model Framework

A. Algorithm

The algorithm which computes the sleep time of a node divides time into epochs (E) and control periods (T_{cp}) (see Figure 1). A single epoch consists of an activity period, t_a, which is then followed by an extended idle period, t_s, during which the node should sleep. A relay node receives and transmits **m** number of packets within a single epoch (where **m** is a random variable). There are n epochs in a single control period and n depends on the packet generation rate of the network. For a given configuration, n is a fixed number. For all our experiments we set $n = 5$. All epochs of a single control period will have the same sleep time. A control period also serves as an observation period for collecting statistics for the next sleep duration of a node.

Within a single control period, the packet monitoring component calculates λ_i and μ_i for all incoming and outgoing packets and store them until the observation time is completed. After n epochs are completed, the monitoring component forwards the parameters n, λ_i and μ_i to the algorithm, which then computes P_0 using Equation 10 and t_s using Equation 14:

$$t_s = P_0 \times \frac{T_{cp}}{n} \quad (14)$$

where n is the number of epochs. The length of T_{cp} in seconds depends on the width of each epoch. Algorithm 1 summarises the basic steps required for dynamically computing the sleep time of a node.

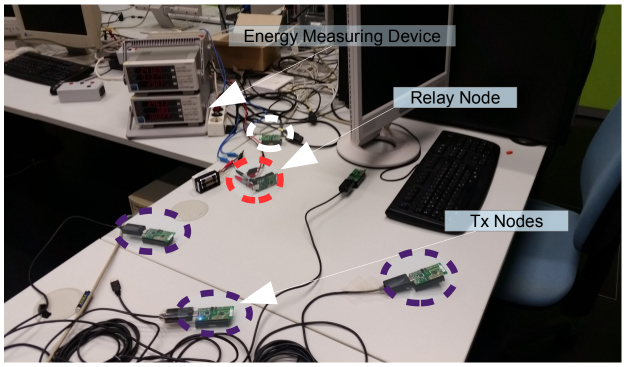
Algorithm 1 Adaptive Sleep Time Managment Algorithm

```
Input: $\lambda_i$, $\mu_i$, $n$
Output: $\Psi$, $\rho$, $P_0$,$T_{cp}$, $t_s$
 1: procedure SLEEP TIME COMPUTATION
 2: top:
 3:     if $i < n$ then
 4:         $\lambda_i \leftarrow \lambda_i$
 5:         $\mu_i \leftarrow \mu_i$
 6:     if $i = n$ then
 7:         $\Psi \leftarrow \Psi$
 8:         $\rho \leftarrow \rho$
 9:         $T_{cp} \leftarrow \sum_{i=1}^{n} E_i$
10:         $P_0 \leftarrow P_0 = \lambda\lambda_0^{-1}\left(\frac{1-\rho}{\lambda\lambda_0^{-1} - \rho\Psi}\right)$
11:         $t_s \leftarrow P_0 \times \left(\frac{T_{cp}}{n}\right)$
12:         $RESET$
13:         $i \leftarrow 0$
14:         close;
15:     goto top
```

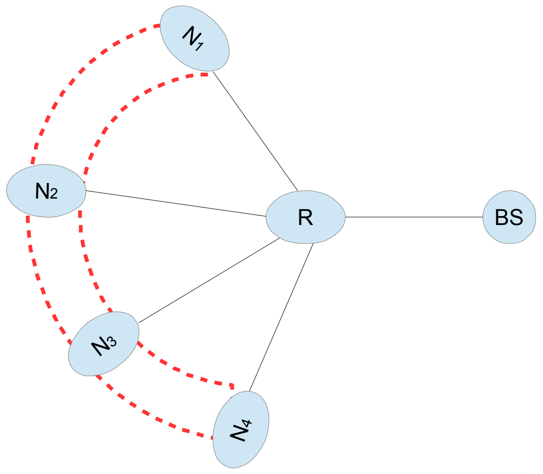
B. Implementation

We integrated our framework into the XMAC [4] and the TinyOS LPL MAC [8][20][21] protocols for TinyOS and deployed the protocols on TelosB nodes. These nodes formed a network consisting of four source nodes, a relay node and a base station. Figure 5 displays the experimental set up of our wireless sensor network.

Both the LPL MAC and XMAC define preambles to enable asynchronous sleep schedules in wireless sensor networks. Even though nodes are free to independently decide when to sleep, the duration of the sleep period is fixed and the length of the preamble is determined by the fixed sleep time. A preamble enables a transmitter node to express its intention to communicate with a relay node. The preamble transmission continues until the transmitter receives an acknowledgement signifying the readiness of the relay node to communicate.



(a) Device and measurement setups.



(b) Network topology.

Figure 5: A wireless sensor network experimental set up for testing the adaptive sleep time of two protocols deployed in TelosB sensor nodes.

V. Evaluation

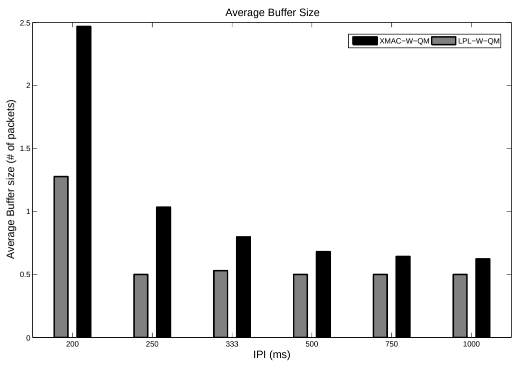
We evaluated the performance of our framework by specifying different performance metrics. These are:

1) Reliability: It is a measure of the ratio of the number of packets successfully received by the base station to the number of packets generated and forwarded to the relay node by the source nodes.
2) Power consumption: We consider the average power consumption of the entire network as well as the instantaneous power consumption of the relay node.
3) Throughput: The number of packets received by the base station for one hour.

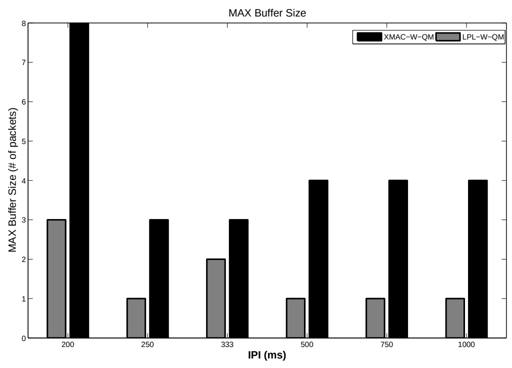
We measured these metrics by operating the network with and without our framework and by varying the packet arrival rates of the source nodes. In all the subsequent figures we append the label -W-QM to the name of a protocol when the queueing model was integrated with it and -WO-QM when it was not.

The source nodes generated packets with different inter-packet intervals (IPI) that vary from 100 ms to 1000 ms. These correspond to packet generation rates varying from 10 packets per second to 1 packet per second. Each node forwarded the packets to the base stations via the relay node after wining the medium through contention, as per the implementation specifics of the XMAC and the LPL protocols. The relay node, too, competes with the source nodes to forward the packets to the base station. We fixed the sleep time of the two protocols to 100 ms when they were used without our framework; the reason for choosing this value was to accommodate the highest traffic density.

We used a digital oscilloscope and digital power analysers (Yokogawa WT210) to measure and analyse the power consumptions of the nodes.



(a) Average Buffer Size



(b) Maximum Buffer Size

Figure 6: The buffer size of a queue for different inter-packet intervals.

A. Queue Size Distribution

Fig. 6 shows the average and the maximum queue buffer length distributions for the different input traffic rates when the two protocols integrated our framework. To measure the buffer size we considered the following IPI: 100, 250, 333, 500, 750 and 1000 ms. The maximum buffer size was eight packets when the XMAC-W-QM was employed (for an IPI of 200 ms, signifying high traffic) and three packets when LPL-W-QM was employed. This observation enabled us to select the appropriate buffer length that prevented the queue from overflowing. Hence, throughout our experiment, we set K = 10.

B. Reliability

As stated previously, the reliability is the measure of the number of successfully delivered packets to the destination node (base station). The sleep time has a direct bearing on the reliability of packet transmission, since packets can be lost if there is a mismatch between the transmission time of source nodes and the sleep time of relay nodes. The drawback of our framework is that the sleep time of the relay node has to be adaptively computed and new values have to be communicated to the source nodes. The drawback of the existing protocols when they are used without our framework is that as the IPI changes, the size of their preamble remains unchanged as a result the schedule mismatch between the source nodes and the relay nodes increases, increasing the packet loss rate.

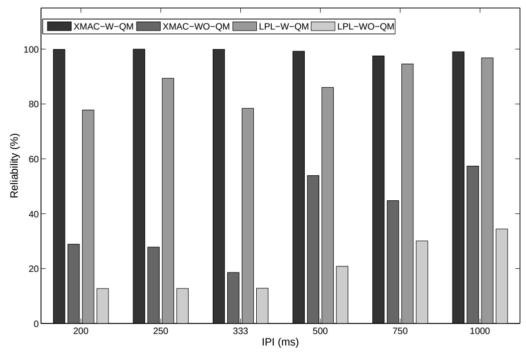


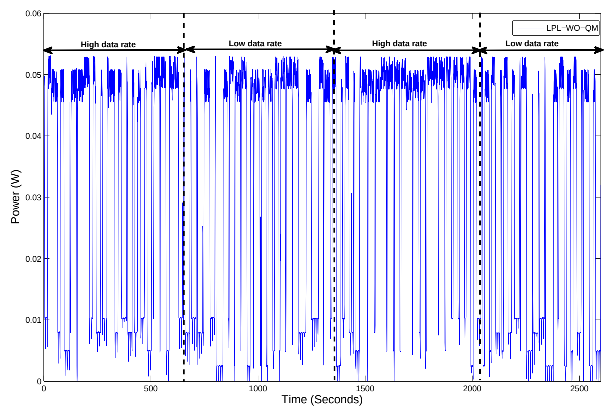
Figure 7: Packet delivery reliability of the two protocols with and without the queueing framework integrated.

As can be seen in Fig. 7, the reliability of the MAC protocols which integrated our algorithm is consistently high. This is particularly true for the XMAC protocol. The reliability of both protocols without the queuing model is low in general, but it is significantly lower for smaller IPI. Overall, the packet transmission reliability increased by 64.5% (for small inter-packet intervals) and by 67.4% (for large inter-packet intervals).

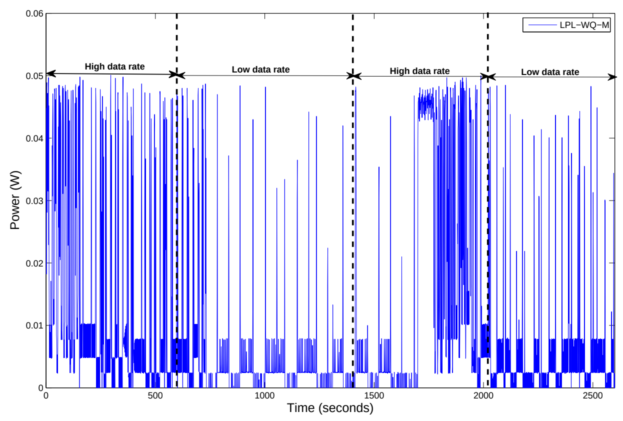
C. Power Consumption

Fig. 8 displays the average power consumption of the entire network during a 1-hour operation. All the protocols consume a comparable amount of power at smaller IPI, which is plausible since the nodes are active much of the time (our framework consumed a relatively large amount of power for IPI = 200 ms, apparently due to the background process that computed the sleep time; when the packet arrival and departure rates increase, the duration of each epoch becomes smaller, as a result, the control period becomes shorter and the framework frequently computes new sleep time). However, as the IPI increases, the idle time plays a significant role in reducing the power consumption of a node. Interestingly, our framework significantly reduced the average power consumption of the default TinyOS LPL MAC protocol almost always.

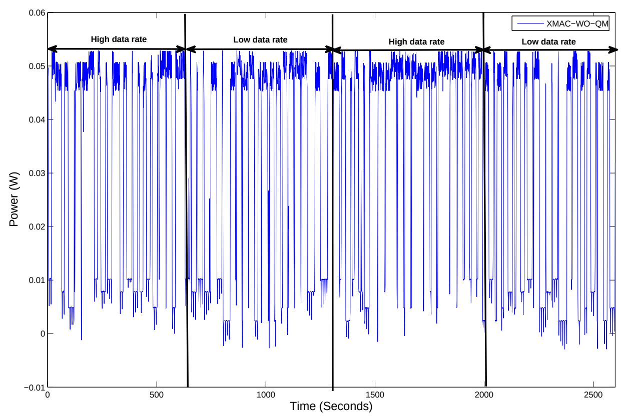
Fig. 9 displays the instantaneous power consumption of the relay node for the four different cases (with and without our framework integrated into the XMAC and LPL protocol). In all the cases, the power consumption of the relay node experiences fast fluctuations, because both protocols support low duty cycle. However, as we vary the IPI, our model changes its sleep time accordingly, allowing the relay node to sleep longer when the IPI was larger. Overall, the average power consumption of the relay node reduced by 11.4% (for small inter-packet intervals) and by 64.8% (for large inter-packet intervals).



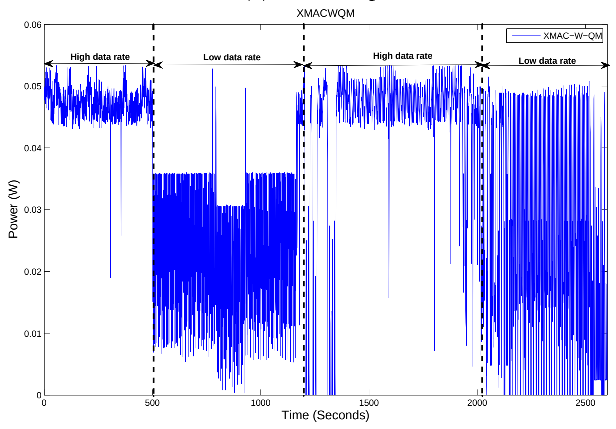
(a) LPL-WO-QM



(b) LPL-W-QM



(c) XMAC-WO-QM



(d) XMAC-W-QM

Figure 9: A snapshot of the instantaneous power consumption of a relay node for different packet generation rates.

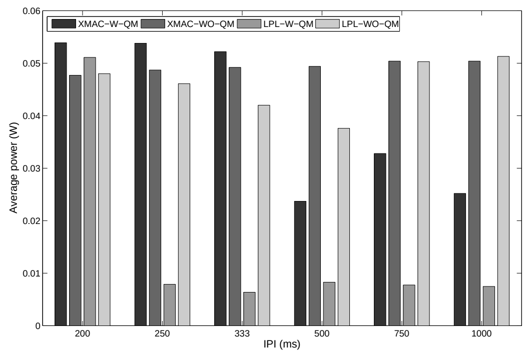


Figure 8: The average power consumption of the entire network with respect to different inter-packet intervals.

D. *Throughput*

We defined throughput as the amount of packets extracted from the network during its 1-hour operation. We expected a modest reduction in throughput when our framework was employed due to the trade-off between the reduction in power consumption (as a result of longer sleep times), on the one hand, and the throughput, on the other. While this is true for the XMAC protocol (compared to XMAC-WO-QM, the XMAX-W-QM yielded lower throughput when the inter-packet interval was 166 ms, 200 ms, 333 ms, and 1000 ms), for the LPL protocol, however, its performance was encouraging, since almost always, our framework produced either comparable or even better throughput even though the power consumption of the LPL protocol was considerably larger when our framework was not integrated into it. Fig. 10 shows the throughput of the network for the different scenarios. The overall throughput of the network increased by up to 24%.

VI. Conclusion

In this paper we proposed, implemented, and tested a dynamic and adaptive sleep management model for wireless sensor networks. Our model is modular and lightweight and can easily be integrated into existing MAC protocols. We integrated it into the default TinyOS LPL MAC protocol and XMAC protocol and quantitatively evaluated various performance and energy metrics. The model implemented a $G/G/1/k$ queueing model, where the two Gs stand for general inter-packet interval and service rates, respectively, and k refers to a fixed buffer size.

In our model, time is divided into epochs and control periods. The sleep duration of all epochs belonging to a single

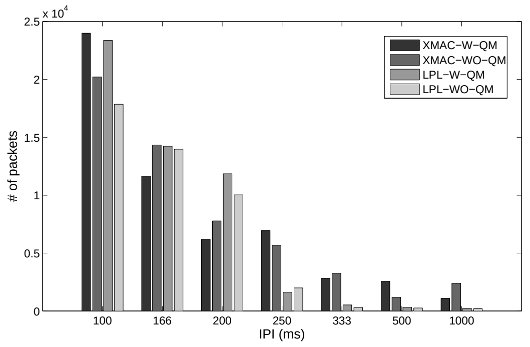


Figure 10: The throughput (the number of packets extracted from the network during its 1-hour operation) for different scenarios.

control period is the same. A control period serves to collect sufficient statistics to estimate inter-packet arrival rates and service rates for the next control period.

Our test network consisted of six TelosB sensor nodes which run TinyOS. Four of these were source nodes which generate packets and their packet generation rate changes over time. A relay node forwarded the packets from these nodes to a base station which was a single-hop away from the relay node. Our model varied the sleep duration of the relay node by observing the packet arrival and departure patters at the relay node. We considered the instantaneous and average power consumption of the network in general and the relay node in particular, packet delivery reliability, and throughput and discrete inter-packet arrivals (100, 250, 333, 500, 750 and 1000 ms.) to evaluate the performance of our model. We operated the network for one hour for each configuration. The experiment results show that the power consumption of the relay node reduced by 11.4% when the inter-packet arrival was small and by 64.8% when it was large. Similarly, the model readily adapted to changes in IPI and increased the packet transmission reliability by 64.5% for small inter-packet intervals and by 67.4% for large inter-packet intervals. The overall throughput of the network also increased by up to 24%.

In future, our aim is to increase the size of the network and to investigate the scalability of our approach and to consider additional protocols into which we can integrate our model

Acknowledgement

This work has been partially funded by the German Research Foundation (DFG) under project agreement: DA 1211/5-1.

References

[1] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *Wireless Sensor Networks* (H. Karl, A. Wolisz, and A. Willig, eds.), vol. 2920 of *Lecture Notes in Computer Science*, pp. 307–322, Springer Berlin Heidelberg, 2004.

[2] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1567–1576 vol.3, 2002.

[3] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, (New York, NY, USA), pp. 171–180, ACM, 2003.

[4] M. Buettner, G. V. Yee, E. Anderson, and R. 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*, pp. 307–320, ACM, 2006.

[5] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 95–107, ACM, 2004.

[6] J. Wang, Z. Cao, X. Mao, and Y. Liu, "Sleep in the dins: Insomnia therapy for duty-cycled sensor networks," in *INFOCOM, 2014 Proceedings IEEE*, pp. 1186–1194, IEEE, 2014.

[7] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele, "ptunes: runtime parameter adaptation for low-power mac protocols," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pp. 173–184, ACM, 2012.

[8] "Tinyos lpl mac.." http://www.tinyos.net/tinyos-2.1.0/doc/html/tep105.html.

[9] W. Dargie and C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[10] R. C. Carrano, D. Passos, L. C. S. Magalhaes, and C. V. Albuquerque, "Survey and taxonomy of duty cycling mechanisms in wireless sensor networks," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 181–194, 2014.

[11] Y. Sun, O. Gurewitz, and D. B. Johnson, "Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 1–14, ACM, 2008.

[12] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pp. 1–14, ACM, 2010.

[13] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pp. 73–84, April 2011.

[14] G. W. Challen, J. Waterman, and M. Welsh, "Idea: Integrated distributed energy awareness for wireless sensor networks," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 35–48, ACM, 2010.

[15] H. Byun and J. So, "Queue-based adaptive duty cycle control for wireless sensor networks," in *Algorithms and Architectures for Parallel Processing*, pp. 205–214, Springer, 2011.

[16] C. M. Vigorito, D. Ganesan, and A. G. Barto, "Adaptive control of duty cycling in energy-harvesting wireless sensor networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*, pp. 21–30, IEEE, 2007.

[17] Z. Li, M. Li, and Y. Liu, "Towards energy-fairness in asynchronous duty-cycling sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 3, p. 38, 2014.

[18] R. B. Cooper, *Introduction to queueing theory*, vol. 2. North Holland New York, 1981.

[19] N. K. Kim and K. C. Chae, "Transform-free analysis of the gi/g/1/k queue through the decomposed little's formula," *Computers & Operations Research*, vol. 30, no. 3, pp. 353–365, 2003.

[20] "Tinyos.." http://www.tinyos.net.

[21] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, *et al.*, "Tinyos: An operating system for sensor networks," in *Ambient intelligence*, pp. 115–148, Springer, 2005.