Adaptive Transmission Scheduling for Energy-Aware Real-Time Wireless Communication

Arda Gumusalan Robert Simon Hakan Aydin
Department of Computer Science
George Mason University
Fairfax, VA 22030
{agumusal, simon, aydin}@gmu.edu

Abstract

Real-time low-power wireless networks are increasingly being used in applications such as: Industrial Internet-of-Things, Smart City technologies, and critical infrastructure monitoring. These networks typically use time-slotted superframe techniques to ensure real-time performance. Often the time slots are assigned statically, and energy conservation is regulated to other mechanisms such as duty cycling. This paper combines real-time performance with novel energy conservation methods by describing a set of dynamic modulation based adaptive packet transmission scheduling algorithms that are designed to reclaim unused slot times. Our approach uses hybrid link access mechanisms. To support our reclaiming method in a wireless environment we introduce a novel low-power listening technique called *reverse-low-power listening* (RLPL) as part of an overall Hybrid Low-Power Listening (HLPL) protocol. We evaluate our algorithms against an oracle-based approach, and show that our dynamic slot reclaiming approach, coupled with HLPL, can introduce substantial power savings without sacrificing real-time support.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Design, Performance

Keywords

Wireless sensor networks, Low-power listening, Dynamic modulation scaling, TDMA

1 Introduction

Industrial control networks (ICNs) have long been used to monitor and manage the real-time activities of a wide range of physical devices, within domains such as manufacturing, electrical generation, and chemical refining [11]. For ease of deployment and reconfiguration, simplified maintenance and lower cost, there is currently tremendous interest in replacing wire-based lCNs with real-time, low power wireless networks running protocols such as IEEE 802.15.4e, WIA-PA, WirelessHART and ISA100.11a [12]. These wireless protocols are also expected to support newer applications such as Smart City and environmental monitoring applications [10].

Low-power real-time wireless protocols typically work by organizing nodes in cluster or star topologies (and sometimes in multi-hop topologies), and then using time-division multiple access (TDMA) transmission scheduling [5]. This is achieved by a cluster-head or gateway coordinator forming a logical time-slotted *superframe*, and then assigning nodes conflict-free transmission slots. The clusterhead coordinates each node's activity during the super frame by transmitting a beacon to all the nodes in the cluster. There have been a significant number of studies focusing on obtaining superframe based real-time wireless communication performance in a variety of environments, including both channel hopping and cluster-tree topologies [15]. However, these studies typically assume that the workload is fully deterministic and known in advance, which is not the case for many newer applications that can be supported by real-time wireless protocols [24]. Then an intuitive question becomes, *is it possible to achieve both real-time performance and energy savings in the face of uncertain workloads?*

This paper aims to answer the above question through the design and analysis of adaptive, superframe based techniques designed to maintain real-time performance guarantees while minimizing energy consumption. One of the advantages of time-slotted superframe approaches is that the nodes can sleep, thereby conserve energy, when it is not their turn to transmit. The basic idea in our approach is to assign nodes time slots in order to meet their communication transmission deadlines, but allow them to *proactively* wake up and determine if other nodes have transmitted all of their packets and no longer require their time slots. If this is the case, a node can begin packet transmission before its scheduled time, and conserve energy by transmitting at reduced modulation levels. In order to accomplish this, we have designed a new low-power listening protocol called *Hybrid Low-Power Listening*. The algorithmic and protocol challenge is to schedule packet transmissions in a manner that reduces energy while maintaining real-time performance, as

International Conference on Embedded Wireless Systems and Networks (EWSN) 2016
15–17 February, Graz, Austria

ISBN: 978-0-9949886-0-7

compared to the traditional static TDMA approach.

Our work makes the following contributions. We first model a basic real-time superframe model, and propose the use of *Dynamic Modulation Scaling (DMS)* in order to opportunistically save energy. DMS saves energy by trading-off the time to transmit a packet with the modulation level of signaling symbol [3, 21, 17]. By increasing (decreasing) modulation levels it takes less (more) time to transmit a bit but more (less) energy is consumed. Despite the wide range of radios available for wireless sensors, the DMS feature is not commonly found on many systems. One exception is CC1200 that supports 2-FSK, 2-GFSK, 4-FSK, 4-GFSK, MSK, and OOK modulations [1].

Under the assumption of a workload that can only be known probabilistically, we then formulate the joint real-time and energy minimization as an optimization problem. Since solving the optimization problem may be computationally complex, we then propose a set of polynomial-time algorithms to address the joint real-time energy optimization problem. To avoid excessive energy expenditures during the times nodes proactively wake up to see if they can prematurely transmit, we propose and analyze a novel Hybrid Low-Power Listening (HLPL) protocol. HLPL incorporates a new technique called *reverse-low-power listening* (RLPL). RLPL is a twist on traditional low-power listening (LPL) protocols, which are well-known methods used in low data rate duty cycling wireless sensor networks [16]. LPL protocols yield significant energy savings. To our best knowledge this is the first usage of a hybrid LPL technique in a joint TDMA-based real-time energy savings protocol, as well as the first one that combines beacon-enabled superframe concept and low power listening.

Using HLPL, we evaluated our optimal and heuristic algorithms against an oracle approach, which has perfect workload knowledge, under a number of workload and deadline constraints. The results show that the hybrid HLPL approach, coupled with DMS, can achieve significant energy savings while maintaining real-time performance, as opposed to the traditional TDMA method.

2 Related Work

The beacon-enabled superframe concept is proposed as an amendment to IEEE 802.15.4 standards and included in 802.15.4e [2], which is aimed towards real-time industrial systems with the assumption that this concept can provide real-time guarantees for wireless sensor networks (WSNs). This standard defines contention-access-period, contention-free-period and guaranteed-time-slot concepts. This basic approach is incorporated in industrial standards such as WirelessHART, ISA 100.11a and WIA-PA [12, 23]. Our work is fully compatible with these standards.

These standards emphasize the necessity of real-time performance in wireless networks. Our work contributes to these by adopting dynamic time slot allocation and adjustment on-the-fly for generic superframe structure. Moreover, we introduce a novel protocol called Hybrid Low-Power Listening (HLPL) as an efficient technique to eliminate the impact of neighborhood for superframe structures by combining two seemingly contradicting ideas such as superframe and low-power listening.

Our work incorporates TDMA, CSMA, and LPL MAC layer protocols; hence, it can be categorized as a hybrid MAC layer protocol. Hybrid MAC-layer approaches have been studied for a number of years [14]. Some well-known examples are Z-MAC, HyMAC, H-MAC, ER-MAC, and Queue-MAC [18, 19, 13, 22, 28]. Our work differs from these because of our introduction of the HLPL concept which allows efficient on-the-fly slot adjustments (even in the existence of interference) and our use of DMS to save energy while maintaining transmission deadlines.

Low-power listening (LPL) is a commonly used MAC-layer protocol that reduces the energy consumption caused by idle listening to the channel for an activity. In LPL, nodes periodically wake up to detect the activities in the channel. LPL techniques are generally categorized as either *sender-initiated, receiver-initiated*, or *hybrid*. Another classification line considers *synchronous* or *asynchronous* nature of the solution. Our proposed HLPL protocol is a sender-initiated, asynchronous LPL and includes a new technique to address high false-alert rates caused by overhearing observed in the traditional LPL protocol frameworks [7].

In our system DMS is simply a control knob that can be replaced with any other energy saving mechanism but it is an important concept for energy minimization and worth mentioning here. One of the earliest papers that applied DMS to real-time traffic is [20]. That work has developed the concept of adjusting modulation scaling on-the-fly for general real-time purposes. However, WSN was not the main focus of the paper. The authors in [25] studied the application of DMS on data gathering scheduling of wireless sensors in a real-time scenario. They have shown that DMS can achieve up to 90% energy savings. However, they have assumed the same constant packet workload for each node in the network. Our work differs from the above by considering a *probabilistic* workload and applying DMS to Time Division Multiple Access (TDMA) based scheduling. We believe non-deterministic workload is more realistic for many applications of wireless sensors and is worth investigating. DVS/DMS joint scheduling for probabilistic workloads is considered in [3], but only for a simplified environment with no interference and no co-scheduling of the nodes.

3 System and Communication Model

3.1 Application Requirements Model

Our work focuses on nodes that form single-hop communication clusters. Each node is assumed to periodically generate some number of packets that it must transmit by a specific deadline. The actual number of packets changes over time, and is only known probabilistically. In order to meet these requirements we use a generic, beacon-enabled superframe architecture for real-time communication. We assume every node participating in the cluster can directly communicate with the coordinator. We also assume that the coordinator or clusterhead is not power-limited.

Our model is shown in Figure 1. As it is seen, at the beginning of each superframe the beacon is transmitted by the coordinator. The beacon contains management information such as TDMA slot assignments, and is received by all the

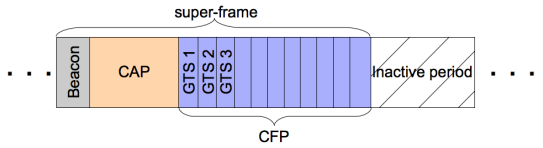


Figure 1: The lay-out of superframe

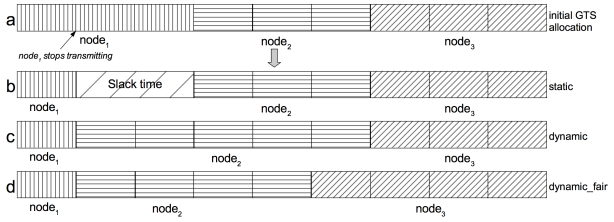


Figure 2: Illustration of offline and online slack reallocation

nodes in the cluster. This is followed by a contention access period (CAP), allowing each node to talk to the coordinator via CSMA. During this phase a node may send future workload information, or may ask to join or leave the cluster. The contention free period (CFP) starts right after the coordinator sends out a beacon. CFP consists of a series of Guaranteed Time Slots (GTSs), which are assigned to specific nodes. In order to provide real-time communication guarantees, each node is assigned a number of GTSs equal to its worst-case traffic workload.

The coordinator manages GTS assignments to avoid collisions and provide real-time guarantees. CAP can be followed by an inactive period during which the nodes can sleep. A traditional approach is to have a node sleep during the GTSs assigned to other nodes. In this paper, we introduce a Hybrid Low-Power Listening protocol that allows nodes to proactively remain awake during time slots assigned to other nodes to attempt to opportunistically transmit their packets at a reduced energy level.

3.2 Communication

We assume that each node is equipped with a DMS-enabled radio capable of dynamically adjusting the modulation levels. We adopt the basic energy model presented in [21]. Specifically, the radio power consumption is divided into two parts. The first is *transmission power*, denoted as p_s, and the second is *electronic circuitry power*, denoted as p_e. These values can be expressed as $p_s = C_s \times \phi(b) \times R_s$ and $p_e = C_e \times R_s$, respectively. Here, R_s is the number of symbols transmitted per second and b is the modulation size. The values C_e and C_s are radio-specific; but C_s can be affected by the current environmental conditions, such as atmospheric noise, transmitter-receiver distance and temperature. In practice C_e and C_s can both be approximated as constants. Finally $\phi(b)$ is the convex scaling function of the modulation used depending on the modulation scheme. For QAM, $\phi(b)$ function is $2^b - 1$ [21], which shows the exponential increase in power consumption in terms of the modulation level (p_e is assumed to be constant). The transmission time, on the other hand, is $\frac{1}{b \times R_s}$ which decreases linearly in terms of modulation level. As a result, the tradeoff of DMS becomes exponential increase (decrease) of transmission power with linear decrease (increase) of transmission time for QAM [21].

Our HLPL protocol uses two schemes commonly used in asynchronous duty-cycled low-power MAC protocols, namely low-power listening (LPL) and embedding information in short preamble (physical layer) packets. A typical LPL protocol such as B-MAC [16] requires a sender to transmit a long preamble. Receivers wake up, sense the preamble, and stay awake to receive data. The duration of listening and sleeping schedules can be adjusted to, for example, maximize energy savings, maximize throughput, or minimize delay. Protocols such as X-MAC [4] extend this idea by using preambles to embed information such as receiver target addresses. The advantage of using preamble addressing is a reduction in the number of bits that need to be transmitted and a flexible, user configurable information, since most radios allow the preamble sequence to be reprogrammed.

4 Hybrid Low-Power Listening (HLPL)

The advantage of DMS is that the nodes can minimize overall energy consumption by using lower modulation levels. The drawback is that lower modulation levels require longer periods of time to transmit the same number of bits. In our environment this means that more GTSs are required. We assume that the nodes can cap their worst-case workload in terms of the number of packets to send, but the actual distribution is only known probabilistically. This means that the nodes may send fewer packets then their worst case estimate. Hence, a node might not use all of the GTSs assigned to it. It is therefore desirable to devise *dynamic* algorithms capable of assigning these unused slots to other nodes. Further, it is possible to use DMS techniques to extend transmission over unused slots, in order to reduce their energy expenditures, as long as the deadlines are maintained. We call the extra time available from unused slots "slack time".

Figure 2 shows how different algorithms may behave when slack time is available. Assume the initial GTS assignments are shown in the superframe labeled a. Here $node_1$ has been assigned 3 packet-length GTS but it only transmits a single packet. The next superframe, labeled b, shows a static approach. The GTS assignments for $node_2$ and $node_3$ do not change, and the available slack time remains un-utilized. The superframe c shows a dynamic approach where these slots are reallocated to $node_2$, which can lower its modulation level but still meet the deadline. A dynamic and fair approach, shown in superframe d, shows how the slack could be allocated to both $node_2$ and $node_3$. Detailed descriptions of these algorithms along with methods for determining modulation levels for the new GTS distribution are provided in Section 6.

For dynamic algorithms to succeed, the nodes need to be aware when the currently scheduled node prematurely finishes transmission. Our approach works by having the coordinator broadcast a relatively short **preamble** that contains the address of the next node to transmit and the modulation levels that the node will use. Nodes hear this preamble by using low-power listening. The selected node may in fact be granted permission to transmit early using the available slack

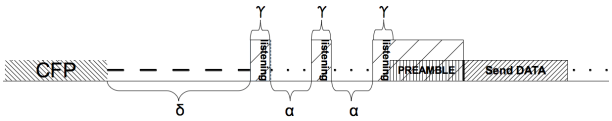


Figure 3: Parameters of the Sleep-Listen Cycle

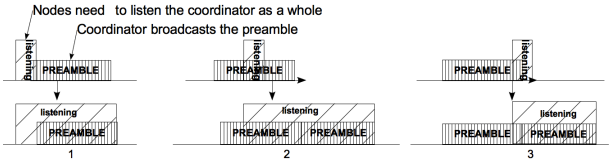


Figure 4: Possible intersection of listening period and preamble

time.

Figure 3 shows the parameters of the listen-sleep cycle. Here δ shows the *wait-duration* before the node starts its low-power listening (LPL) cycle. The node is entirely asleep during the period δ. For the LPL phase, we use parameters α and γ, referred to as the *sleep-duration* and *listening-duration*, respectively. γ is the time during which the node is listening to the medium whereas α is the time during which the node is in sleep mode.

The length of the coordinator's preamble has to be greater than the LPL period of $\alpha+\gamma$. $L_{preamble} \geq \alpha+\gamma$ guarantees that the receiver will hear the preamble. However, it does not guarantee that the receiver will listen to the preamble as a whole. Figure 4 shows the possible intersections of the LPL period and preamble. Among these three possibilities, the first one is desired, since the receiver receives the entire preamble. In the second and third cases, the receiver will hear the preamble; however, it will not know who the preamble is addressed to. As a result, the receiver will need to keep listening even after the preamble message is over, in order to learn the address of the preamble. This adds to the power consumption of the receiver. The coordinator needs to make sure that after sending the preamble message, the intended node starts transmitting. If not, the coordinator needs to re-send the preamble. We evaluate the effect of these parameters in Section 6.

An additional problem exists in that the absence of transmission activity being detected by a node does *not* necessarily mean that nodes are not transmitting in the cluster. Two nodes may be entirely out of each others' radio range. Another possibility is that a node may be in another node's interference range, but not its transmission range. This means that a node can hear another node transmit but cannot decode the transmission. Also, when a node is in the transmission range of another node, it overhears the communication. However, nodes are only interested in transmissions from the coordinator. Listening to the other nodes in addition to the coordinator increases the energy consumption. For our scheme, the practical impact is that a node may not be able to hear another node that is in the process of sending a packet to the coordinator, and therefore cannot tell if the slot is used or idle. Also, it may overhear the unintended communications

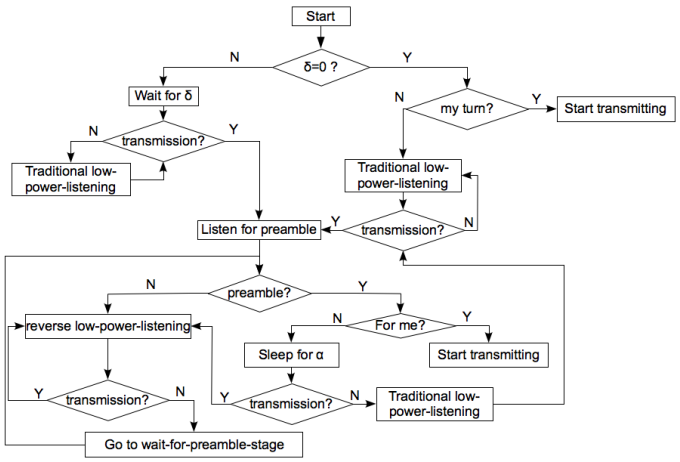


Figure 5: The steps of HLPL

with additional energy cost. We refer to these issues as the *neighborhood* problem.

In order to overcome the *neighborhood* problem, we define the hybrid-low-power listening (HLPL) protocol. HLPL is a combination of traditional low-power listening and a new scheme called *reverse-low-power listening* (RLPL). On the receiver side, the node needs to decide which LPL mode it needs to be in. In HLPL, if a node receives a preamble and learns that it is not scheduled next **and** it senses any transmission during its *first* wake-up after this preamble, the node goes into the RLPL (described below) stage. For non-zero wait-duration values, when the node wakes up, it listens to the channel for a preamble. If it senses any transmission and this transmission is not a preamble then the node goes into the RLPL stage. However, if this transmission is a preamble that is not addressed to itself and if it does not hear any transmission during its first wake-up after the preamble, it goes into the traditional LPL stage.

Further, for zero wait-duration values, the nodes (except for the first scheduled node) start with traditional LPL. During listening phases if they do not hear any transmission, they stay in the tradition LPL stage. However, when a node senses a transmission after a preamble, it goes into the RLPL stage. The logic behind this process is the fact that hearing a transmission during the first wake-up right after the "false" preamble indicates that there is an interference since the sleep-duration is smaller than length of a single packet. If the node does not hear any transmission after a preamble addressed to another node means there is no interference. On the sender side, the coordinator waits for sleep-duration amount of time before it broadcasts the preamble, unlike in traditional LPL, where a sender broadcasts the preamble as soon as the current node stopped transmitting. Waiting for *sleep-duration* amount of time ensures that all the nodes that are in RLPL mode are currently in the *wait-for-preamble* stage. Figure 5 shows the flow chart for HLPL.

RLPL differs from traditional LPL in its conditions to transit between listening and sleeping stages. In RLPL when the node wakes-up and hears a transmission, it goes back to

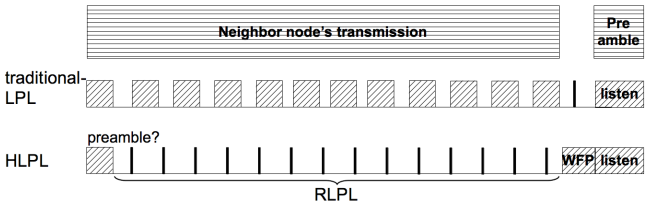


Figure 6: Comparison of traditional LPL and RLPL

sleep. However, if it does not hear any transmission, it starts listening, which is different than the traditional low-power listening. In RLPL, this listening phase is called *wait-for-preamble* stage. As explained, when a node stops transmitting, the coordinator waits for *sleep-duration* amount of time before it broadcasts the preamble. Hence, *wait-for-preamble* stage can last at most for sleep-duration time. *Wait-for-preamble* guarantees that when the coordinator broadcasts the preamble, the nodes will be listening to the channel.

The core idea of HLPL is to save energy when there is *constant* traffic in the network. In the absence of this, HLPL behaves very similar to traditional LPL. Figure 6 aims to clarify the difference between traditional LPL and RLPL during a constant traffic. Under traditional LPL, the node wakes up periodically and each time it senses a transmission. Then it stays awake long enough to conclude that the transmission is not from the coordinator. On the other hand, under RLPL, the node first listens to the channel, realizes that it is not a preamble, and goes to the RLPL stage. With RLPL, a node still periodically wakes up but stays awake enough to detect that there is *some* transmission. If so, the node goes back to the sleep mode. Otherwise, the RLPL stage concludes, and the node transitions to the wait-for-preamble stage. It stays in that stage until it receives the preamble.

5 Joint Deadline-Energy Optimization problem

Based upon the number of nodes, the real-time constraints, and the actual workload, the question remains how to set the modulation levels to achieve all deadlines and conserve energy. We now show how to formulate this question as an optimization problem.

Earlier research in DMS has shown that there exists a *constant* optimal modulation level that minimizes the energy consumption while meeting all deadlines under *deterministic* workloads [21]. However, in [3] observed that under probabilistic workloads, this is not the case. Instead, the optimal solution to minimize the *expected* energy consumption consists in transmitting the first packets at low speed (modulation), and increasing the speed gradually for the subsequent packets when approaching the deadline. This is based on the observation that in the more likely scenarios where the actual workload deviates from the worst-case, low modulation levels are sufficient to meet the deadline while saving significant energy. However, as more packets are transmitted, the modulation level is gradually increased to meet the deadline. The framework to find the optimal modulation levels given a deadline and probabilistic workload profile is called *speed scheduling* in [3] and we also adopt this approach.

Table 1: List of symbols

symbol	description
n	number of nodes
m_i	upper limit on the number of packets $node_i$ can send
D	length of superframe
$p_i(k)$	the probability that $node_i$'s workload is k packets
$a_i(k)$	the probability that $node_i$ sends k or more packets
$b_i(k)$	the modulation level used by $node_i$ to send its k^{th} packet
L	the maximum transmission unit of the underlying communication protocol
t_{bit}	the time to send a bit
t_{symbol}	the time to send a single symbol
R_s	symboling rate
b_{min}	the minimum modulation level that a node can use
b_{max}	the maximum modulation level that a node can use
$\beta_l^{i,k}$	the binary indicator equals 1 for the selected modulation level l for $node_i$'s k^{th} packet
p_e	power consumption of the electronic circuitry
p_s	power consumption from transmission

In our application, each node has a varying communication workload determined by a known probabilistic distribution. The $node_i$ can have from 1 to m_i packets to transmit in a given superframe. $p_i(k)$ represents the probability distribution function of $node_i$'s workload. Specifically, $p_i(k)$ denotes the probability that $node_i$ will transmit exactly k packets during a superframe.

The energy needed to transmit a single packet, e_{packet}, is the product of time to send a single bit (t_{bit}), the length L of the maximum transmission unit (in bits), and the total power $(p_s + p_e)$. Moreover, $t_{bit} = \frac{1}{b.R_s}$ where b indicates the modulation level and R_s is constant. A typical value for R_s is 62500 symbols/second for 802.15.4 [8]. By using the radio power consumption formula from Section 3, we get:

$$e_{packet} = L \cdot (p_s + p_e) \cdot t_{bit} = \frac{L \cdot (C_s \cdot \phi(b) + C_e)}{b} \tag{1}$$

The total *expected* energy consumption is the sum of expected energy consumption of n nodes:

$$e_{expected} = \sum_{i=1}^{n} \sum_{k=1}^{m_i} e_{packet} \cdot a_i(k) \tag{2}$$

In Equation (2) $a_i(k) = \sum_{x=k}^{m_i} p_i(x)$ indicates the probability that node i will actually transmit the k^{th} packet. By denoting the modulation level of the k^{th} packet of the node i by $b_i(k)$, we obtain:

$$e_{expected} = \sum_{i=1}^{n} \sum_{k=1}^{m_i} \frac{L.a_i(k)}{b_i(k)} . [C_s . \phi(b_i(k)) + C_e] \tag{3}$$

Note that the k^{th} packet of node i can potentially be transmitted with any of the discrete modulation levels in the range $[b_{min}, .., b_{max}]$. Let $\beta_l^{i,k}$ be a binary indicator variable $\in \{0,1\}$ to represent whether the k^{th} packet of node i is transmitted using the modulation level l or not. Then an integer programming formulation to minimize the expected energy can be obtained as:

$$\text{minimize} \quad \sum_{i=1}^{n}\sum_{k=1}^{m_i}\sum_{l=b_{min}}^{b_{max}} \beta_l^{i,k} \cdot \frac{L \cdot a_i(k)}{b_l} \cdot [C_s \cdot \phi(b_l) + C_e] \tag{4a}$$

$$\text{subject to} \quad \sum_{l=b_{min}}^{b_{max}} \beta_l^{i,k} = 1 \ \forall i,k \tag{4b}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m_i}\sum_{l=b_{min}}^{b_{max}} \beta_l^{i,k} \cdot \frac{L}{b_l.R_s} \leq D \tag{4c}$$

$$\beta_l^{i,k} \in \{0,1\} \ \forall i,k \tag{4d}$$

The objective function gives the sum of the energy consumption of all the packets over all the nodes, by considering their probability of being transmitted and all possible modulation levels. The constraints (4b) and (4d) indicate that exactly one modulation level will be assigned to each packet in the workload. The constraint (4c) enforces that all the modulation levels must be selected in a way that all the transmissions will be completed before the deadline (the end of the superframe). Although integer programming problems are known to be intractable in the general case, moderate-size instances can be solved using the existing optimization tools such as CPLEX.

6 Performance Evaluation

To evaluate the performance of the several variants of the proposed framework under different workload conditions, we developed a Matlab based simulation system. We simulated a system with a coordinator and 10 nodes arranged in star topology, and with communication range set to $r_{communication} = 30$m. The work done in [9] shows that DMS is effective for distances greater than 25 meters. Each node's workload in a superframe varies between 1 to 10 packets and is derived from a probability distribution. We assumed DMS-capable systems (with QAM modulation) where the modulation levels can vary from 2 to 8.

We ran various simulations for different superframe lengths (deadlines) to analyze how the energy consumption varies. The minimum deadline D_0 is assumed to be the superframe length necessary to allow the transmission of the worst-case workload (10 packets) by each node at the maximum modulation level, considered to be equal to 208 ms[1]. The actual deadline for a given experiment is then computed as $D = \frac{D_0}{load}$ where the system's *load* is in the range $[0.1, 1.0]$.

We implemented the following algorithms:

i.) Oracle: This is the *yardstick* algorithm where the exact number of packets that each node will transmit is known in advance, at the beginning of each superframe. As a result, it does not need to assume the worst-case workload. This scheme does not require any LPL because it knows the exact time each node will stop transmitting. Hence, the overhead of LPL is also omitted. Although it is not a feasible algorithm in practice, it provides the minimum energy consumption that is theoretically possible for a given experiment.

[1] As in low-power listening mode each node can miss up to 2 preambles before it can start transmitting, this duration as well as the transmission delay are included in D_0 to ensure feasibility.

ii.) Static: In this algorithm, assuming the worst-case workload (i.e., 10 packets) for every node, the smallest possible modulation level with which the deadline can be met is assigned to all the nodes in uniform manner statically. The assigned modulation levels do not change for the duration of the superframe, even though the actual workload of a node may deviate from the worst-case (i.e., slack is not reclaimed).

*iii.) Static**: This is similar to *Static* algorithm in the sense that the modulation levels are computed statically and slack is not reclaimed. However, instead of assigning a constant modulation level to every node, the nodes use a speed schedule that gradually increases the modulation levels by exploiting the probabilistic workload profile. This is computed by solving the integer programming problem with the objective function given in Equation (4a).

iv.) Dynamic: This algorithm makes initial modulation level assignments as in *Static* but then dynamically adjusts the modulation levels in order to take advantage of the available slack time after the end of each node's transmission and allows only the following node to reclaim this slack time by adaptively reducing the modulation level. At slack reclamation times, each node uses the smallest feasible modulation level to use the duration of its originally allocated slots and reclaimed slots.

*v.) Dynamic**: This algorithm enables dynamic reclaiming of the unused slots by adaptively reducing the modulation level at run-time. However, the initial modulation levels are computed using *Static** and the node that reclaims the slack uses the speed scheduling solution to re-assign possibly different modulation levels to each of its packets.

vi.) Dynamic_f: The *fair* version of the *Dynamic* algorithm in the sense that the available slack time is distributed evenly among *all* subsequent nodes rather than being assigned entirely to the next node. The modulation levels of all the subsequent nodes are dynamically adjusted after the end of each node's transmission to the lowest feasible modulation level.

Static, *Dynamic*, and *Dynamic_f* are polynomial-time algorithms. They only iterate over each modulation level (from 2 to 8) once and select the minimum feasible one. *Static** solves the Binary-Integer Programming Problem introduced in Section 5 but it is executed offline by the coordinator and only once unless the probability distribution changes. *Dynamic** also solves the same Binary Integer Problem but only for a single node. In practice, a look-up table can easily be constructed with the pre-computed modulation levels as a function of available slack.

Consider the example of 5 nodes with maximum workload of 10 shown in Figure 7. Initially, each node is assigned slots with total length equal to 10 packets with the modulation level b where $b > b_{min}$. When it is $node_1$'s turn, it sends 6 packets using modulation level b which yields a slack time of four packets long. *Dynamic* and *Dynamic** allocate this slack time to the next scheduled node, namely $node_2$. Now, $node_2$ has effectively additional slots, giving a transmission time equal to 14 packets. However, $node_2$ will transmit at most 10 packets so it can reduce its modulation levels. In the case of *Dynamic*, $node_2$ uses the lowest feasible modulation, b_D, where $b_D < b$ for each of its 10 probable pack-

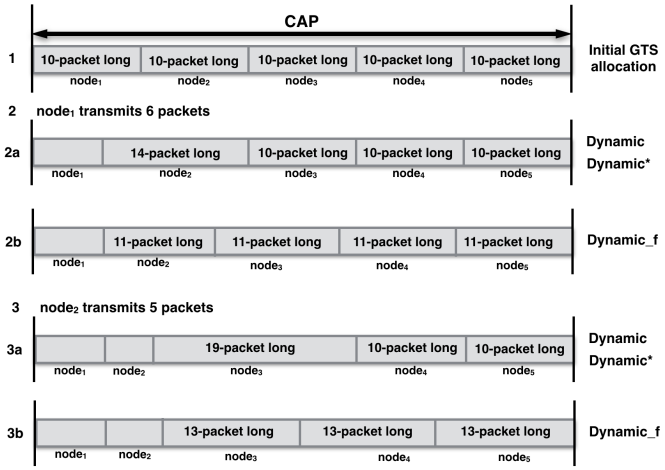


Figure 7: An example for dynamic algorithms

ets. *Dynamic** uses the optimal modulation levels computed by solving Equation (4a) only for its probabilistic workload and slot length. When it is $node_2$'s turn, it ends up transmitting 5 packets implying there is a slack time of 9 packets with modulation b. Similarly, *Dynamic* and *Dynamic** assign this slack time to the next scheduled node, namely $node_3$. In the *Dynamic* case, $node_3$ uses the lowest feasible modulation level, $b_{D'}$, where $b_{D'} \leq b_D \leq b$. $node_3$ uses the optimal solution computed for its own packets with its own deadline. In the *Dynamic_f* case, the 4-packet long slack time after $node_1$'s transmission is distributed among $node_2$, $node_3$, $node_4$, and $node_5$. These nodes have 11-packet long slack time with the modulation level b. The lowest feasible modulation level, b_{D_f}, that will meet with the deadline with 40 possible packets is computed where $b_{D_f} \leq b$. After $node_2$ stops transmitting, the 9-packet long slack is distributed among $node_3$, $node_4$, and $node_5$. The new lowest feasible modulation level $b_{D'_f}$ for all 30 possible packets to meet the deadline is computed where $b_{D'_f} \leq b_{D_f} < b$.

For each *load* value, our simulator generated 300 workload instances for 10 nodes using a discretized Normal distribution function with the mean of $\mu = 5$ packets and the standard deviation $\sigma = 2$. We also ran experiments with Uniform, Pareto and Flipped-Pareto distributions. We will present detailed simulation results for Normal distribution and underline the trends and relative ordering of the schemes for the remaining distributions.

All the proposed algorithms run on the coordinator. The computed modulation levels are transmitted to the nodes using beacons and preambles. Hence, the computation overhead at the nodes is minimal. We set the preamble size to 14 bytes, which includes a 1 byte for sender address, a 1 byte for receiver address, 10 bytes for the calculated modulation levels, and 2 bytes for the CRC footer. For the *Static*, *Static**, and *Oracle* algorithms the beacon size is set to 124 bytes; while it is 24 bytes for the *Dynamic*, *Dynamic**, and *Dynamic_f* algorithms.

All the dynamic algorithms require same amount of signaling. Furthermore, we assumed the coordinator uses the maximum modulation level to broadcast the preamble. In our simulation settings, the *listening-duration* $\gamma = t_{symbol}$ and the *sleep-duration* $\alpha = t_{preamble} - \gamma$ where t_{symbol} is the time necessary to listen to 8 bits and $t_{preamble}$ is the time needed to transmit the preamble with modulation size of 8. For the C_e and C_s values described in Section 3, we have used 15×10^{-9} and 12×10^{-9} Joules, respectively, and $b_{min} = 2$, $b_{max} = 8$, after [3, 21]. All the simulation results are presented at 95% confidence level. In all the plots presented, the energy consumption values of various schemes are normalized with respect to the energy consumption of *Static* at $load = 1.0$.

6.1 Analysis of the Ideal Case

In this section we evaluate the proposed algorithms' ideal case performances. In *ideal case*, we assume that the nodes have exact knowledge about the time at which they need to wake up, in advance. The static algorithms can incorporate this information in the beacon message. For dynamic algorithms we assumed the same beacon message structure. Obviously in this ideal case, the need for low-power listening disappears. Still, we believe the analysis of this case reveals some important patterns because it points to the upper bounds on the energy savings that each algorithm can provide with zero-overhead low-power listening.

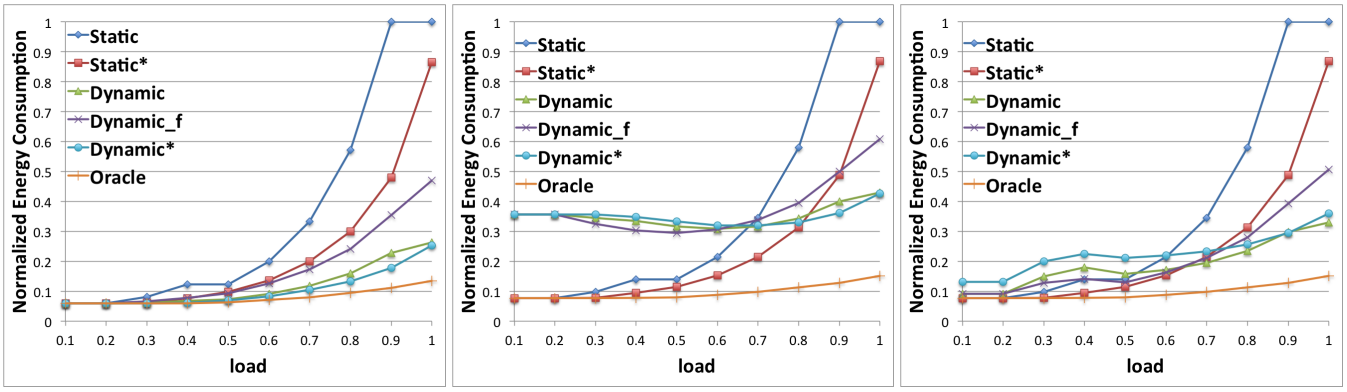
Figure 8a shows the normalized energy consumption of the proposed algorithms. We observe that on higher load values the *Dynamic*, *Dynamic**, and *Dynamic_f* algorithms give significant energy savings compared to *Static* and *Static** algorithms. Moreover, *Dynamic* and *Dynamic** perform better than *Dynamic_f*. However, at lower load values, the dynamic algorithms provide only limited gains; this is because even the static algorithms are able to assign low modulation levels when the system has ample time to finish the workload.

It is important to compare *Dynamic* and *Dynamic**. The winner algorithm is *Dynamic** as expected; but *Dynamic* comes very close. Recalling that solving the integer programming formulation for *Dynamic** may be computationally expensive, we can reach the conclusion that using *Dynamic* may be a reasonable approximate solution in practice.

6.2 Analysis of the effect of traditional LPL with no interference

In this section, we will show the effect of low-power listening on the proposed algorithms. The ideal case where the nodes know exactly when the previously scheduled node stops transmitting cannot be implemented in real-life scenarios. The nodes need to listen for a preamble from the coordinator to see when they can start transmitting. One possibility is to use the traditional low-power listening (without the HLPL enhancement described in Section 4) and our results in this section consider this case, by further assuming that the cross-node interference is negligible. In Section 6.3, we will re-analyze these settings within the HLPL framework, and by considering the impact of the interference.

Figure 8b shows the normalized energy consumption of greedy low-power listening enabled algorithms. The compared algorithms are *greedy* in the sense that they use traditional low-power listening with the wait-duration δ set to zero. We need to recall that only *Dynamic*, *Dynamic**, and



(a) Energy consumption in the ideal case (LPL not needed)

(b) Energy consumption of greedy-LPL where $\delta = 0$

(c) Energy consumption of smart-LPL where δ =expected-wait-time

Figure 8: Simulation results with no interference

Dynamic_f require low-power listening. The remaining algorithms have pre-determined wake-up times. We can see that the dynamic algorithms perform poorly compared to static algorithms when $load \leq 0.7$. This is due to the fact that for lightly loaded systems, the gain from dynamic reclamation of the slack times is offset by the additional energy consumption due to energy overhead of traditional low-power listening activity. The dynamic algorithms' energy performance improves only when *load* approaches and exceeds 0.7 – this is when the overhead of low-power listening (necessary to implement the reclaiming mechanism) becomes resonably low compared to the gains of adaptive modulation downscaling at run-time.

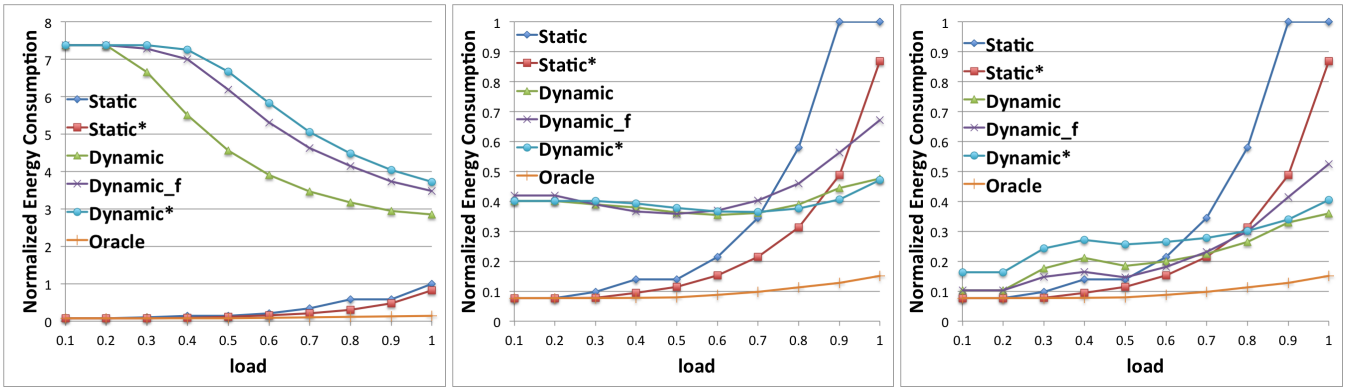
Another possibility for the implementation of the traditional low-power listening in these settings is to have each node wait for a time duration δ equal to the *expected time needed for the completion of the packet transmissions by the previous nodes.* The idea is to take advantage of the known probability distribution. Rather than letting nodes start low-power listening as soon as the collision-avoidance-period starts, the nodes calculate the expected number of packets that will be transmitted by the previously scheduled nodes based on the known probability distribution function. We call this scheme *smart-LPL*. The expected-number-of-packets before $node_i$ can start to transmit is $\sum_{k=1}^{i-1}\sum_{l=1}^{m_l} p_k(l) \times l$. The scheduling order is embedded into the beacon message. Two observations are in order here: *i*) if the node starts low-power listening before its actual turn then the node spends more energy for low-power listening but does not miss any of its slack time. However, if the node wakes up after its turn starts then the node loses some portion of the given slack time (the node could not reduce its modulation levels as much as it could have) but spends less energy on low power listening. Hence, there is a trade-off between the gain from low-power listening and loss from smaller slack times. *ii*) The modulation level assumed in the calculation of the expected-wait-time for the previous nodes is another critical variable. The nodes know the expected number of packets to be sent before their turn, but they do not know what modulation levels have been used by the previous nodes (they cannot accurately map the expected number of packets to the expected amount of waiting time). In our simulation settings, we have used the modulation level calculated by *Static* to compute the expected-wait-time values.

Figure 8c illustrates the normalized energy consumptions with smart-LPL. Our analysis reveals that with smart-LPL, the energy consumption of the dynamic algorithms is reduced compared to the greedy-LPL case.

One important observation here is the effect of low-power listening on the ordering of the dynamic algorithms. In Figure 8b and Figure 8c, for the load value roughly less than 0.7, *Dynamic_f* performs better than *Dynamic*, which outperforms *Dynamic**. This is in contrast to the ideal case results (Figure 8a). This is a result of two factors: *i)* lower energy consumption in ideal case means the transmissions are taking longer time which leads to higher overhead created by low-power listening; *ii)* For the case shown in Figure 8c, the expected-wait-time values are calculated using the modulation level given by the *Static* algorithm. However, higher ideal case performance implies that the previously scheduled nodes have used smaller modulation levels than initially computed. This leads to less accuracy in predicting expected-wait-time and as a result, a longer duration for traditional low-power listening. This is a crucial result that shows how dynamic modulation levels can a ffect (up to the point where the ordering of algorithms change) low-power listening and becomes one of the fundamental reasons necessitating the use of HLPL protocol.

6.3 Analysis of the impact of neighborhood/interference and HLPL

In this section, our aim is to evaluate the effect of neighborhood/interference on traditional low-power listening and also include our newly proposed HLPL in the comparison. In real settings when a node wakes up to check for a preamble, it has to listen to its neighbors' communications to make sure that the communication it is sensing is not a preamble. In order for a node to make sure that it is not receiving a preamble, it may need to listen the channel for up to two



(a) Energy consumption with greedy-LPL and interference

(b) Energy consumption with greedy-HLPL where $\delta = 0$ (with interference)

(c) Energy consumption with smart HLPL where $\delta =$ expected-wait-time (with interference)

Figure 9: Impact of interference

preamble transmission times as shown in Figure 4. We used the neighborhood size as 4 in our simulations, assuming the interference range $r_{interference} = 1.2 \times r_{communication}$ after [27].

Figure 9a shows the normalized energy consumption with traditional low-power listening and possible interference. A striking observation is the significantly increased energy consumption of the dynamic algorithms for most of the spectrum, due to the prohibitive energy consumption of *false alerts* induced by the interference due to the naive application of the traditional low-power listening framework. In this case, the nodes receive *false alerts* from their neighbors and they need to verify the content of these transmissions. The length of preamble message is 14-byte long whereas the MTU of 802.15.4 is 127 bytes. This indicates that even for a single packet with the highest modulation level, the node has to consume an additional energy of listening up to $2/3$ of a packet (which is 84 bytes) to see if there is or there is not a preamble addressed to itself. In a neighborhood of size 4, this may create and additional overhead up to 26 packets per node. For lower modulation levels, this number is even higher. As can be seen from Figure 9a, the energy consumption of the dynamic algorithms tends to go down with the higher utilization factor. This is due to the fact that, in lower utilization factors, the nodes use lower modulation levels and longer transmission times which significantly increases the interference caused by the neighbors.

The overhead created by the interference also depends on the values used for sleep-duration and *preamble size*. In our simulations, we have observed that larger *sleep-duration* values tend to decrease the overhead induced by the interference. However, longer sleep-duration has other consequences such as longer superframe lengths and larger losses in the available slack times. In order to ensure the deadlines, we have to account for the maximum time a node can miss before it hears a preamble. This maximum time needs to be added to the minimum feasible superframe length to ensure the feasibility of the system. Some optimal values of preamble length and sleep-duration values that will minimize this overhead may exist. However, we believe even this minimized overhead will still be undesirable especially for lower utilization factors where *offline* algorithms perform well. Finding this minimized overhead value is left as a future work.

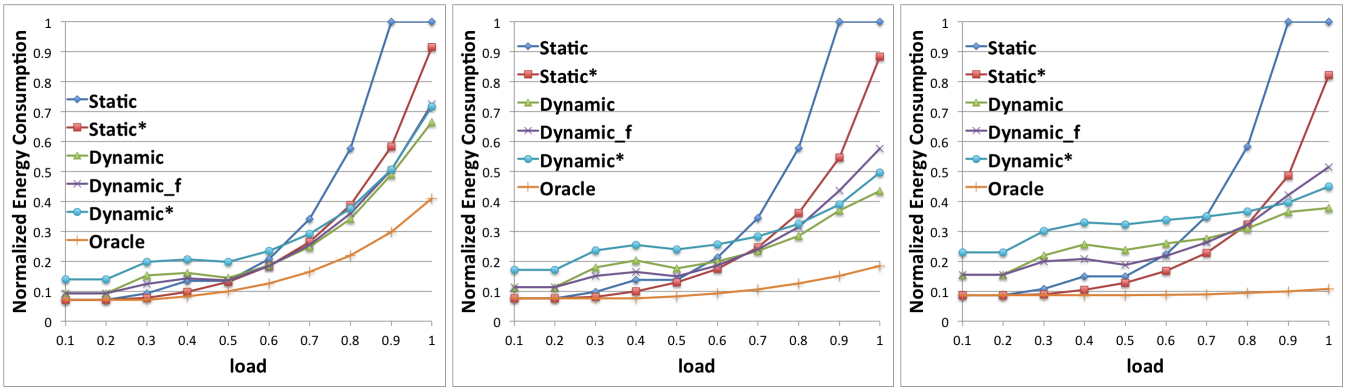
Figure 9b shows the simulation results obtained after adopting greedy-HLPL. Comparing to Figure 9a, one can see the drastic energy savings provided by the greedy-HLPL. *Dynamic* and *Dynamic** outperform the *Static* algorithm for load values higher than 0.7. For load value 0.85 and higher, we observe that *Dynamic* and *Dynamic_f* have less energy consumption than *Static**. *Dynamic_f* outperforms *Static** for load values roughly after 0.93. If we compare Figure 9b with Figure 8b, we can see that the performance of dynamic algorithms in the presence of interference is rather close to the one in the no-interference case. Figure 9c shows the normalized energy consumption of smart-HLPL when wait-duration is equal to expected-wait-time. This case further reduces the overall energy consumption of dynamic algorithms. In this case, *Dynamic* outperforms the static algorithms for load values roughly larger than 0.70.

As can be seen, HLPL successfully addresses the neighborhood/interference problems and yields significant energy savings. Finally, the comparison of Figure 8a with Figure 9c shows that the results of HLPL are reasonably close to the ideal case, showing the potential of the framework.

6.4 Effect of different probability distribution functions

All the results presented so far were obtained under the Normal probability distribution. We have also repeated all the simulations with Uniform, Pareto and Flipped-Pareto distributions with $k = 10$ (shape parameter), $\sigma = 3$ (scale parameter), and $\theta = \frac{10}{3}$ (threshold value).

An important difference is in terms of the average energy consumption of different distributions. Our simulation results show that the Pareto distribution has the lowest average energy consumption followed by Normal, Uniform, and Flipped-Pareto distributions. This is expected due to the fact that each distribution function has different expected workload figures which are 3.22, 5.04, 5.5, 7.78 for Pareto, Normal, Uniform, and Flipped-Pareto distributions, respectively.



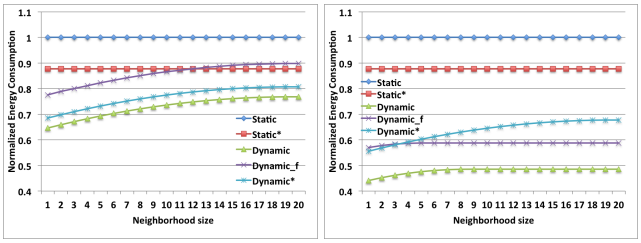
(a) Flipped-Pareto distribution (b) Uniform distribution (c) Pareto Distribution

Figure 10: Energy consumption of smart-HLPL with different probability distribution functions

Figure 10 shows the energy consumptions for each probability distribution function. We can draw several conclusions: *i)* Distribution functions did not have any major impact on the results presented in previous sections. The ordering of the algorithms is still the same for each of the tested probability distribution functions. *ii)* For all the cases analyzed in previous sections: the gap between the average energy consumption values of the algorithms got smaller for Flipped-Pareto case especially for load values greater than 0.7. Also, the *dynamic* algorithms have performed very close for these load values. *iii)* In the Pareto distribution case, *Dynamic* and *Dynamic_f* outperformed *Static** for the load value roughly 0.8 and *Dynamic** did the same for 0.85. These values are slightly higher than the results presented in previous sections. *iv)* Finally, we can say that when the nodes have higher workloads, the performance gaps between *dynamic* algorithms get smaller and for the cases where the nodes have lower workloads, the gap between *Dynamic_f* and *Dynamic** as well as the gap between *Dynamic** and *Dynamic* get larger for load values greater than 0.7.

6.5 Analysis of scalability

We have analyzed the scalability of HLPL in terms of the neighborhood size, number of nodes, and number of packets. For the neighborhood size analysis, we created a simulation with 20 nodes with the workload up to 10 packets with Uniform probability distribution. We repeated the simulation for neighborhood size from 1 to 20 for the load value of 1. Furthermore, we conducted this simulation for greedy-HLPL and smart-HLPL. The results presented in Figure 11 show that the average energy consumption of dynamic algorithms is an increasing but concave function of neighborhood size. The reason behind this is the *scheduling* of the node-level transmissions. A node that has the i^{th} slot in the superframe does not have to listen more than $i-1$ neighbors to check for its preamble (for example, $node_5$ will have to listen only to the 4 of the previously scheduled nodes even though it can overhear 19 other nodes in the cluster). Figure 11b shows that, when the nodes wait for expected amount of time before they start reverse-low-power listening, we can say that even with the largest neighborhood size, the dynamic algorithms outperform static ones. However, we cannot conclude the same for greedy-HLPL. In the latter case, *Static** outperforms *Dynamic_f* for neighborhood size greater than 13.



(a) greedy-HLPL (b) smart-HLPL

Figure 11: Scalability in terms of neighborhood size

In Figure 11a we see that the neighborhood size has similar overhead for all *dynamic* algorithms since the concavity of the graph looks almost identical for each of these algorithms. The results shown in Figure 9c indicate that *Dynamic** has lower energy consumption than *Dynamic_f* for a 10-node cluster with the neighborhood size of 4 at load 1. However, the results shown in Figure 11b indicate that there exists a neighborhood size where *Dynamic_f* starts to outperform *Dynamic**. This is because the low-power listening and waiting for expected amount of time have different effects on different dynamic algorithms due to previously used modulation levels and mismatch between the expected-number-of-packets and the expected-time (as explained in Section 6.2).

We have also analyzed the scalability in terms of number of nodes and packets. For the number of nodes case, we have conducted simulations from 1 to 20 nodes each with uniformly distributed workload of 10 packets and for the number of packets case, 10 nodes with uniformly distributed 1 to 20 packets of probabilistic workload. Furthermore, we have repeated these simulations with neighborhood size of 4 and with no neighborhood. Figure 12 shows the scalability in terms of number of nodes and packets – the simulations ran for the neighborhood size of 4 for both cases. The linear regression analysis shows that for all of the scheduling algorithms, the average energy consumption grows linearly with

respect to number of nodes and number of packets.

6.6 Discussion of the Evaluation Methodology

Our simulator was developed to assess the performance of the proposed schemes that combine DMS and low-power listening. As a proof-of-concept approach, we focused on a small-scale system connected through a star-topology (single-hop) network, which may be found in industrial control applications. We believe that the presented results give sufficient evidence on the applicability of the schemes, at least in similar settings.

Radio irregularity in wireless sensor networks is a known fact [26], which includes non-isotropic radio range and non-isotropic signal strength phenomena. [26] indicates that, in principle, radio irregularity has a more tangible impact on the routing layer than the MAC layer. Consequently while the main findings of our evaluation on our small-scale star topology should remain valid even with the consideration of the radio irregularity, a more sophisticated evaluation tool, such as a network simulator augmented with radio irregularity models (e.g., *ns3*), should be used when the framework is extended to larger-scale systems or to different (e.g., mesh or multi-hop) topologies. Radio irregularity will also have an effect on the neighborhood size, whose analysis was presented in Section 6.3. In order to extend our work to networks with single-hop but asymmetrical topologies, the distances between the sensors and the coordinator need to be known a priori in order to set-up a precise superframe length. In the absence of this a worst-case distance can be considered. Notice that the radio irregularity and varying link quality will also translate to increased energy consumption due to packet loss for all the schemes presented.

It should be noted that an extension to multi-hop topology requires addressing several additional issues; in particular, scheduling of the transmissions on neighboring nodes must be carefully planned to address collisions which may have an accumulative and adverse effect on the end-to-end delay. As an example, [6] proposes a graph-theoretical solution based on a modified Bellman-Ford algorithm to find minimum delay end-to-end TDMA schedules in multihop wireless networks. In DMS-enabled settings, there will be a need to extend those solutions to factor in the potential impact of slack reclaiming on the end-to-end delay. We leave this rather interesting problem as a future work.

6.7 Analysis of packet loss

To analyze the packet loss probability in an actual deployment, we implemented an *emulation* of DMS on Zolertia Z1 motes (that do not have DMS capability) using ContikiOS. Specifically, the transmission times that correspond to individual modulation levels are computed by scaling the transmission time that corresponds to the maximum modulation level which is 8. In general, the transmission time increases linearly with the modulation level, because the time to send a single bit is $\frac{1}{b \times R_s}$ where b is the modulation size and R_s is a constant denoting the number of symbols transmitted per second. Based on this relationship, the packet loss rate is analyzed for different modulation levels in a series of experiments.

We have used 10 Z1 motes with a coordinator mote connected to a desktop computer which formed a star topology. We have implemented the emulated version of proposed *Static* and *Static** algorithms on the conventional superframe structure as explained in Section 3. At the beginning of each superframe, the coordinator broadcasts a beacon which has the CAP start and end times. During the CAP, each node is allowed to request a GTS from the coordinator by registering their workload probability. The coordinator then computes the modulation levels according to the corresponding static algorithm. These modulation levels and GTS start and end times were embedded into the beacon of the next superframe for every node. We have emulated the Normal probability distribution of the workload as presented in Section 6.

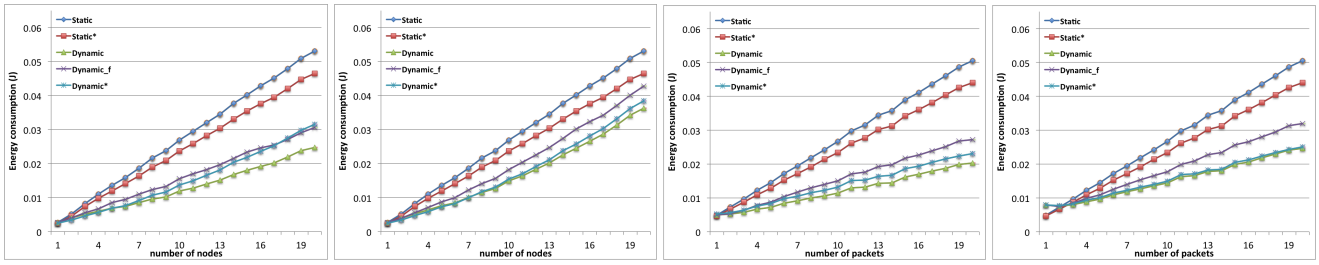
It is expected to have low packet loss during CFP due to minimized collisions. In our experiment, we have disabled the low-power listening for the coordinator mote. We aimed to analyze the packet loss as a function of transmission time which varies with the modulation level. In our experiments, retransmission during CFP is disabled and when the motes did not receive an ACK after their transmission, they stopped transmitting. We observed the probability of losing packet during CFP as 1.3%. Theoretically this ratio should be the same for any modulation level since in DMS we increase the energy to noise ratio according to the scaling function. However, we have observed a slight increase on the packet loss at reduced modulation levels, which shows that increased transmission times have a higher (and increasing) impact on the packet loss rate. The packet loss rates that are observed at different modulation levels are summarized in Table 2. In case of a packet loss, the motes add the missed packets into their future workload while making sure their workload do not exceed maximum workload to ensure the feasibility. This fact does not change the modulation levels for the static algorithms and the energy consumption increases proportionally by the corresponding packet loss rate.

Table 2: Packet Loss Rate at Different Modulation Levels

Modulation Level	2	4	6	8
Packet Loss Rate	0.017	0.016	0.014	0.013

7 Conclusions

This paper addressed the problem of ensuring real-time guarantees while minimizing the overall energy consumption in wireless sensor networks. We have studied the dynamic reallocation of slack times of a superframe, which is the mostly adopted technique for giving real-time guarantees in industrial wireless standards, by using dynamic modulation scaling. We also analyzed the effect of interference and dynamic modulation levels on low-power listening. Lastly but not the least, we have introduced a new low-power listening protocol called *hybrid-low-power listening* (HLPL) in order to overcome the interference problem caused by neighborhood. Our experiments show that dynamic slot readjustment saves important amount of energy under highly loaded systems. They also indicate that HLPL overcomes the interference caused by neighborhood and significantly reduces the overall energy consumption of the system.



(a) number of nodes smart-HLPL (b) number of nodes greedy-HLPL (c) number of packets smart-HLPL (d) number of packets greedy-HLPL

Figure 12: Scalability in terms of number of nodes and number of packets

8 Acknowledgments

This work has been supported by the US National Science Foundation award CNS-1116122. Special thanks to James Pope for sharing his expertise on ContikiOS. We would also like to thank our shepherd, Prof. Luca Mottola, for his suggestions that helped us to improve the paper.

9 References

[1] www.ti.com/lit/ds/symlink/cc1200.pdf.

[2] I. 802.15.4e Standard. https://standards.ieee.org/findstds/standard/802.15.4e-2012.html. Last accessed on 2015-May-22.

[3] M. Bandari, R. Simon, and H. Aydin. Energy management of embedded wireless systems through voltage and modulation scaling under probabilistic workloads. In *Proceedings of IEEE Green Computing Conference*, 2014.

[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 4th ACM international conference on Embedded networked sensor systems*, 2006.

[5] V. Cionca, T. Newe, and V. Dadarlat. Tdma protocol requirements for wireless sensor networks. In *Proceedings of Second IEEE International Conference on Sensor Technologies and Applications*, 2008.

[6] P. Djukic and S. Valaee. Delay aware link scheduling for multi-hop tdma wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 17(3):870–883, 2009.

[7] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. A-mac: a versatile and efficient receiver-initiated link layer for low-power wireless. *ACM Transactions on Sensor Networks (TOSN)*, 8(4):30, 2012.

[8] F. Easdy. *Hands-On ZigBee: Implementing 802.15.4 with Microcontrollers*. Newnes, 2007.

[9] B. Fateh and M. Govindarasu. Joint scheduling of tasks and messages for energy minimization in interference-aware real-time sensor networks. *Mobile Computing, IEEE Transactions on*, 14(1):86–98, 2015.

[10] L. Filipponi, A. Vitaletti, G. Landi, V. Memeo, G. Laura, and P. Pucci. Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors. In *Proceedings of Fourth IEEE International Conference on Sensor Technologies and Applications*, 2010.

[11] B. Galloway and G. Hancke. Introduction to industrial control networks. *IEEE Communications Surveys Tutorials*, 15(2):860–880, 2013.

[12] W. Liang, X. Zhang, Y. Xiao, F. Wang, P. Zeng, and H. Yu. Survey and experiments of wia-pa specification of industrial wireless network. *IEEE Wireless Communications and Mobile Computing*, 11(8):1197–1212, 2011.

[13] S. Mehta and K.-S. Kwak. H-mac: a hybrid mac protocol for wireless sensor networks. *ITC-CSCC: 2007*, pages 755–756, 2007.

[14] C. Na, Y. Yang, and A. Mishra. An optimal {GTS} scheduling algorithm for time-sensitive transactions in {IEEE} 802.15.4 networks. *Computer Networks*, 52(13):2543 – 2557, 2008.

[15] M. Palattella, N. Accettura, L. Grieco, G. Boggia, M. Dohler, and T. Engel. On optimal scheduling in duty-cycled industrial iot applications using ieee802.15.4e tsch. *IEEE Sensors Journal*, 13(10):3655–3666, 2013.

[16] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of 2nd ACM International Conference on Embedded Networked Sensor Systems*, 2004.

[17] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, 2002.

[18] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu. Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(3):511–524, 2008.

[19] M. Salajegheh, H. Soroush, and A. Kalis. Hymac: Hybrid tdma/fdma medium access control protocol for wireless sensor networks. In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pages 1–5. IEEE, 2007.

[20] C. Schurgers, V. Raghunathan, and M. Srivastava. Modulation scaling for real-time energy aware packet scheduling. In *Proceedings of IEEE Global Telecommunications Conference*, volume 6, 2001.

[21] C. Schurgers, V. Raghunathan, and M. B. Srivastava. Power management for energy-aware communication systems. *ACM Transactions on Embedded Computing Systems*, 2(3):431–447, 2003.

[22] L. Sitanayah, C. Sreenan, and K. Brown. Er-mac: A hybrid mac protocol for emergency response wireless sensor networks. In *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*, pages 244–249, July 2010.

[23] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, and M. Nixon. Wirelesshart: Applying wireless technology in real-time industrial process control. In *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium*, 2008.

[24] R. Szabo and et. al. Framework for smart city applications based on participatory sensing. In *Proceedings of 4th IEEE International Conference on Cognitive Infocommunications*, 2013.

[25] Y. Yu, B. Krishnamachari, and V. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. In *Proceedings of Twenty-third IEEE Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, 2004.

[26] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138. ACM, 2004.

[27] G. Zhou, T. He, J. Stankovic, T. Abdelzaher, et al. Rid: radio interference detection in wireless sensor networks. In *Proceedings of 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 891–901, Mar 2005.

[28] S. Zhuo, Y.-Q. Song, Z. Wang, and Z. Wang. Queue-mac: A queue-length aware hybrid csma/tdma mac protocol for providing dynamic adaptation to traffic and duty-cycle variation in wireless sensor networks. In *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, pages 105–114, May 2012.