Duty-Cycle-Aware Real-Time Scheduling of Wireless Links in Low Power WANs

Md Tamzeed Islam
Department of Computer Science
UNC at Chapel Hill
tamzeed@cs.unc.edu

Bashima Islam
Department of Computer Science
UNC at Chapel Hill
bashima@cs.unc.edu

Shahriar Nirjon
Department of Computer Science
UNC at Chapel Hill
nirjon@cs.unc.edu

***Abstract*—Low Power Wide Area Networks (LPWANs) are an excellent fit to city-scale IoT applications—offering a long range, long battery life of several years, and a data rate of 25-50kbps, which is sufficient to carry IoT traffic. However, a practical limitation in realizing a LPWAN-based real-time wireless network is the duty-cycle limit that is imposed on the sub-1GHz band by the FCC. In this paper, we overcome this challenge by proposing the first duty-cycle-aware wireless link scheduling algorithm for real-time LPWANs that considers the urgency of the packets as well as the availability of the channels. The proposed algorithm is implemented in a five-node city-wide test-bed in multiple real-world scenarios. Simulation results are provided to quantify its performance under different settings (e.g. larger networks, variety of workloads, and multiple baselines). In both real-world deployments and simulations, the proposed scheduling algorithm has outperformed all the baselines in terms of link schedulability, deadline misses, and buffer size.**

I. Introduction

The concepts of smart cities and smart communities have started to become a reality in this age of the Internet of Things (IoT). In the midst of this IoT revolution, recently, low power wide area networking (LPWAN) technologies [1]–[4] have become very popular, as they are an excellent fit to the IoT data traffic that are generated and consumed by many smart cities applications. For instance, if we think of city-scale IoT applications like smart metering, environment monitoring, road traffic monitoring, facility management, smart parking, street lighting, vehicle tracking, waste management, precision agriculture, and home automation, we observe that the basic communication requirements in these applications include a *long radio range* (i.e. several hundred meters of range), *low power* (i.e. an extended battery-life of several months or years), and *low bandwidth* (i.e. a data rate of few kbps). Thus, low power WANs are being considered as the enablers of city-scale IoT.

Among different choices of low power WANs, we study one of the most popular technologies of today, which is called the LoRa WAN [2]. LoRa has so far been mainly adopted by the European countries, although recently, over 100 cities in the USA have begun to deploy city-wide LoRa networks [5]. LoRa has an advertised radio range of up to 9 miles (in line-of-sight), a data rate of up to 50kbps, and a battery life of around 10 years. While these properties make LoRa perfect for IoT applications, unfortunately, there is a regulatory constraint on its *duty-cycle*, which does not allow a device to send data packets at will. A device in a LoRa network must wait a certain period after each successful transmission. For example, in EU, LoRa has a duty-cycle limit of 1%, i.e. once a device has used a particular communication channel for 10ms, it has to wait for another 990ms for that channel to be available to it again. The device, however, can send the packet over other available channels, and other devices can send their packets over that channel, as long as the duty-cycle constraint on any channel, for any device is not violated. In other words, duty-cycle constraint applies to each $(device, channel)$ pair.

Although we study a specific network protocol in this paper, the duty-cycle constraint in LPWANs is not a protocol specific one, rather it is band specific. From the fundamentals of wireless communication, the higher the frequency band is, the shorter is its the communication range. Hence, for a long range wireless communication, bands below 1 GHz are used. For example, LoRa uses the 902–928MHz band in the USA. Because of the long radio range, a large number of devices (in a large geographic area) compete for the same set of frequencies, and their transmissions are more susceptible to collision. Hence, duty-cycle limits are imposed by the authority to ensure fair access to the air for all devices. We consider this limit as a challenge in designing real-time IoT systems where a large number of connected devices have to send data wirelessly to a central gateway over a long distance in real-time, i.e. within an application-specific deadline.

The generic problem of scheduling wireless transmissions dates back to decades [6]–[8]. Theoretical analysis as well as results from practical deployments have been published on various categories of real-time wireless networks such as ad-hoc and sensor networks [9], [10] and WiFi [11]. The problem we study in this paper has similarity to several of these works that consider single-hop network topology [12], time division multiple access (TDMA)-based link scheduling approaches [13], [14], use of laxity to schedule packets [15], and channel selection [16]. However, ours is the first work that brings an additional pragmatic issue in real-time wireless link scheduling algorithms, which is the *duty-cycle-awareness*. Note that, although the term 'duty-cycle' is commonly used in the wireless sensor network community to refer to the sleep-vs-awake ratio of a node, the duty-cycle in the LPWAN context is tied to both a node and a specific channel. Therefore, we are required to design a solution to a new class of link scheduling

problems where both the packet and the channel need to be scheduled judiciously.

When compared to classical real-time scheduling problems, the problem at hand is analogous to scheduling tasks in a multiprocessor system, where a specific processor becomes unavailable to a specific task (but not necessarily to other tasks) for a specific duration after an instance of it has been executed. We propose a simple yet effective scheduling strategy for this scheduling problem by introducing a new metric that dynamically scores each processor with respect to a given task and the task's remaining waiting time for that processor due to the duty-cycle limit. We name this new metric: *'gravity'*. At each scheduling step, a task (a wireless link) having the least laxity [17] is scheduled on a processor (channel), which is determined by a duty-cycle-aware *maximum gravity* processor (channel) selection algorithm.

In order to demonstrate the performance of our proposed scheduling algorithm, we implement a complete system—consisting of five low-power LoRa nodes, a LoRa gateway, and a server in the cloud. We deploy this network in the city of Chapel Hill, NC. Our system is up and running since September 1, 2017, and the packets sent from the nodes can be viewed at [18]. Each node periodically generates a packet and follows an offline-generated transmission schedule (according to our duty-cycle-aware scheduling algorithm) to send the packets to the gateway. The gateway forwards the packets to the server over the Internet. We also have developed a Java-based simulation software that generates the schedule for a given a workload description. We use this software to simulate the workload to analyze its schedulability as well as to generate the schedule when the workload is schedulable.

We evaluate the real-time performance of the network in an outdoor and an indoor setup. Although LPWANs are meant for outdoors, we wanted to see its performance in indoor scenarios as well, so that we can compare the two. Besides the real deployments, we also have conducted multiple simulations to quantify the real-time performance of the proposed algorithm for large scale networks and for different real-time workloads.

The contribution of this paper are the following:

- We demonstrate the effect of duty-cycle on the real-time performance of LPWANs, illustrate the need for scoring communication channels, and propose a new metric called the *'gravity'* to score channels dynamically.
- We propose the first wireless transmission link scheduling algorithm that explicitly handles the duty-cycle constraints in LPWANs. The time-complexity of this offline scheduling algorithm is $O(J^2 \log J)$.
- We develop a complete system consisting of a five-node LoRa network and deploy the network in two real world scenarios. We also conduct simulation-based experiments under different settings (e.g. larger networks, variety of workloads, and multiple baselines). In both real-world deployments and simulations, the proposed scheduling algorithm has outperformed all the baselines in terms of link schedulability, deadline misses, and buffer size.
- We have open-sourced the software for the LoRa nodes and the simulator. They are accessible from here [18].

II. Background

A. Overview of LoRaWAN

LoRa [2] stands for *'Long Range'*. It defines the physical layer of an emerging network technology that offers low data rate wireless communication over long distances, while consuming very little power. For example, LoRa radios have a battery lifetime of around 10 years, a communication range of up to 9 miles (line-of-sight), and a data rate of 27kbps–50kbps. Because of these properties, LoRa has gained a lot of attention in the Internet of Things (IoT) applications where battery operated devices require access to the Internet but are physically located miles apart from an Internet gateway.

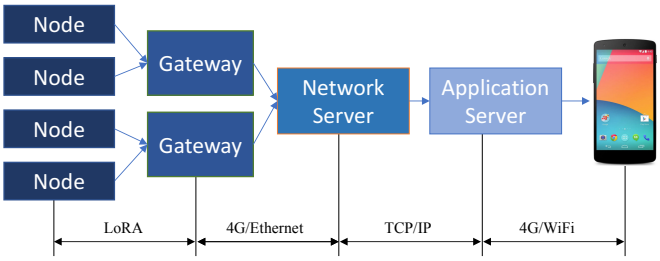


Fig. 1. LoRaWAN Network Architecture.

LoRaWAN is a specification for Low Power Wide Area Network (LPWAN) that defines the system architecture and network protocols for LoRa capable devices. LoRaWAN networks are organized as a star of stars topology as shown in Figure 1. Four types of entities are present in a LoRaWAN. The sensor nodes or end nodes send data packets to a LoRa capable gateway. A single LoRa gateway is able to cover an entire city (hundreds of square kilometers). Gateways are connected to a network server over a backhaul network such as 4G or Ethernet. Network servers are connected to an application server via TCP/IP. Users can access the data from application servers on any device with an Internet access such as smartphones or personal computers.

Although LoRaWAN has a long range and a long battery life, the low data rate limits its usage to applications which do not generate large amount of data traffic. IoT applications where LoRa has shown promising results include smart metering, facility management, smart parking, street lighting, vehicle tracking, home automation, waste management, and remote health-care.

B. LoRa Physical Layer Properties

LoRa physical layer handles the lower level details of wireless communication. LoRa operates in 433, 868 or 915MHz ISM bands. Key properties of this layer are as follows:

- *Chirp Spread Spectrum (CSS) Modulation:* The LoRa physical layer uses a special type of spread spectrum

modulation technique where information bits are encoded as frequency *chirps* (frequency varying sinusoidal pulses) [19]. The use of chirps improves its robustness against interference, Doppler effect, and multipaths [20]. Each symbol is encoded with 2^{SF} chirps, where SF is called the *spreading factor* and takes a value between 7 to 12. There is a trade-off between the spreading factor and the communication range. A higher value of the spreading factor results in a longer time for each symbol transmission and yields a longer communication range. The way chirps are designed for different spreading factors, they are orthogonal to each other at different values of $SF \in [7, 12]$, and thus multiple data packets can be sent in parallel as long as their spreading factors are different.

- *Time-On-Air:* The *Time-on-Air* of a packet, T_a is the duration for transmitting a LoRa packet. It is expressed as a function of the number of symbols per packet n_s, chirp time T_c, and spreading factor SF as follows:

$$T_a = n_s \times 2^{SF} \times T_c \tag{1}$$

  Since the communication bandwidth and time-resolution are inversely related ($BW \approx 1/T_c$), we can use their relationship to express the above equation as:

$$T_a = n_s \times \frac{2^{SF}}{BW} \tag{2}$$

- *Duty-Cycle Limit:* The duty-cycle is defined as the fraction of time an end-device keeps the channel occupied for communication. To reduce collisions as well as to increase the fairness of channel use by different transmitters, there is a limit on the maximum duty-cycle for an end-device. For example, European FCC allows a maximum duty-cycle of 1% for EU 868 end-devices [21]. Therefore, if an end-device uses a channel to transmit a frame, the limit on duty-cycle restricts it to transmit on the same channel again until after a period of silence. The device, however, can use other available channels (as long as the duty-cycle limits on those channels are maintained, of course). Formally, given the duty-cycle limit δ, an end-device must not transmit anything on the most recently used channel for a minimum off-period, T_{off}

$$T_{off} = T_a \times \left(\frac{1}{\delta} - 1\right) \tag{3}$$

  Note that, if there are 8 channels and the duty-cycle is limited to 1%, then the duty-cycle per channel is 1/8%. For example, if an end-device transmits on a channel for 1 second, the channel will be unavailable for it for the next 799 seconds.

C. LoRa MAC Layer Properties

LoRa MAC layer determines how multiple end-devices access the wireless media to communicate with the gateways. Key properties of LoRa MAC layer are as follows:

- *Sub-bands and Channels:* LoRa operates on a specific range of frequencies (an ISM band). Each band is divided into multiple sub-bands, and each sub-band is further divided into a number of channels. For example, in the USA, LoRa operates on the 915MHz ISM band that contains the frequencies between 902–928MHz. This band is divided into eight sub-bands, and each sub-band contains 10 channels (eight 125KHz downlink channels, one 500 KHz downlink channel, and one 500KHz uplink channel).
- *Interference:* Each gateway in a LoRa network listens on a particular sub-band. When two end-device communicates with the same gateway, at the same time, at the same channel, and using the same spreading factor, they will cause interference and their packets will collide.
- *Device Classes:* LoRaWAN defines three classes of devices: class A, class B, and class C, in order to meet the demands of different types of applications. Class A devices use ALOHA [22] protocol for an uplink packet transmission, followed by two short downlink receive windows. This class is defined for battery operated devices. It does not require carrier sensing and thus helps keep the energy consumption of an end-device to the minimum. Class B is designed for devices which may require additional downlink communication. Class C devices always listen for ongoing transmissions before transmitting anything. In this paper, we consider only the class A devices which are low power and suitable for IoT applications.
- *Pure and Slotted ALOHA:* ALOHA is a MAC layer protocol that allows a node to send data whenever it is ready. Because there is no coordination among different transmitting nodes, ALOHA yields a high rate of collisions. As the number of devices on the network increases, the number of collisions increases.
  Slotted ALOHA introduces the concept of time-slots and allows a node to send a packet only at the beginning of a time-slot. It eliminates partial collisions (i.e. collisions in the middle of a packet transmission) but the medium access is still not controlled. Collision occurs whenever more than one end device become ready with a packet to transmit. Due to the lack of coordination or a packet transmission schedule, the real-time performance of both pure and slotted ALOHA is extremely poor.

III. Problem Formulation

In LoRaWAN, a set of end-devices or nodes talk to a specific gateway in a single-hop network by forming a star topology. Similarly, multiple gateways form another star topology centering a network server. In this paper, our focus is on the real-time communication issues in a LoRa network, i.e. the network formed by the end-nodes and the gateways. Ensuring the end-to-end real-time guarantee between an end-device and a data consumer like the smartphone in Figure 1 is a completely different problem as it involves multiple types of intermediate

networks and devices, and hence, is out of the scope of this paper.

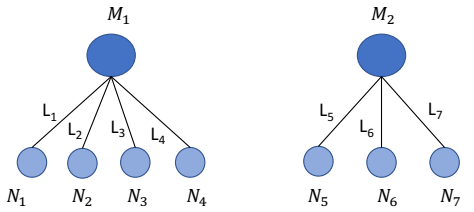


Fig. 2. Nodes and gateways in a LoRa network form a bipartite graph where links $\{L_i\}$ exist only between a gateway $\{M_i\}$ and a node $\{N_i\}$.

Communication networks are typically modeled using graphs where nodes represent communicating entities and edges represent communication links. In a LoRa network, we have two types of communication entities— the end-devices and the gateways. Information flows only between an end-device and a gateway. Since no two gateways or no two end-devices communicate between themselves, we model a LoRa network as a bipartite graph, $G = (M, N, E)$, where $M = \{M_1, M_2, \ldots, M_n\}$ represents the set of n gateways, $N = \{N_1, N_2, \ldots, N_k\}$ represents the set of k nodes, and the set of edges $E = \{e_{ij}\}$ denotes all communication links between M and N. An edge $e_{ij} = (N_i, M_j)$ exists only if there is a reliable communication link between a node N_i and a gateway M_j.

We consider k communication links, $L = \{L_1, L_2, \ldots, L_k\}$ between N and M. Because LoRa is a single-hop star network, the links L are similar to the edges E of the network graph in this context, but with the difference that links have additional properties. For each link $L_i = (N_i, M_j, T_i, A_i, D_i)$, a packet is generated at the node N_i periodically at every T_i unit of time, and is destined to reach the gateway M_j on or before the deadline D_i. The time-on-air for a packet transmission for L_i is A_i. We consider the spreading factor to be constant for all links. We denote the kth packet generated at link L_i by τ_{ik}. The generation of each packet at a node creates the need for a link to be scheduled for transmission. Hence, scheduling a *link* is similar to scheduling a *task* in real-time systems, and like *jobs* are defined as invocations of tasks, *transmission of a packet* can be thought of as activation of a *link*.

We denote the set of m channels as $C = \{C_1, C_2, \ldots, C_m\}$. In an ideal world where there is an infinite number of available communication channels and there is no limit on the duty-cycle, each link would require exactly 1 time-slot. So, in practice, since we are limited to a fixed number of channels, two links cannot be scheduled on the same channel at the same time slot (unless they are so far apart that they are out of each otherś interference range). Furthermore, because of the duty-cycle constraint, a node cannot transmit packets even if the channel is free. Hence, the end-to-end latency of a packet depends on the duration a node has to wait in order to meet these constraints before it can transmit a packet. We express the end-to-end latency of a link by $d = (f - r + 1)$, where f and r denote the time slots in which a packet is generated and gets scheduled, respectively.

When a link L_i uses a channel C_j for its kth packet transmission τ_{ik}, and the time-on-air for this packet is A_i, the link can not use C_j for the next $t_{off}(L_i, C_j, A_i, \sigma)$ slots. The value of $t_{off}(L_i, C_j, A_i, \sigma)$ is calculated using the Equation 3.

Given a set of links $L = \{(N_i, M_j, T_i, A_i, D_i)\}$, duty-cycle limit δ, and the number of channels m, our objective is to schedule the links such that, $d_i \leq D_i$, and $\frac{A_i}{A_i + t_{off}(L_i, C_j, A_i, \sigma)} \leq \frac{\sigma}{100}, \forall L_i \in L, C_j \in C$.

IV. Motivation

Traditional real-time scheduling algorithms have been used in scheduling data transmissions in both wired and wireless networks [14], [15], [23]. However, these algorithms do not take a duty-cycle constraint into their consideration. The duty-cycle constraint in LPWANs makes the problem of scheduling packet transmissions in a wireless network unique. Duty-cycle forces an end-node to migrate from one channel to another after using the channel for a fixed amount of time that is regulated by the FCC [24]. In a multiprocessor scheduling scenario, this is analogous to a scheduling problem where a processor becomes unavailable to a task for a certain period, after the processor has been used by the task recently.

TABLE I
Example: two links and their parameters.

Link	Release Time R_i	Time-On-Air A_i	Deadline D_i	Period T_i
L_1	0	2	3	5
L_2	0	4	5	5

For example, lets consider a network with two end-devices N_1 and N_2, and one gateway G_1. Two links L_1 and L_2 are generating packets periodically at N_1 and N_2. Table I lists their release times (R_i), time-on-air (A_i), deadlines (D_i), and periods (T_i). We assume that there are two channels C_1 and C_2 to which links can be scheduled for packet transmission. Moreover, we impose a duty-cycle limit of 40%, so that a channel becomes unavailable for L_1 and L_2 for 3 and 6 time-slots, respectively, after it has been used by a link.

Now, let us simulate the scheduling steps for an arbitrary scheduling algorithm.

- At time-slot 0, both L_1 and L_2 generate their first packet. Both channels C_1 and C_2 are available to the links. The packet transmission of L_1, τ_{11} uses channel C_1 and packet transmission of L_2, τ_{21} uses channel C_2. In Figure 3 we show that due to the duty-cycle limit, L_1 and L_2 can not use C_1 and C_2 until the $2 + 3 = 5$th and the $4 + 6 = 10$th time slot, respectively. Therefore, $t_{off}(L_1, C_1, A_1, \sigma) = 3$ and $t_{off}(L_2, C_2, A_2, \sigma) = 6$.
- At time-slot 5, both L_1 and L_2 generate their second packet τ_{12} and τ_{22}. Both packets have the same laxity of 1. Suppose, τ_{12} chooses channel C_1, which is currently available to it. Because traditional scheduling algorithms

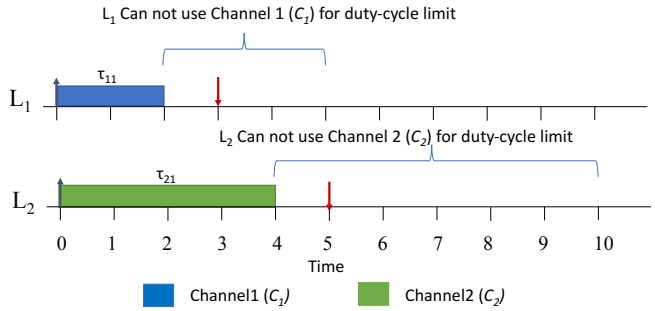


Fig. 3. Link 1 ($L1$) and Link 2 ($L2$) releases their first transmissions at time slot 0. Due to duty-cycle limit L_1 and L_2 can not use C_1 and C_2 until the $2+3=5$th and the $4+6=10$th time slot, respectively.

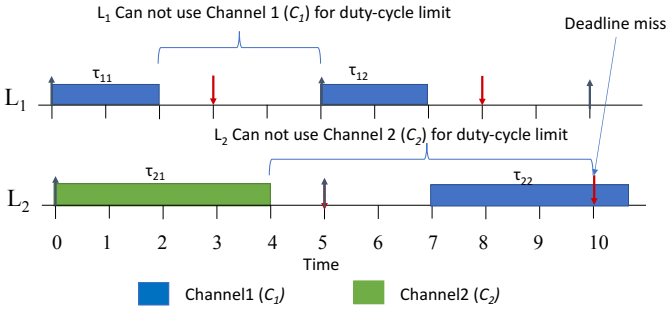


Fig. 4. The second transmission τ_{12} of $L1$ uses C_1. At time slot 5, the second transmission τ_{22} of L_2 can not use either of C_1 or C_2. Eventually, τ_{22} misses deadline.

do not impose any restriction on channel/processor selection, this choice is arbitrary (we discuss the other selection option in the next bullet point). However, τ_{22} can not use channel C_2 for its transmission at this moment, as C_2 is unavailable to L_2 until time-slot 10. At time slot 7, C_1 becomes available to $L2$, and τ_{22} can use it for transmission. However, from Figure 4, we see that τ_{22} still misses the deadline.

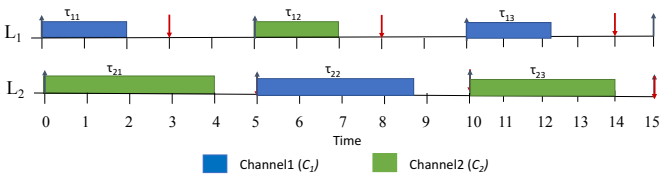


Fig. 5. The second transmission τ_{12} of $L1$ uses C_2. So, At time slot 5, the second transmission τ_{22} of L_2 is able to use C_1 and does not miss its deadline.

- At time-slot 5, we have another option, which is shown in Figure 5. Suppose, τ_{12} chooses channel C_2 this time, as it is also available to it at time-slot 5. Given this, τ_{22} can use channel C_1 for its transmission, which is available and does not restrict τ_{22} at that moment due to duty cycle limit. Therefore, by using C_1, τ_{22} makes the deadline.

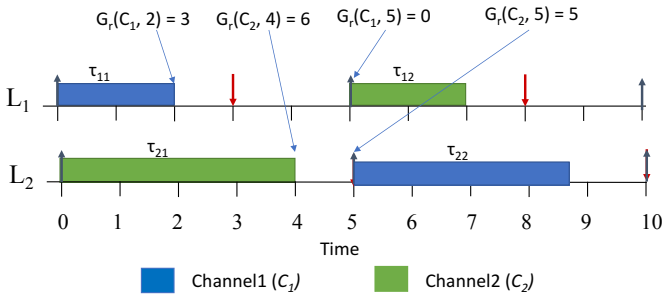


Fig. 6. C_1's gravity is updated at time slot 2, $G_r(C_1, 2) = 3$ after being used by L_1 for its transmission. Likewise, C_2's gravity is set after being used by L_2, $G_r(C_2, 4) = 6$.

It is evident from the above example that L_2 would not have missed deadline if L_1 used C_1 for its second transmission τ_{12}. Therefore, unlike traditional scheduling algorithm, we need to have some mechanism to choose a right channel from the available ones. We need to force L_1 select C_2 at time slot 5, to make the links schedulable. To enable this, we propose a scoring-based channel selection algorithm.

The goal of the algorithm is to let a link select its channel in a way that the selection helps other links to avoid the channels that are unavailable to them. In other words, when a link has multiple channels to choose from, it should choose the one that hurts the other links the least. To implement this, we score each channel based on the number of time-slots they are unavailable due to the last successful transmission of a link on it. We call this the *'gravity'* of a channel.

V. Scheduling Algorithm

A. Defining Channel Gravity

The gravity is a dynamic property of a channel. Gravity is defined by the maximum unavailability of a channel over all links. At each time-slot, a node gets to use the channel that has the highest gravity among all the available channels at that moment. The intuition behind this scoring is that the channel that is unavailable to other links for the longest period should be selected to the next packet transmission so that other links can use the remaining channels when needed. The value of gravity is updated at each time-slot. After a channel C_i has been used by a link L_j, the gravity of that channel $G_r(C_i)$ is updated using the following equation:

$$\mathrm{G_r(C_i, 0)} = 0 \tag{4}$$
$$\mathrm{G_r(C_i, t)} = \max_j \Big\{ G_r(C_i, t-1), t_{off}(L_j, C_i, A_j, \sigma) \Big\}$$

B. An Example

We revisit the example from the previous section but this time we also demonstrate the role of gravity. At time-slot 0, the gravity of each channel is set to zero. At time-slot 2, link L_1 finishes its first transmission τ_{11} over channel C_1. As mentioned earlier, $t_{off}(L_1, C_1, A_1, \sigma) = 3$. Therefore, at time-slot 2, the gravity of C_1 is updated to $G_r(C_1, 2) = \max\{0, 3\} = 3$.

Likewise, at time slot 4, after the end of transmission τ_{21}, the gravity of C_2 is updated to $G_r(C_2, 4) = \max\{0, 6\} = 6$.

After each time-slot, the gravity of all channels is decremented. At time slot 5, $G_r(C_1, 5) = 0$ and $G_r(C_2, 5) = 5$. Since our proposed scheduling algorithm picks the channel with the maximum gravity, L_1 will choose C_2 for its second transmission τ_{12}, as opposed to C_1. This enables L_2 to choose C_1 for its second transmission τ_{21}, which is our desired schedule.

C. Algorithm Design

The proposed duty-cycle-aware link scheduling algorithm works in two steps. Since two transmission cannot use the same channel at the same time-slot, at each time-slot, the following two steps are applied multiple times, until there is no channel that can be used in that time-slot.

- *Packet Selection:* In the first step, we select a transmission packet based on its laxity. Among all the transmissions that are ready to go, the packet with the least laxity [17] is selected for transmission. In case of a tie, the packet having the earliest deadline is selected. For further ties, we choose the transmission arbitrarily [15].
- *Channel Selection:* In the second step, we select a channel for transmitting the selected packet based on its gravity. Among all available channels for the selected link, the channel with the highest gravity is selected for transmission. After using the channel, the gravity is updated according to Equation 4.

D. Pseudocode

Algorithm 1 shows the pseudocode of the scheduling algorithm. We call this D-LLF. It takes a set of packets to transmit $\{\tau_{ik}\}$, total number of channels m, and the duty-cycle σ as inputs, and outputs a 2D scheduling table S specifying which packet is scheduled in which channel at each time-slot.

In lines 1 to 3, we initialize time-slot to 1, set all unscheduled transmissions at time-slot 1 and set all of channels' gravity to 0. In line 5, we get all the released transmissions at time-slot s. Line 6 decreases the gravity of all channels if it is greater than 0 at that time slot.

In line 7, the function $sort_{gravity}(ch)$ sorts the channels in the descending order of their gravity. In line 8, $sort_{laxity}(Released(s))$ sorts the released transmissions in an ascending order of laxity of the transmissions.

In line 10 to 12, if any of the transmissions misses the deadline, we declare the workload as unschedulable. We select the channel for a transmission in line 13 to 20. While selecting a channel for a ready transmission, the function $channelAvailable(\tau_{ik}, c)$ in line 14, returns a boolean to indicate the availability of channel c to transmission τ_{ik}. This function checks two things: a) if channel c is being used by any other transmission, and b) if τ_{ik} is restricted from using channel c due to the duty-cycle limit. If both of these cases are false, the function returns $true$, otherwise, it returns $false$. We update the gravity of a channel in line 17. Note that the gravity is updated at the end of a transmission. Therefore, instead of updating the gravity at time-slot s, we update it at time-slot $s + A_i$. Finally, in line 22, we move to the next time-slot.

Algorithm 1: D-LLF Scheduling Algorithm

```
   Input  : {τ_ik} ← All packets to transmit.
   m ← Total number of channels.
   σ ← Duty-cycle limit.
   Output: s[1...T][0...m − 1] // schedule
1  s ← 1 // initialize time slot
2  τ_s ← {τ_ik} // unscheduled transmission
3  Set All Channel gravity to 0
4  while τ_s ≠ φ do
5      Released(s) ← set of released transmissions at slot s
6      Reduce all channel gravity by 1 if greater than 0
7      sort_gravity(ch)
8      sort_laxity(Released(s))
9      for each τ_ikϵReleased(s) do
10         if τ_ik misses deadline then
11             return unschedulable
12         end
13         for c ϵ ch do
14             if channelAvailable(τ_ik, c) = true then
15                 S[s][c] ← τ_ik
16                 τ_s = τ_s − τ_ik
17                 G_r(c, s + A_i) = Max(G_r(c, s − 1 +
                     A_i), t_off(L_i, c, A_i, σ))
18                 break
19             end
20         end
21     end
22     s ← s + 1
23 end
```

E. Complexity Analysis

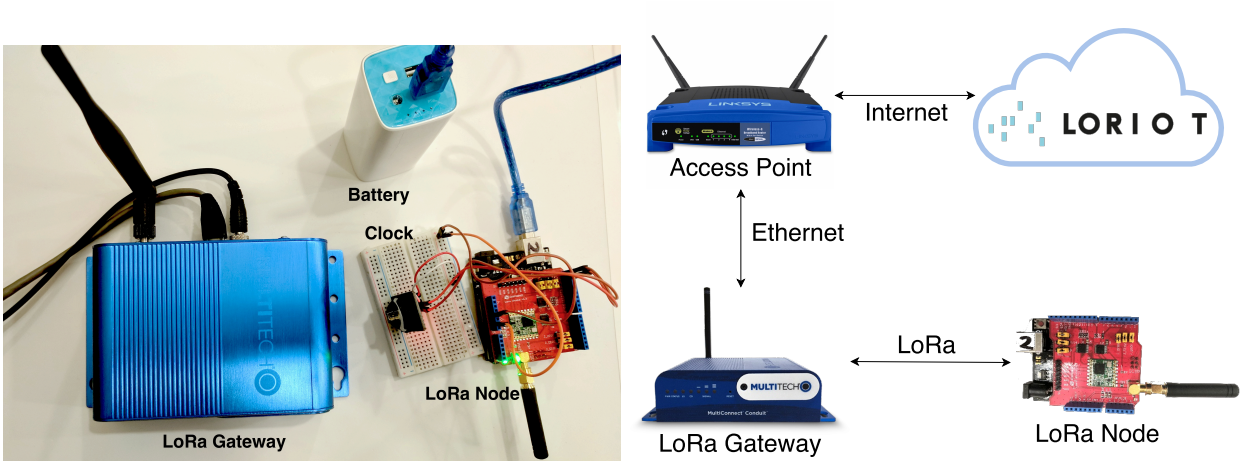
An upper bound of the released transmissions at any time slot is $O(J)$, where J is the total number of packet transmissions to schedule. The sorting of channels and released transmissions take $O(C \log C)$ and $O(J \log J)$, respectively. Here, C is the number of channels. Finding available channels for each released transmission is $O(JC)$. Hence, the total time complexity of our algorithm is $O(J(C \log C + J \log J) + JC)$. Since the number of channels is constant for a given network, the overall time complexity of the algorithm is $O(J^2 \log J)$.

VI. System Development

This section provides some highlights from our implementation of the LoRa network that we feel would be helpful to anyone who wants to replicate the complete system. Figure 7(a) shows a photo of the main elements of our LoRa network.

A. Developing the LoRa Nodes

We develop LoRa nodes in our lab by interfacing a LoRa radio shield [25] with an Arduino Uno [26] that hosts an ATmega328P microcontroller. The radio shield internally uses



(a) (b)

Fig. 7. (a) A LoRa Node is connected with a clock and battery. A Gateway is placed beside the node. (b) A LoRa node communicates with gateway using LoRa protocol. The gateway relays the node's message to access point via Ethernet. The access point connects with server using internet.

a transceiver SX1272/73 [27] which is controlled from the Arduino using a modified software library from IBM [28]. Each node is powered by a 10,000mAh USB power bank. The internal 16MHz quartz crystal of Arduino Uno is unreliable for time synchronization as the clock drifts over time. Hence, to time synchronize all the nodes in our network, we interface an external real-time clock [29] with the Arduino board. These real-time clocks are powered by their own battery and their drift over 24 hours after synchronization is too small to be noticeable ($<$ 1ms). Both the modified library and our customized application are written in C. Our source code is open and accessible online from here [18].

B. Configuring the Gateway

We use a Multitech Conduit device [30] as the gateway. This is a configurable and scalable Internet gateway for industrial IoT applications where LoRa is used for the local wireless network. The gateway is equipped with an ARM9 processor having a 32-bit ARM and 16 bit thumb instruction set, 16K data cache, 256 MB flash memory and 128$\times$16MB DDR RAM. This runs on an enhanced closed source embedded Linux platform. We use the gateway as a LoRa packet forwarder. The gateway listens to one sub-band at a time, and therefore, a gateway can listen to eight channels simultaneously.

To configuring the Gateway, first we connect it to a computer via the Ethernet port. We set the gateway as a DHCP network via WAN. Finally, we connect it to a WiFi access point via Ethernet. In order to program it, we connect a computer to the same access point and remotely log in to the gateway via secure shell ssh. To enable the packet forwarder, we run a script which also logs the packet information on the device. The setup is shown in Figure 7(b).

C. Configuring the Server

For the application server, we use a free and open server called the LORIOT [31]. LORIOT is a cloud based LoRaWAN network server. This server platform contains both the network and the application server which are required to setup a LoRaWAN. The platform provides APIs for IoT applications to access the data streams from the end nodes. We use a community network account which has a limit of 1 gateway and 10 nodes. Because of the free community account, we faced some limitations that made the application at the server end unreliable in terms of real-time display of packets, although the gateways were receiving them in real-time. For this reason, we rely upon the packet information logged in the gateway.

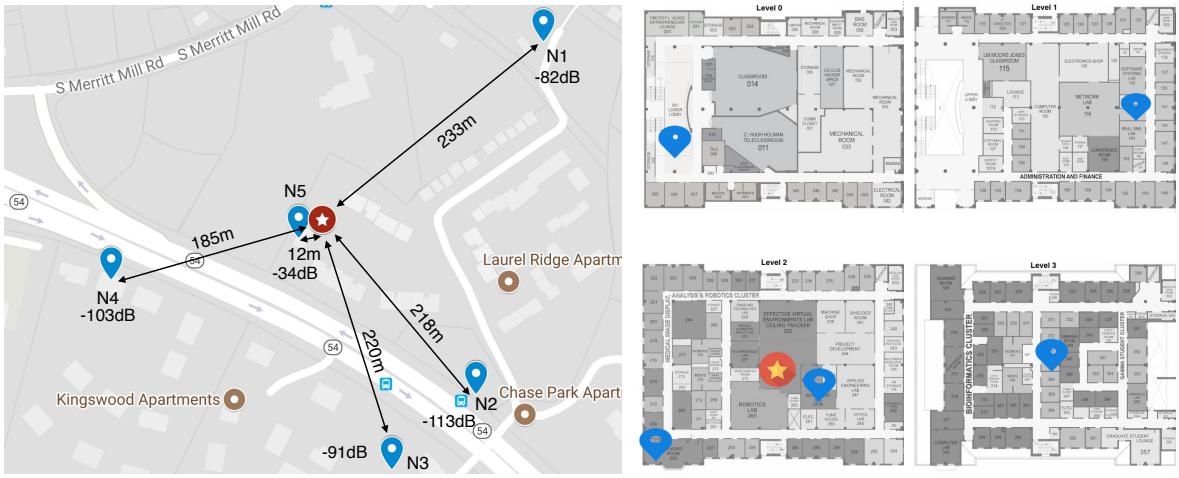
VII. Real-World Deployment

We setup a LoRa network consisting of five LoRa nodes and a gateway, and conduct experiments in two real-world scenarios–an outdoor and an indoor scenario.

A. Testbeds and Workload

We setup a five-node outdoor LoRa network in the city of Chapel Hill. Two residential areas, separated by a highway, were chosen to place the nodes. The map of the test environment is shown in Figure 8(a). We positioned the gateway in the balcony on the first floor of a two storied building. We had to place the gateway inside the building as it was powered from an electric outlet. The nodes were placed around the gateway within a radius of 220m and were powered by USB power banks. Prior to choosing the exact locations of the nodes, we did a day long survey to measure signal strengths and the reliability of the communication links at various locations in the test area. Finally, we selected the locations where we observed the least packet drops and that were at a reasonably long distance from the gateway. For the LoRa network, a moderate spreading factor of 9 and a code rate of 4/5 were chosen to have a bandwidth of approximately 125KHz.

For indoor test-bed, we place the nodes and the gateway inside the Computer Science building at the UNC. The gateway was placed on the second floor of the building. Two of



(a) Outdoor Scenario (b) Indoor Scenario

Fig. 8. Placement of the nodes and the gateway in the real-world experiment.

the nodes N_1 and N_2 were placed on the same floor, but in different rooms. N_3, N_4, and N_5 were placed on the ground, the first, and the third floor, respectively. Figure 8(b) shows the positions of the nodes and the gateway on the floor plan.

We send one-byte payloads from four of the nodes and five-byte payloads from one node. Given the A_i and t_{off} for one-byte payloads, we set the period of each node to $(A_i + t_{off})$. To stress-test our algorithm, we set the deadlines of all the nodes to their time-on-air. We run the whole experiment for both least laxity first (LLF) and our duty-cycle-aware algorithm (D-LLF) for a duration of twenty hyper-period.

B. Experimental Results

In order to compare the real-time performance of our proposed approach (D-LLF) with the baseline least laxity first (LLF), we count the number of packets that missed the deadline. In Figures 9 and 10, we report the percentage of packets dropped as well as the percentage of packets that actually missed the deadline for both algorithms, for outdoor and indoor scenarios, respectively.

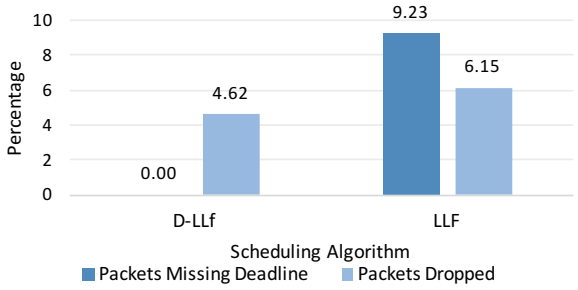


Fig. 9. The proposed D-LLF finds a feasible schedule for the outdoor scenario. Thus, no packet misses the deadline. The baseline LLF does not find a feasible schedule, and as a result, 9.23% packets miss their deadlines. We observe typical packet drops in both cases.

In the outdoor scenario, proposed D-LLF outperforms LLF. When links are scheduled using D-LLF, no packet misses the deadline, whereas, for the regular LLF, the percentage of packets missing deadline is 9.23%. We observe about $4.62 - 6.15\%\%$ packets were dropped, which is typical in a LPWAN.

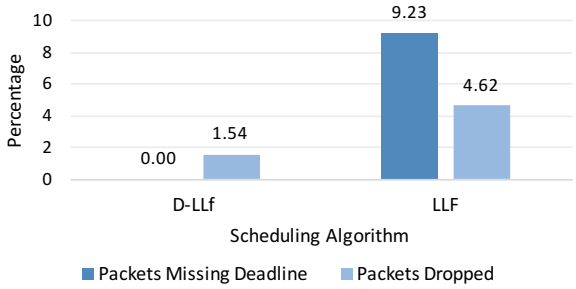


Fig. 10. The D-LLF finds a feasible schedule for indoor scenario as well. With LLF, 9.23% packets miss their deadlines. We observe less packet drops indoors than outdoors.

In the indoor scenario, no packet missed the deadline in case of the D-LLF, but in case of the LLF, 9.23% of the total packets missed the deadline. We observe about $1.54\% - 4.62\%$ deadline misses due to packet drops, which is less than what we observed in the outdoor scenario.

C. Lessons Learned

During the deployment, we observe that using the highest spreading factor (12) increases packet drops. It increases the time-on-air and thus not ideal for scenarios where a gateway listens for multiple nodes at the same time. Hence, we use moderate spreading factor of 9 to achieve reliable communication at the cost of slightly reduced radio range.

Ideally, a gateway should be placed at a high location, e.g. on a tower to get the best range. But due to power supply and Internet connectivity issues we place it in a building. To

test the different, we place the gateway both inside (inside a room) and outside (on a balcony) on the first floor of a building. There was a noticeable difference in radio range and we could not receive packets from the nodes that were placed far (on the other side of the highway).

Some LoRaWAN servers have restrictions and inefficiencies. For example, the popular Things Network [32] has a limit of only 30 seconds of air time per day, which is too small to conduct scientific experiments. Therefore, we chose LORIOT [32], which does not have such a limit on air-time. However, this server supports only 10 nodes and does not show the received packets in real-time. Hence, we developed a logger application at the gateway to get useful information, e.g. number of successful packet receptions, received signal strength, time-on-air, and time-stamps.

VIII. Simulation Experiments

This section provides some additional simulation-driven experiments that provides more insight on our algorithm.

A. Simulation Setup

We compare our proposed scheduling algorithm's (D-LLF) performance with four baseline scheduling algorithms: 1) Least Laxity First (LLF) [17], 2) Earliest Deadline First (EDF) [33], 3) Deadline Monotonic (DM) [34], and 4) Rate Monotonic (RM) [35] scheduling. We exclude ALOHA from this list since it almost always failed to find a feasible schedule in our simulation.

We use three comparison metrics: 1) schedulability ratio (i.e. ratios of schedules for which an algorithm finds a feasible solution), 2) deadline miss ratio (i.e. percentage of packets that miss the deadline for all links), and 3) buffer size (i.e. the maximum number of packets buffered at each node).

To simulate a LoRa network, we randomly choose a spreading factor from 7 to 12 for the links. We randomly choose 1–5 byte sized packets for each link. We assume a duty-cycle constraint $\delta = 1\%$ to calculate T_{off} of a link for a channel. To generate schedulable links, we set the period to the minimum $T_{off} + time\ on\ air$ among all links. We set the deadline to time-on-air multiplied by α, which is a random number between 1 and 5. All transmissions are released at the same time-slot.

All the simulations are performed on a 2013 MacBook Air having an Intel Core i5 dual-core processor and 4GB DDR3 RAM. The simulation software is written in Java.

B. Simulation Results

• (Figure 11) We compare the schedulability ratio of all five algorithms by varying the number of links to schedule from 8 to 40. Ten different link sets were generated for each test case. We assume eight channels. In Figure 11 we observe that the proposed D-LLF outperforms all four baselines for any number of links. For 8 links, D-LLF achieves a schedulability ratio of 1, whereas the baseline algorithms achieve 0.6. Since we keep the number of channels fixed, with an increased number of links, the schedulability ratio drops for all algorithms, because more links are contending for limited number of channels . Yet, the proposed D-LLF outperforms the baselines by scheduling $20\% - 40\%$ more link-sets on average.

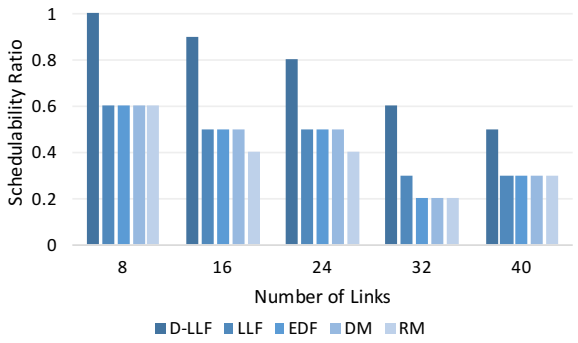


Fig. 11. D-LLF has better schedulability ratio with varying number of links.

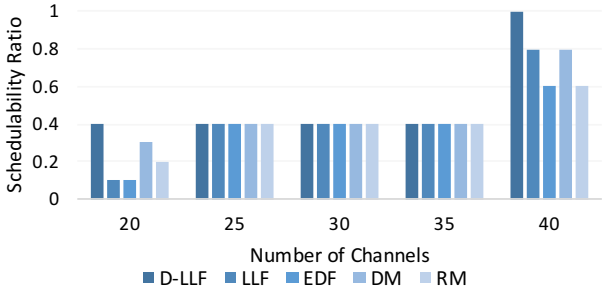


Fig. 12. D-LLF performs better than baseline algorithms with different number of number of channels

• (Figure 12) We vary the number of channels to see its effect on different algorithms. We use ten link-sets in this simulation, where each set has 40 links. α was chosen randomly between 1–2. In Figure 12 we observe that when the number of channels is 8, D-LLF achieves a schedulability ratio of 0.4, whereas the baselines achieve a maximum of 0.3. As the number of channels increase, the scheduling task becomes easier and the baseline algorithms catch up with the D-LLF. However, with 40 channels, D-LLF achieves a schedulability ratio of 1, whereas the baselines achieve a maximum of 0.8.

• (Figure 13) We evaluate the performance of the algorithms for a tight scenario where we set the deadline to the execution time or time-on-air ($\alpha = 1$). We use ten link-sets, each having 8 links. We use three different periods: $T1 = \min(T_{off}) + time\text{-}on\text{-}air(TOA)$, $T2 = 2T1/number\text{-}of\text{-}channels(\#Ch)$, and $T3 = 0.5T2$. From Figure 13 we observe that D-LLF outperforms all baselines by a large margin. D-LLF achieves a schedulabity ratio of 1 for both $T1$ and $T2$. On the other hand, LLF has the best schedulability ratio among the baselines, and achieves 0.5 and 0.4 for $T1$ and $T2$, respectively. For, $T3$ D-LLFś schedulability ratio drops to 0.4, which is still twice of LLF's.

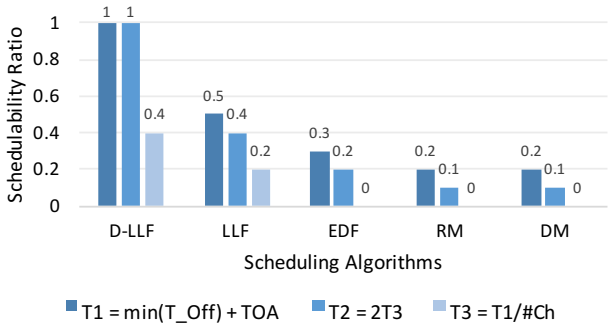


Fig. 13. D-LLF has better schedulability ratio with deadline being equal to execution time under different periods

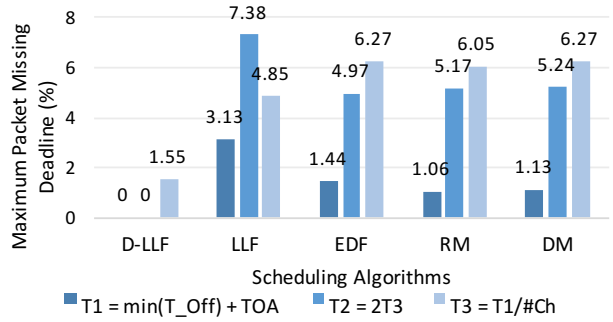


Fig. 14. D-LLF has least maximum percentage of packets missing deadline for different periods among all algorithms

• (Figure 14) For the same scenario as in Figure 13, we report the maximum percentage of packets that miss the deadline as a metric in Figure 14. We observe that D-LLF achieves 0 deadline miss for both $T1$ and $T2$. For $T3$, D-LLF's maximum deadline miss is 1.55%. On the other hand, LLF has the least maximum deadline miss of 4.85% among the baselines for $T3$.

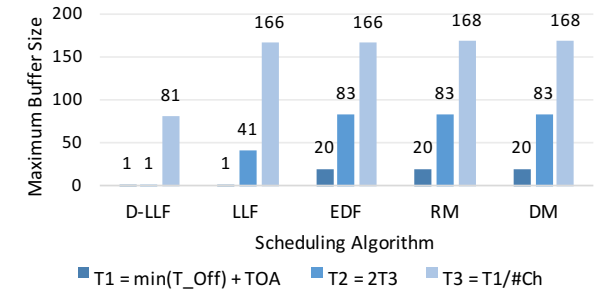


Fig. 15. D-LLF has the lowest maximum buffer size for different periods.

• (Figure 15) We compare the maximum buffer size of a node for different algorithms in Figure 15. D-LLF results in a maximum buffer size of 1, 1, and 81, for $T1$, $T2$, and $T3$, respectively. LLF has the smallest maximum buffer size among the baseline algorithms. For $T1$, $T2$ and $T3$, LLFś maximum buffer size reaches 1, 41 and 166, respectively, which is up to 41X larger than D-LLF.

IX. Related Work

Scheduling in wireless communication has been studied by many. [6] introduced a topology dependent transmission scheduling. [7], [36], [37] proposed distributed scheduling algorithms in wireless networks. [38] leverages wireless communication to achieve sub-nanosecond level clock synchronization. [39] analyzes time sensitive network protocols of IEEE 802.1 for their suitability to real-time communication. [40] proposes a dynamic network scheduling solution to minimize errors in a wireless control system. [41] introduced a network reconfiguration framework to tackle network delay, packet loss, and time-correlated link failures in wireless control system. [42] proposes an algorithm that minimizes the buffer space for target priority-aware network. [9], [43] used schedulability algorithms to minimize power consumption. However, none of these algorithms explicitly deal with duty-cycle constraints.

[11], [13], [14], [23] proposed time-division multiple access (TDMA) based scheduling algorithms for single channel wireless communication. In this paper, we are dealing with TDMA based multi-channel wireless communication network where selecting the channel is one of the challenges.

Multi-channel wireless communication scheduling has been explored in [12], [15], [16], [44], [45]. However, they do not assume any constraints on the duty cycle. We tackle the duty cycle constraint provided by LoRa network protocol.This constraint decreases the efficiency of the wireless network.

A few works have been done considering duty cycle constraint in wireless system. [10], [46]–[48] consider wireless networks with duty-cycle limit imposed on nodes. Here, nodes can not send or sense continuously, rather, they have to maintain a duty-cycle limit to reduce energy consumption. In this paper, the duty-cycle limit is imposed on a (node, channel) pair rather than only on the node.

In real time multi-processor scheduling [49]–[51] processor affinity has been considered such that there is a restriction on the migrations of any task to a specified subset of processors. We take inspirations from these multiprossesor scheduling works but solve our constrained wireless network problem differently.

X. Conclusion

We present the first duty-cycle-aware wireless link scheduling algorithm for LPWAN. We demonstrate the effect of duty-cycle on real-time link scheduling, illustrate the need for scoring wireless channels, and propose a scheduling algorithm that considers both the laxity of a packet and the availability of the channels. We implement a complete system by deploying a long-range LPWAN network in the city of Chapel Hill, NC. We evaluate the performance of the proposed algorithm in multiple real testbeds as well as with simulations.

REFERENCES

[1] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, Secondquarter 2017.

[2] "The lora alliance," https://www.lora-alliance.org/.

[3] http://www.weightless.org/.

[4] Wikipedia, "DASH7 — Wikipedia, the free encyclopedia," http://en.wikipedia.org/w/index.php?title=DASH7oldid=801485799, 2017, [Online; accessed 04-October-2017].

[5] S. K. M. Editor, "100 us cities covered by senet lora network for iot," Jun 2016. [Online]. Available: https://www.rcrwireless.com/20160615/internet-of-things/100-u-s-cities-covered-senet-lora-network-iot-tag17

[6] Z. Tang and J. J. Garcia-Luna-Aceves, "A protocol for topology-dependent transmission scheduling in wireless networks," in *WCNC. 1999 IEEE Wireless Communications and Networking Conference (Cat. No.99TH8466)*, 1999, pp. 1333–1337 vol.3.

[7] N. Vaidya, A. Dugar, S. Gupta, and P. Bahl, "Distributed fair scheduling in a wireless lan," *IEEE Transactions on Mobile Computing*, vol. 4, no. 6, pp. 616–629, Nov 2005.

[8] X. Liu, E. K. Chong, and N. B. Shroff, "Transmission scheduling for efficient wireless utilization," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2001, pp. 776–785.

[9] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, ser. WSNA '02. New York, NY, USA: ACM, 2002, pp. 32–41. [Online]. Available: http://doi.acm.org/10.1145/570738.570744

[10] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 4, March 2005, pp. 2470–2481 vol. 4.

[11] D. Panigrahi and B. Raman, "Tdma scheduling in long-distance wifi networks," in *IEEE INFOCOM 2009*, April 2009, pp. 2931–2935.

[12] F. Jia, B. Mukherjee, and J. Iness, "Scheduling variable-length messages in a single-hop multichannel local lightwave network," *IEEE/ACM Trans. Netw.*, vol. 3, no. 4, pp. 477–488, Aug. 1995. [Online]. Available: http://dx.doi.org/10.1109/90.413222

[13] T. W. Carley, M. A. Ba, R. Barua, and D. B. Stewart, "Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks."

[14] K. Liu, N. Abu-Ghazaleh, and K.-D. Kang, "Jits: Just-in-time scheduling for real-time sensor data dissemination," in *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*. IEEE, 2006, pp. 5–pp.

[15] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for wirelesshart networks," in *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*. IEEE, 2010, pp. 150–159.

[16] X. Wang, G. B. Giannakis, and A. G. Marques, "A unified approach to qos-guaranteed scheduling for channel-adaptive wireless networks," *Proceedings of the IEEE*, vol. 95, no. 12, pp. 2410–2431, Dec 2007.

[17] S.-H. Oh and S.-M. Yang, "A modified least-laxity-first scheduling algorithm for real-time tasks," in *Real-Time Computing Systems and Applications, 1998. Proceedings. Fifth International Conference on*. IEEE, 1998, pp. 31–36.

[18] "Project real-time lora," http://lora.web.unc.edu/.

[19] "Chirp signal," https://en.wikipedia.org/wiki/Chirp.

[20] "Doppler effect," https://www.nutaq.com/blog/doppler-shift-estimation-and-correction-wireless-communications-0.

[21] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[22] N. Abramson, "The aloha system: another alternative for computer communications," in *Proceedings of the November 17-19, 1970, fall joint computer conference*. ACM, 1970, pp. 281–285.

[23] O. Chipara, C. Lu, and G.-C. Roman, "Real-time query scheduling for wireless sensor networks," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*. IEEE, 2007, pp. 389–399.

[24] "Federal communications commission(fcc)," https://www.fcc.gov/.

[25] "Lora dragino shield," http://www.dragino.com/products/module/item/102-lora-shield.html.

[26] "Arduino uno," https://store.arduino.cc/usa/arduino-uno-rev3.

[27] "Semtech sx1272," http://www.semtech.com/wireless-rf/rf-transceivers/sx1272/.

[28] "Ibm lmic," https://github.com/matthijskooijman/arduino-lmic.

[29] "Timer module," https://partnums.com/gtin/00747465491461.

[30] "Multitech conduit," https://www.thethingsnetwork.org/docs/gateways/multitech/.

[31] "Loriot, lorawan services," https://us1.loriot.io/.

[32] https://www.thethingsnetwork.org/.

[33] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo, *Deadline scheduling for real-time systems: EDF and related algorithms*. Springer Science & Business Media, 2012, vol. 460.

[34] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings, "Real-time scheduling: the deadline-monotonic approach," in *in Proc. IEEE Workshop on Real-Time Operating Systems and Software*. Citeseer, 1991.

[35] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. G. Harbour, *A practitioner's handbook for real-time analysis: guide to rate monotonic analysis for real-time systems*. Springer Science & Business Media, 2012.

[36] T. Zhang, T. Gong, C. Gu, H. Ji, S. Han, Q. Deng, and X. S. Hu, "Distributed dynamic packet scheduling for handling disturbances in real-time wireless networks," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 261–272.

[37] T. Gong, H. Ji, S. Han, T. Zhang, C. Gu, X. S. Hu, and M. Nixon, "Demo abstract: A cross-device testing and reporting system for large-scale real-time wireless networks," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 157–158.

[38] A. Dongare, P. Lazik, N. Rajagopal, and A. Rowe, "Pulsar: A wireless propagation-aware clock synchronization platform," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 283–292.

[39] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Synchronization quality of ieee 802.1as in large-scale industrial automation networks," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 273–282.

[40] W. Wang, D. Mosse, J. G. Pickel, and D. Cole, "Work-in-progress: Cross-layer real-time scheduling for wireless control system," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017.

[41] ——, "Work-in-progress: Wireless network reconfiguration for control systems," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 145–148.

[42] H. Kashif and H. Patel, "Buffer space allocation for real-time priority-aware networks," in *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2016, pp. 1–12.

[43] T. ElBatt and A. Ephremides, "Joint scheduling and power control for wireless ad hoc networks," *IEEE Transactions on Wireless communications*, vol. 3, no. 1, pp. 74–85, 2004.

[44] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and real-time communication in industrial wireless mesh networks," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE*. IEEE, 2011, pp. 3–12.

[45] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen, "Schedulability analysis under graph routing in wirelesshart networks," in *Real-Time Systems Symposium, 2015 IEEE*. IEEE, 2015, pp. 165–174.

[46] L. Cheng, J. Niu, Y. Gu, T. He, and Q. Zhang, "Energy-efficient statistical delay guarantee for duty-cycled wireless sensor networks," in *Sensing, Communication, and Networking (SECON), 2015 12th Annual IEEE International Conference on*. IEEE, 2015, pp. 46–54.

[47] Y. Gu and T. He, "Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 12, pp. 1741–1754, 2011.

[48] Y. Gu, T. He, M. Lin, and J. Xu, "Spatiotemporal delay control for low-duty-cycle sensor networks," in *Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE*. IEEE, 2009, pp. 127–137.

[49] V. Bonifaci, B. Brandenburg, G. DAngelo, and A. Marchetti-Spaccamela, "Multiprocessor real-time scheduling with hierarchical processor affinities," in *Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on*. IEEE, 2016, pp. 237–247.

[50] S. Baruah and B. Brandenburg, "Multiprocessor feasibility analysis of recurrent task systems with specified processor affinities," in *Real-Time*

Systems Symposium (RTSS), 2013 IEEE 34th. IEEE, 2013, pp. 160–169.

[51] A. Gujarati, F. Cerqueira, and B. B. Brandenburg, "Multiprocessor real-time scheduling with arbitrary processor affinities: from practice to theory," *Real-Time Systems*, vol. 51, no. 4, pp. 440–483, 2015.