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 because of their long range and a battery life of several years, and a data rate of 25-50kbps, which is sufficient to carry IoT traffic. However, a practical limitation of a LPWAN-based real-time wireless network is the duty-cycle limit 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 wireless channels. The proposed algorithm is implemented in a five-node, wide-area outdoor 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 algorithm outperforms standard scheduling algorithms in terms of link schedulability, deadline misses, and buffer size.

Keywords—Sensor Network; Real-time Scheduling; LPWAN

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[22] have become very popular, as they are an excellent fit to the IoT data traffic that are generated and consumed by many smart city 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 [7].

Among different choices of low power WANs, we study one of the most popular technologies of today, which is called the LoRa WAN [1]. 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. 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 for a certain period after each successful transmission. In EU, LoRa has a strict duty-cycle limit of 1%, and in the USA, the duty-cycle is configured by the network administrator.

For instance, when the duty-cycle limit is 1%, a device that has recently used a specific wireless channel for 10ms has to wait for another 990ms for that channel to be available to it again. The device, however, can send packets 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 long range wireless communications, bands below 1 GHz are used. Because of the long radio range, a large number of devices (in a large geographic area) compete for the same frequencies, and their transmissions are more susceptible to collision. Hence, duty-cycle limits are imposed by the authority (or an admin) to ensure fair access to the air for all devices.

We consider the duty-cycle 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. Examples of such real-time wide-area IoT applications include monitoring vehicles in smart cities for detecting and predicting traffic congestion, smart parking, early detection of wildfire and volcanic eruption, monitoring with swarms of nano drones. All these applications require a certain level of guarantee on real-time wireless communication.

The generic problem of scheduling wireless transmissions dates back to decades [18, 26, 28]. Theoretical analysis as well as results from practical deployments have been published on various categories of real-time wireless net-
works such as ad-hoc and sensor networks [19, 27] and WiFi [21]. The problem we study in this paper has similarity to several of these works that consider single-hop network topology [16], time division multiple access (TDMA)-based link scheduling approaches [6, 17], use of laxity to schedule packets [24], and channel selection [29]. 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 [20] 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 [2]. 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 called Duty-cycle aware Least Laxity First (D-LLF) that explicitly handles the duty-cycle constraints in LPWANs. The time-complexity of this offline 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 [2].

II. BACKGROUND

A. Overview of LoRaWAN

LoRa 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 0.6 miles (non 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.

![LoRaWAN Network Architecture](image)

Figure 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. 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.

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). The use of chirps improves its robustness against interference, Doppler effect, and multipaths. Each symbol is encoded with 2\(SF\) chirps, where \(SF\) is called the spreading factor and takes a value between 7 to 12.

- **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: \(T_a = n_s \times 2^{SF} \times T_c\)

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: \(T_a = n_s \times 2^{SF}/BW\)

- **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 [4]. 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 \(\sigma\), an end-device must not transmit anything on the most recently used channel for a minimum off-period, \(T_{off}\)

\[
T_{off} = T_a \times \left( \frac{1}{\sigma} - 1 \right) \quad (1)
\]

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 [3] protocol for an uplink packet transmission, followed by two short downlink receive windows. 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. An increase in the number of devices on the network causes more collision.

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 is out of the scope of this paper.

![Diagram](https://example.com/diagram.png)

Figure 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 \( A_i \). For
simplicity, we consider the spreading factor to be constant
for all links. Since our algorithm takes duty-cycle limit into
consideration, it is able to handle varying spreading factor.
We denote the \( k \)-th packet generated at link \( L_i \) by \( \tau_{ik} \). 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 other’s 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 \( k \)-th packet transmission \( \tau_{ik} \), 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 1.

Given a set of links \( L = \{(N_i, M_j, T_i, A_i, D_i)\} \), duty-
cycle limit \( \delta \), 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 [9, 17, 24]. 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. In a multiprocessor
scheduling scenario, this is analogous to a scheduling prob-
lem where a processor becomes unavailable to a task for a
certain period, after the processor has been used by the task
recently.

<table>
<thead>
<tr>
<th>Link</th>
<th>Release Time</th>
<th>Time-On-Air</th>
<th>Deadline</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_1 )</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>( L_2 )</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

For example, lets consider a network with two end-devices
\( N_1 \) and \( N_2 \), and one gateway \( G \). 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 \( L_1 \), \( \tau_{11} \) uses channel \( C_1 \)
  and packet transmission of \( L_2 \), \( \tau_{21} \) uses channel \( C_2 \). In Figure 3a
  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 \( \tau_{12} \) and \( \tau_{22} \). Both packets have the same laxity of
  1. Suppose, \( \tau_{12} \) chooses channel \( C_1 \), which is currently
  available to it. Because traditional scheduling algorithms
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, \( \tau_{22} \) 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 \( L_2 \), and \( \tau_{22} \) can use it for transmission. However, from
Figure 3b, we see that \( \tau_{22} \) still misses the deadline.

- At time-slot 5, we have another option, which is shown
  in Figure 3c. Suppose, \( \tau_{12} \) chooses channel \( C_2 \) this time,
as it is also available to it at time-slot 5. Given this, \( \tau_{22} \)
can use channel $C_1$ for its transmission, which is available and does not restrict $\tau_{22}$ at that moment due to duty cycle limit. Therefore, by using $C_1$, $\tau_{22}$ makes the deadline. 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 $\tau_{12}$. 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:

$$G_r(C_i, t) = \max \left\{ G_r(C_i, t-1) - 1, t_{off} \{L_j, C_i, A_j, \sigma\} \right\}$$

B. An Example

We revisit the example from section IV 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 $\tau_{11}$ 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 $\tau_{21}$, 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 $\tau_{12}$, as opposed to $C_1$. This enables $L_2$ to choose $C_1$ for its second transmission, which is our desired schedule.

C. Duty-Cycle Aware Least Latency First(D-LLF)

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

- **Packet Selection**: In the first step, among all ready-to-go packets, the one with the least laxity [20] 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 [24].

- **Channel Selection**: In the second step, among all available channels for the selected link, the one with the highest gravity is selected for transmission. After using the channel, the gravity is updated according to Equation 2.

Due to space limitations, the pseudo code and complexity analysis of the algorithm is shown in [15]

VI. SYSTEM DEVELOPMENT

A. Developing the LoRa Nodes

We develop LoRa nodes in our lab by interfacing a LoRa radio shield with an Arduino Uno that hosts an ATmega328P microcontroller. The radio shield internally uses a transceiver SX1272/73 which is controlled from the Arduino using a modified software library. 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 with the Arduino board. This real-time clock has no overhead on a LoRa node’s battery-life as it is powered by its own battery and draws minimal current (110-300 μA) to enjoy a lifetime of several months at continuous operation. Both the modified library and our customized application are written in C. Our source code is open and accessible online from here [2].

B. Configuring the Gateway

We use a Multitech Conduit device as the gateway. This is a configurable Internet gateway for industrial IoT applications where LoRa is used for the local wireless network. Figure 4(a) shows a photo of the main elements of our setup. The gateway listens to one sub-band at a time, and therefore, a gateway can listen to eight channels simultaneously. The setup is shown in Figure 4(b).

VII. REAL-WORLD DEPLOYMENT

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, are chosen to place the nodes. We position the gateway in the balcony on the first floor of a two storied building. We placed the gateway inside the building as it is powered from an electric outlet. The nodes are placed around the gateway within a radius of 220m and are powered by USB power banks. Note that although the maximum reported range of LoRa in non-line-of-sight scenario in the literature is 863m [1], we obtain a shorter range than this to ensure reliable communication. Prior to choosing the exact locations of the nodes, we perform a day long survey to measure signal strengths and the reliability of the communication links at various locations in the test area. Finally, we select the locations where we observe the least packet drops and that are 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 ∼125KHz.

For the indoor test-bed, we place the nodes and the gateway inside the Computer Science building at the UNC. The gateway is placed on the second floor of the building. Two of the nodes N1 and N2 are placed on the same floor, but in different rooms. N3, N4, and N5 were placed on the ground, the first, and the third floor, respectively.

We send one-byte payloads from four of the nodes and five-byte payloads from a node. We send different sizes of payloads to see its effect on our algorithm. Given the Ai and t_{off} for one-byte payloads, we set the period of each node to (A_i + t_{off}). We empirically determine that this is the minimum period to obtain schedulable links. 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-periods [14].

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 Figure 5a, 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.

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. 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. In both cases, D-LLF outperforms LLF as it chooses channels based on gravity and thus is able to mitigate the effect of duty-cycle limit.

VIII. SIMULATION EXPERIMENTS

A. Simulation Setup

We compare our proposed scheduling algorithm’s (D-LLF) performance with 5 baseline scheduling algorithms: 1) Least Laxity First (LLF) [20], 2) Earliest Deadline First (EDF) [25], 3) Deadline Monotonic (DM) [25], 4) Rate Monotonic (RM) [25] scheduling, and 5) ALOHA. ALOHA always failed to find a feasible schedule in our simulation. Therefore, we do not report its performance in the results.

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. This does not
have any effect on the performance of our algorithm as we define gravity based on duty-cycle, but it affects other algorithms. Since IoT devices send data in small chunks, we randomly choose 1–5 byte sized packets for each link. We assume a duty-cycle constraint of $\delta = 1\%$ 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. This is an empirically obtained lower bound on the period to get the maximum schedulability for all algorithms. We set the deadline to time-on-air multiplied by $\alpha$, which is a random number between 1 and 5. As the value of $\alpha$ gets lower, the scenario becomes harder to schedule. The first transmission of all links are released at the same time slot (time slot zero).

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 5b) We compare the schedulability ratio of all five algorithms by varying the number of links to schedule from 8 to 40. We start from 8 as we assume the links have 8 channels to be scheduled and for more than 40 links, the performance of all algorithms drop significantly. Ten different link sets were generated for each test case. In Figure 5b 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. Note that the baseline algorithms achieve similar results when the deadline is too large, and therefore, the duty-cycle limit practically has no effect.

- (Figure 5c) 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. In Figure 5c 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 5d) We evaluate the performance of all 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: $T_1 = \text{min}(T_{off}) + \text{time-on-air}(TOA)$, $T_2 = 2T_1/\text{number-of-channels}(#Ch)$, and $T_3 = 0.5T_2$. We choose $T_1$ in the same way as earlier simulations. To make the workload harder to schedule, we set $T_2$ and $T_3$ such that each link has to use all the channels more frequently. In Figure 5d we observe that D-LLF outperforms the baselines by a large margin. D-LLF achieves a schedulability ratio of 1 for both $T_1$ and $T_2$. On the other hand, LLF has the best schedulability ratio among the baselines, and achieves $0.5$ and $0.4$ for $T_1$ and $T_2$, respectively. For, $T_3$ D-LLF’s schedulability ratio drops to 0.4, which is still twice of LLF’s.

- (Figure 5e) For the same scenario as in Figure 5d, we
report the maximum percentage of packets that miss the deadline as a metric in Figure 5e. 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 smallest maximum deadline miss of 4.85% among the baselines for T3.

• (Figure 5f) We compare the maximum buffer size of a node for different algorithms in Figure 5f. 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, LLF’s maximum buffer size reaches 1, 41 and 166, respectively, which is up to 41X larger than D-LLF. Since D-LLF is able to schedule more transmissions, it requires a lower buffer size than others.

IX. RELATED WORK

Scheduling in wireless communication has been studied by many. [26] introduced a topology dependent transmission scheduling. [11, 28] proposed distributed scheduling algorithms in wireless networks. [10, 27] used schedulability algorithms to minimize power consumption. However, none of these algorithms explicitly deal with duty-cycle constraints. [9, 17] 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 [14, 23, 24]. 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. [8, 12] 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 [5, 13] 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 multiprocessor 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. For future work, we want to propose a distributed scheduling system for a decentralized network setup.

XI. ACKNOWLEDGMENT

This paper was supported, in part, by NSF grant CNS-1704469.

REFERENCES

[3] N. Abramson. The aloha system: another alternative for computer communica-