Beartooth Relay Protocol: Supporting Real-Time Application Streams with Dynamically Allocated Data Reservations over LoRa

Batuhan Mekiker  
Montana State University  
Bozeman, MT, USA  
batuhanmekiker@montana.edu

Mike P. Wittie  
Montana State University  
Bozeman, MT, USA  
mike.wittie@montana.edu

Jefferson Jones  
Beartooth Radio, Inc.  
Bozeman, MT, USA  
jeffersonmjones92@gmail.com

Michael Monaghan  
Beartooth Radio, Inc.  
Bozeman, MT, USA  
michael@beartooth.com

Abstract—Convenient solutions that cater to different types of applications always gather more users. LoRa, one of the solutions in the IoT space is quite flexible at the physical layer, but link layer LoRa protocols lack similar flexibility. As a result, their performance remains suitable for the exchange of short messages in sensor networks, but does not extend to real-time and streaming IoT applications. Beartooth Relay Protocol (BRP) aims to address these limitations with a configurable link layer solution without compromising performance to sustain real-time flows over LoRa. We demonstrate that BRP expands the performance envelope of LoRa and makes it a suitable PHY technology for a wide range of IoT applications.

Index Terms—LoRa, IoT, Real-Time, Low Latency, Streaming Data, Medium Access Control

I. INTRODUCTION

People gravitate to convenient, one-fits-all solutions and expect products to work well for different applications in various scenarios. The same holds true for wireless networks. Bluetooth, for example, supports wireless music streaming, file sharing, and even control of mobile applications via car receivers. As of yet, networks in the Sub-GHz Industrial, Scientific, and Medical (ISM) band lack the same flexibility and primarily cater to IoT sensor communications [1]–[6].

One of the recent and exciting physical layer protocols in the ISM band is Long Range (LoRa) from Semtech [7]. LoRa’s chirp spread spectrum (CSS) modulation makes possible long-range transmissions with low power consumption. LoRa’s physical layer is also quite configurable in terms of bandwidth, transmit power, coding rate and spreading factor giving LoRa a broad performance envelope in terms of range and data rate [8]. However, link-layer protocols designed for LoRa do not fully take advantage of LoRa’s flexibility and cater to specific applications in specific settings [2], [4], [5], [9]–[12]. As a result, LoRa remains a niche technology restricted to sensor networks, rather than broader Internet of Things (IoT) use cases.

To illustrate the above claim, LoRaWAN [13] is a link layer protocol base on LoRa for long-range, low-power, short-burst sensor communications. However, the collision avoidance mechanisms of LoRaWAN borrowed from ALOHA [14], lead to unpredictable message delay and loss. The resulting mean latency of 11 s and throughput of 28 bit/s make it unsuitable for real-time and reliable applications such as health monitoring [4], [13]. To provide predictable delay in LoRa networks several solutions proposed time-slotted medium access control (MAC) [2], [3], [9]. These protocols provide bounds on delay and high packet delivery rates (PDR). Yet, these protocols are hamstrung by the European Union (EU) regulatory duty cycle limitations on the ISM band, and so have low throughput and latency in the tens of seconds that limits their use for streaming applications [15]. A few approaches aim to sustain streaming data by abandoning PDR as a primary metric and focusing in improving LoRa throughput instead. Industrial LoRa [5] combines contention and contention-free transmissions to provide sustained throughput of 28 bit/s. DQ-LoRa [12] uses distributed queuing to provide throughput of 0.7 kbit/s. Nevertheless, these protocols still achieve latency of more than 4 s limiting their use in real-time applications such as vehicular networks [16]. In summary, none of the existing LoRa MAC protocols support real-time data streams and moreover lack the flexibility to customize their parameters to meet the requirements of the specific application flows present in a network deployment.

We propose Beartooth Relay Protocol (BRP), a flexible MAC protocol that supports, among others, real-time and streaming applications over LoRa. We also present the design of a frequency hopping LoRa radio developed by Beartooth that works in tandem with the BRP. This paper offers the following contributions:

1) We describe BRP, a highly configurable gateway protocol supporting real-time and streaming applications. We control the BRP performance envelope with a configuration file distributed to nodes in a network deployment to match application performance requirements.

2) BRP provides latency under 500 ms making it suitable for real-time messaging, such as location updates within the Team Awareness Kit (TAK) tactical situational awareness application [17].

3) BRP provides flow throughput just shy of 0.8 kbit/s letting it support bidirectional, real-time voice flows (a first in
LoRa) encoded with Codec 2 [18].

4) We describe the mechanisms behind the Beartooth radio and BRP, specifically frequency hopping and scheduling of multiple transmission opportunities, that enable their performance while meeting Federal Communications Commission (FCC) regulations.

These results demonstrate that BRP has the potential to be adopted into many IoT applications with different performance profiles that go beyond sensor network communications.

The rest of this paper is organized as follows. In Section II we discuss the limitations of existing LoRa technologies. Section III introduces the Beartooth radio and its frequency hopping mechanism. Section IV details the BRP. In Section V we present a measurement study of a Beartooth network. Finally, we conclude and present directions for future work in Section VI.

II. RELATED WORK

To frame the need for a flexible LoRa protocol that caters to both real-time and streaming data, we discuss the limitations of existing LoRa protocols in the commercial and research spaces.

Semtech introduced LoRa chirp spread spectrum (CSS) chip in 2009. LoRa encodes information with chirps, or transmissions of rising frequencies within the width of a channel (125 kHz or 250 kHz), where the starting frequency of chirp indicates a symbol [8]. The slope of a chirp is a function of channel width and the spreading factor, which defines the duration of each chirp and its resiliency to radio interference. The wider the channel and the longer the chirp the more resilient is the transmission to interference as these make it easier for the receiver to determine the starting frequency of the chirp. The CSS encoding gives LoRa resilience to multipath effects, fading, and Doppler frequency shifts [19]. However, the main advantage of LoRa is its range; radios based on Semtech chips support data rates between 0.3 and 37.5 kbit/s and robust links at distances up to 9 km in urban and over 30 km in rural scenarios [13], [20].

The LoRa Alliance publishes a carrier sensing multiple access (CSMA) LoRaWAN protocol, which allows devices to communicate via gateways [21]. LoRaWAN utilizes an ALOHA based approach where end devices transmit containing frames, which leads to error rates as high as 70% depending on the frame size [9]. Although LoRaWAN utilizes other mechanism to boost performance such as Adaptive Data Rate (ADR) and Channel Activity Detection (CAD) collision avoidance protocol design is ill-suited for application streams with Quality of Service (QoS) requirements [22].

To address the unpredictable latency and unreliable delivery of LoRaWAN a number of protocols propose time-slotted approaches to medium sharing. Hoang et al. propose ST/CA – a slotted LoRa protocol with collision avoidance [9]. A gateway beacon synchronizes nodes to the start of a frame divided into transmission slots. Each transmission slot contains a number of delay slots. To transmit data a node engages CAD in a random delay slot and, if is does not detect a competing signal, proceeds to transmit data for the remaining delay slots and the rest of the transmission slot. While time synchronization and short delay slots reduce the impact of collisions, nodes still compete for each transmission opportunity. The evaluation shows maximum per node throughput of 11.3 bit/s under a PDR of 0.87. Frame duration is 25.7 s leading to the maximum latency of 25.65 s after a 500 ms beacon.

Piyare et al. propose an on-demand time division multiple access (TDMA) scheme for LoRa [10]. A gateway uses a separate wake-up receiver (WuRX) radio to wake up nodes and synchronize them to the start of a cycle. A node then chooses a transmission slot based on its node ID. The scheme assigns node IDs statically during network configuration, which makes it unable to deal with node churn and leads to higher latency for higher ID nodes. The evaluation shows a PDR of 100%, but high mean latency of over 2 s. The authors do not provide throughput performance.

Zorbas et al. propose TS-LoRa – a slotted LoRa protocol that allows nodes to autonomously compute their slot number [2]. TS-LoRa nodes operate in two stages, registration and transmission. During registration the gateway assigns a unique timeslot to a registering node, which the node then uses to transmit data. The gateway initiates data transmission with a SACK packet that also acknowledges previous transmissions. The scheme however leaves timeslots unused for nodes that chose not to transmit in their slot. The evaluation shows maximum per node throughput of 45.6 bit/s under a PDR of 0.9986.

Singh et al. propose a gateway-coordinated channel hopping scheme [11]. The protocol uses detailed on-air time calculation for beacon packets to synchronize node clocks. The gateway announces a schedule of timeslots and channels than nodes switch to for transmission. The paper evaluates the performance of scheduling and channel hopping scheme, but not of data transmission.

Leonardi et al. propose Industrial LoRa MAC that provides both contention and contention-free transmission slots within a frame [5]. Nodes compete for contention slots using ALOHA, while contention-free slots are scheduled. However, Industrial LoRa include time in the frame to communicate a schedule and the authors propose that the schedule should be specified offline, with dynamic scheduling left to future work. The authors evaluate Industrial LoRa in an OMNeT++ simulation showing best-case mean latency of 9.67 s and PDR of 0.34 for contention slots and 1.0 for contention-free slots.

Wu et al. propose DQ-LoRa based on distributed queuing (DQ) [12]. A DQ-LoRa frame is divided into a small number of contention slots, a data slot, and a gateway acknowledgement slot. Nodes compete for data slots by transmitting their ID in one of the contention slots and the gateway acknowledges non-colliding IDs. Nodes that do not experience a collision use the data slots in subsequent frames to transmit, while other nodes continue to compete in contention slots. The work demonstrates that a small number of contention slots is sufficient to fully utilize data slots, even with a large number of nodes. The authors evaluate DQ-LoRa analytically showing
bots-case delay of around 4 s and throughput of 0.7 kbit/s, a gain factor of 2.6 over LoRaWAN throughput.

Finally, Leonardi et al. underline the need for bounded end-to-end delay and higher reliability in IoT applications and introduce RT-LoRa to address delay and reliability with scheduling for real-time traffic [4]. The RT-LoRa is able to accommodate contention and contention-free transmissions for periodic and aperiodic data generation. Their results show RT-LoRa’s packet loss rate and maximum end-to-end delay under two configurations. When compared to author’s earlier approach Industrial LoRa, RT-LoRa has significantly higher PDR of 0.97 especially on shorter distances to the sink and latency of 20.7 s in best case scenario.

In summary, none of the approaches address the needs of latency sensitive real-time traffic, or streaming data flows. In this paper, we propose a solution that applies time-slotted frame scheduling on a frequency hopping LoRa radio. Our approach not only noticeably reduces LoRa delay and increases throughput, but also provides the flexibility to configure network parameters to the performance needs of real-time and streaming applications present in a network deployment.

![Image](image_url)

**Fig. 1.** Beartooth radio LoRa shield paired with a Raspberry Pi 4.

### III. BEARTOOTH RADIO

The Beartooth radio behind the BRP is a custom LoRa shield paired with a Raspberry Pi 4 controller, as shown in Figure 1. The shield includes a SX1276 LoRa chipset, which communicates with the BRP on the controller’s CPU via the Pi’s Serial Peripheral Interface (SPI). The SX1276 chipset modulates a CSS radio signal in the 900 MHz band amplified to 30 dbm at the antenna. The SX1276 provides seven spreading factors, SF6 to SF12, with SF6 creating the steepest slope providing the highest data rate, 37.5 kbit/s, and SF12 creating the flattest slope providing the greatest robustness and therefore range with 0.26 kbit/s [6]. LoRa also protects bits in transmission with a configurable error correction rates using Hamming codes and with a cyclic redundancy check (CRC).

One of our contribution is the frequency-hopping mechanism used by the Beartooth radio since 2017 [23]. The FCC Title 47 part 15 limits the maximum transmission duration on a channel, also known as dwell time, to 400 ms in the ISM band [24]. This regulation limits the throughput LoRa devices can achieve on a single frequency. However, SX1276 chipset supports internally timed frequency hops [19] and Beartooth radios change frequencies during message transmissions every 0.08 s, which effectively allows for indefinite transmission durations. Because LoRa supports 50 frequency channels, Beartooth radios use 50 semi-orthogonal hopping sequences. A relay and its clients agree on a hopping sequence and multiple sequences allow the coexistence of co-located network deployments. We handle occasional collisions on sequence timeslots with forward error correction (FEC) embedded in Beartooth frames. Beartooth radios and their approach to the use of the ISM band has been approved for operation by the FCC [23].

### IV. BEARTOOTH RELAY PROTOCOL (BRP)

#### A. Protocol Requirements

In designing the BRP we had to consider Beartooth customer requirements, constraints of the LoRa chipset, and FCC regulations. Beartooth customers want to build networks that cover hundreds of square miles and support support two types of applications: situational awareness and team voice communications. The situational awareness application exchanges short messages that carry text, or GPS location. These messages are under 20 B and should be delivered in under 500 ms. The team voice application sends encoded voice streams that should be delivered to multiple recipients, again, in under 500 ms. We encode voice transmissions with Codec 2 for a throughput requirement of 700 bit/s [18]. We also explore a scenario to support generic sensor network compliant with EU duty cycle limitations in the ISM band, where duty cycle is restricted to 1% [5]. Messages in this network should achieve a PDR above 90% as in existing LoRa protocols [2], [4]. This application allows us to compare BRP performance to other EU-compliant LoRa protocols.

We aim to support these applications within the limits placed by the SX1276 chipset. The SX1276 chipset provides 50 10.9 kbit/s channels at SF7, which forces a uniquely pithy BRP control signalling within the available bandwidth, as discussed in Section V-C. Further, we determined experimentally that the SX1276 chipset faces limitations in transmitting back-to-back frames. For example, frames over 30 B transmitted every 150 ms result in the chipset becoming unresponsive until we cycle its power. The chipset, however, can transmit repeatedly frames under 30 B at that interval, or frames larger than 30 B are longer intervals. As a result of these hardware constraints, we configure node data frames under 30 B, but may still use larger frames for relay transmissions, as discussed in Section IV-B.

Finally, the FCC limits transmit power to 30 dbm [24]. The power limits restrict the communication range of Beartooth radios to 15.2 km as reported in our preliminary work [25]. Our transmissions also comply with FCC’ dwell time regulations, as described in Section III where a radio cannot occupy one channel more than 400 ms [24].

3
Fig. 2. BRP cycle, stages, and frames.

Frames | Fields | Size (B) | Description
--- | --- | --- | ---
RLY_ANNC | type | 1 | Relay announces available LES control time slots and the network configuration.
 | confid_id | 1 |  
 | ctrl_tbl | 2 |  
ND_REQ | type | 1 | Node requests a number of DE time slots for data transmission.
 | node_id | 4 |  
 | de_req_cnt | 1 |  
RLY_ACK | type | 1 | Relay broadcasts DE time slot allocations and the number of DE stages.
 | traffic_map | 12 |  
 | traffic_map_size | 1 |  
 | de_cnt | 1 |  
ND_DATA | type | 1 | Node transmits their data.
 | dest_id | 4 |  
 | data_len | 1 |  
 | data | 20 |  
RLY_TX | type | 1 | Relay rebroadcasts received ND_DATA frames in RLY_TX.
 | nd_data_len | 1 |  
 | nd_data_buffer | 78 |  

TABLE I. List of protocol frames with descriptions.

Fig. 3. Link Establishment and Scheduling message sequence.

B. Protocol Operation

At a high level BRP operates by repeating a transmission cycle of protocol frames shown in Figure 2. We list the details of the BRP frames in Table I. Each cycle contains two types of stages, the Link Establishment & Scheduling (LES) stage and the Data Exchange (DE) stage. In the LES stage the relay synchronizes the nodes to the cycle start time and allows them to request transmission opportunities. In the DE stages nodes send data to the relay, which forwards the data the receivers via a broadcast. Depending on the amount of data nodes seek to transmit, the relay may allocate transmission opportunities in consecutive DE stages. While nodes compete in the LES stage and their request frames may collide, the data transmissions in the DE stages are collision-free.

a) Link Establishment: Referring to Figure 3, to start a cycle a relay broadcasts the Relay Announce (RLY_ANNC) frame (Step 1), which includes configuration ID (confid_id) and a LES control timeslot table (ctrl_tbl) with available timeslots for nodes to establish connections. The confid_id specifies network parameters, discussed in Section IV-C.

A node may listen to frame preambles on different frequencies (in turn) to receive RLY_ANNC frames from different relays in an area. A node will chose the one with the strongest signal to noise ratio (SNR) and thereafter listen for RLY_ANNC frames on that channel. Nodes use the reception of RLY_ANNC to synchronize with a relay by establishing the start time of a cycle. If a node does not receive RLY_ANNC in some time, then it sequentially listens other channels and accepts announcements from other relays.

To connect with a relay, a node chooses a random available LES timeslot from ctrl_tbl (Step 2 in Figure 3) in which to send the Node Request (ND_REQ) frame containing the node ID (node_id) and how many DE stages (de_req_cnt) it needs for its traffic (Step 3). For example, to send a ND_REQ in timeslot 2, the node waits for the time it takes to transmit two ND_REQ frames after the reception of RLY_ANNC.

b) Scheduling: The relay collects ND_REQ’s and adds node_id’s into a traffic queue used to schedule nodes in the DE stages. For simplicity, we implement a simple sticky scheduler, which gives priority to continuing flows from the previous cycle up to a limit. Otherwise the relay schedules new requests in the traffic queue randomly. The stickiness allows nodes to effectively reserve bandwidth for continued data streams across multiple cycles, while the limit and randomness provide fairness. The result of the scheduling
The process is a traffic map (traffic_map) containing node_id's corresponding to DE timeslots of each scheduled node. For example, a traffic map [215, 328, 328] means that node 235 may transmit data in DE timeslot 0 and node 328 in timeslots 1 and 2, if it sent two ND_REQ's.

Following the scheduling decision, the relay updates its ctrl_tbl (Step 4) by setting the bits in the LES timeslots used by ND_REQs of the scheduled nodes.

The number of DE stages used by the relay provides a tradeoff between latency and throughput as discussed in Section V-B. The number of DE stages may be fixed at the relay, though we implement a dynamic approach where the number of DE stages is the median of the ND_REQ de_req_cnt values to provide a balance between fairness and performance.

To announce the scheduling decision, the relay forms a Relay Acknowledgement (RLY_ACK) frame (Step 5) by including the traffic_map and the number of DE stages (de_cnt). A node receiving a RLY_ACK (Step 6) checks if its node_id is in the traffic_map and if so considers itself scheduled on the LES timeslot it used to send its ND_REQ. The node also records its DE timeslot (the position of its node_id in the traffic_map) and the number of DE stages.

If two ND_REQ’s collide, and a node cannot find its node_id in the traffic_map, the node backs off the repeats the LES stage on a random available timeslot in the RLY_ANNC ctrl_tbl bitmap. It is important to note that as RLY_ANNC advertises only available control timeslots, nodes trying to connect to a relay will only consider available timeslots thus, ND_REQ can collide with only those of other, unconnected nodes.

To ensure that nodes do not need to repeat the connection process, entries in the ctrl_tbl on both nodes and the relay include a time to live (TTL) of 5 cycles. When a node stops receiving RLY_ANNC, it decrements the TTL of the connection. Similarly the relay decrements the TTL of a connection, if it does not receive a ND_REQ within a cycle. Reception of these frames resets the TTL to 5.

c) Data Exchange: Referring to Figure 4, after a node, here Node A, receives RLY_ACK (Step 1) it looks up its data timeslot(s) shared in traffic_map (Step 2). Then, it forms Node Data (ND_DATA) frame and sends it in the assigned timeslot (Step 3). Another node follows the same pattern and sends its ND_DATA in (Step 4). ND_DATA contain destination address (dest_id), either node_id or group_id, the length of the data payload (data_len) and the data payload (data).

The relay collects all ND_DATA frames (Step 5), encapsulate them in a buffer (nd_data_buffer), and finally broadcast the RLY_TX frame (Step 6). Nodes receiving the RLY_TX pass onto the higher layer data if the encapsulated ND_DATA are addressed to their node_id, or a group_id subscribed to by the application layer (Step 7). Nodes and relay will repeat the DE stage de_cnt number of times (Steps 8-10).

C. Protocol Configuration

The BRP is quite flexible allowing Beartooth networks to support applications with different performance requirements. While the number of network parameters is quite large, we observe that the number of useful combinations is small. As a result, we let the relays to specify the set of parameters with a config_id, which then allows a node to look up a specific combination of parameters pre-loaded onto Beartooth nodes.

The configurable parameters include BRP parameters such as: the number of LES control timeslots, DE timeslot duration and count, and sleep time before a RLY_ANNC to reduce protocol duty cycle. The configurable parameters also include LoRa PHY settings such as spreading factor, channel bandwidth, coding rate, and whether or not the CRC is used.

V. EVALUATION

To demonstrate BRP flexibility of performance, we evaluate its performance in three network scenarios, while measuring latency, cycle duration, PDR, and throughput.

A. Setup

We configure the Beartooth radios to use SF 7, coding rate of 4/5, channel bandwidth of 250 KHz, and the use of the CRC. The achievable channel bandwidth in this configuration is 10.9 kbit/s and represents an attractive tradeoff between channel capacity and range [25].

Further, we configure three network scenarios as follows. Scenario 1 supports the exchange of short messages. The application generates 20 B messages every second with the goal of delivering them within 500 ms without relying on multiple DE stages. We vary the number of transmitting nodes between 1 and 3 while using 3 LES timeslots.

Scenario 2 supports real-time data streams. The goal of this scenario is to demonstrate BRP’s ability to accommodate voice streams within 500ms latency. We encode voice data with Codec 2 at 700 bit/s, which generates 20 B messages every 228 ms [18]. In the experiment we also vary the transmission interval to find the maximum flow throughput. We configure the number of DE stages to 3.

Finally, Scenario 3 supports BRP performance under the EU duty cycle regulations, which restrict nodes to transmit only 1% of the time [5]. The goal of this scenario is to demonstrate BRP’s ability to meet EU regulations and compare its performance against other LoRa MAC protocols in terms of PDR.
and control overhead. To do so, we restrict the frequency of RLY_ANNC messages by adding a sleep interval to the cycle.

B. Results

a) Scenario 1: Short messages: Our first experiment includes two devices: a relay and a sender/receiver. To measure the latency, we include the timestamp in the data packet and let node addresses the packets to itself. This approach allows us to measure the time difference between transmission (through the relay) and reception on the same node, without the need for synchronizing clocks between the sender and the receiver.

Figure 5 shows the CDF of cycle duration (blue solid line) and latency (orange dashed line) on the y-axis and time (ms) on the x-axis. We measure the latency at the sender between when the input data appears at the send buffer (from the application) and when it appears in the receive buffer (from the RLY_TX). We measure the cycle duration between the reception of RLY_ANNC and of RLY_TX.

We observe that latency ranges from 293 ms to 1.47 s with the mean of 478 ms. This result indicates that on average the delay of short messages remains under 500 ms. The variation in latency comes from the timing of receiving application-layer data in relation to the state in the protocol cycle. Data received just prior to the transmission of ND_DATA achieves the lower latency of less than a full cycle, while data received just after needs to wait for the next cycle for transmission. When a ND_REQ is lost due to collision, the node may need to wait for the start of the next cycle, thus increasing the measured latency producing the tail in the CDF.

We also see in Figure 5 that the cycle duration is fairly stable averaging around 362 ms. However, when a node is unable to get a control timeslot in the first cycle, and the second data packet is queued, relay may schedule two DE stages consecutively resulting in longer cycle of 553 ms, shown as the step in the cycle duration CDF.

We also wanted to investigate how different numbers of LES timeslots in RLY_ANNC effect nodes’ ability to schedule transmission, and so application layer latency. Our experiment consists of four devices, one relay and three nodes, where one to three nodes send periodic packets as in the previous experiment. In Figure 6 we show application latency on the y-axis and the number of LES timeslots on the x-axis. The different series in the plot show the number of active sender nodes.

With one and two active senders, we see an upward linear trend in latency with increasing number of LES timeslots. This increase is due to a longer cycle duration needed to accommodate the extra LES timeslots. However, with three sending nodes, we see latency decrease. This effect is due to the collisions of ND_REQ from simultaneous senders in randomly chosen control timeslots. Increasing the number of available LES timeslots decreases the probability of such collisions, thus the number of retries in subsequent cycles, and so message latency.

Thus, BRP cycle duration can be tuned by the number of LES timeslots to accommodate the number of simultaneous transmitters in the network. We note that BRP can accommodate a large number of nodes and the LES timeslot tuning applies merely to the number of simultaneous senders required by the application.

Figure 7 shows PDR, obtained in the previous experiment, on the y-axis and number of LES timeslots on the x-axis. We observe that PDR stays essentially the same regardless of the number of LES timeslots. This is because, number of LES timeslots has no effects on DE timeslots and when relay schedules three nodes within a cycle, nodes use all the available DE.
timeslots. We also observe that the PDR decreases slightly with additional simultaneous transmitters. Even though DE timeslots are strictly scheduled, a few milliseconds of shift can cause data frames to overflow into neighboring timeslot causing collisions on a very rare occasions. These rare collisions are due to the clock drift of the SX1276 chipset.

b) Scenario 2: Real-time flows: We want to demonstrate BRP can accommodate data streams without compromising latency by enabling consecutive DE stages. In this experiment we utilize two devices; one node and one relay, configured to schedule 3, 5 or 7 data exchange stages consecutively.

Figure 8 shows latency on the y-axis and the number of consecutive data exchanges on the x-axis. We observe that the mean latency is 305 ms when relay schedules 3 DE stages back to back. Latency decreases to 283 ms when number of consecutive data exchanges are increased to 5. Finally at 7 DE stages, latency hits 278 ms and shows marginally diminishing returns, because the duration of LES stage remains undiminished and represents an increasingly large proportion of cycle duration.

To observe the effects of number of active nodes in scenario 2 we set up another experiment with four devices – three nodes and one relay. In this experiment we focus on throughput observed on the relay and average PDR observed on nodes. Figure 9 shows network throughput (blue solid line) in kbit/s on left y-axis, PDR (orange dashed line) on the right y-axis and number of active senders on the x-axis. Experiments starts with 1 active node and the figure shows network throughput (and flow throughput) at just below 0.8 kbit/s. When another node is activated we see a linear upward trend where network throughput reaches just below 1.6 kbit/s with almost no drop in PDR, staying at 98%. The 3rd and the last node is activated, we still see an increase in network throughput reaching 2.25 kbit/s however, the rate of the increase is diminished due to PDR getting a hit. As discussed earlier, the drop in PDR is due to an occasional shift in ND_DATA transmissions and an overlap with neighboring DE timeslot resulting in ND_DATA collisions. Even under such collisions, the per flow throughput remains above 750 bit/s and allows the Beartooth network to deliver not one, but three simultaneous real-time voice flows encoded with Codec 2 [18].

Figure 10 displays network throughput from the previous experiment in kbit/s on the y-axis and number of data exchange patterns per cycle on the x-axis. We observe that as the number of DE stages increases, so does network throughput. This effect is due to the fact that the single LES stage is amortized by multiple DE stages.
c) Scenario 3: EU duty cycle: Finally, we want characterize the performance of BRP under EU duty cycle limitation and compare it against the performance of other LoRa MAC protocols. As mentioned earlier, the EU restricts duty cycle to 1% [5]. Doing so significantly reduces network throughput and increases latency. In the BRP the time-on-air (ToA) for relay transmissions at 27%. The relay ToA dominates node ToA, and so we add a sleep time of before each RLY_ANNC frame to bring relay (and node) ToA under 1% and into EU compliance.

Figure 11 shows the relationship between network throughput (blue solid line), maximum latency (orange dashed line) in seconds, and the number of consecutive DE stages with the appropriate cycle sleep time for each number of stages. The left y-axis marks network throughput for three simultaneous transmitters, while the x-axis shows the number of DE stages. While additional DE stages increase network throughput, the overall throughput remains constrained by the added sleep time. The right y-axis shows the maximum latency corresponding to the number of DE stages. We observe that the maximum latency increases with the number number of DE stages, because each DE stage requires an increase in sleep time to keep BRP duty cycle under 1%.

C. Comparing BRP to Other LoRa MAC Protocols

In comparing BRP performance to that of other LoRa protocols discussed in Section II, we observe that the combination of BRP and the frequency hopping Beartooth radio significantly outperforms existing approaches in terms of latency and throughput. The closest competitor to BRP is DQ-LoRa, which throughput gain of 0.7 kbit/s, but a latency of around 4 s [12]. BRP delivers similar flow throughput, but with mean latency under 500 ms.

PDR values are comparable across the protocols and do not benefit from Beartooth radio’s frequency hopping. We observe that BRP PDR of 0.98 is on par with that of TS-LoRa (0.9986) [2] and RT-LoRa (0.98) [4], and above that of ST/CA (0.87) [9].

Finally, we compare BRP’s control overhead with the control overhead from other LoRa protocols. We calculate BRP control overhead ratio by counting the cycle bytes that represent protocol control (all the fields except data) and dividing it by total payload data (data). The smaller the ratio, the more efficient the protocol. In our calculations, we assume BRP has three active senders and three DE stages scheduled. Referring back to Table I, in the LES stage, the control overhead comes from a RLY_ANNC, 3×ND_REQ, and a RLY_ACK, or

\[4 + 3 \times 6 + 15 = 37 \text{ B.}\]

In the DE stage, the control overhead comes from 6 B in ND_DATA and 20 B in RLY_TX. With one LES stage for every three DE stages the total control overhead is

\[37 + 3 \times (3 \times 6 + 20) = 151 \text{ B.}\]

Total payload data in the DE stage, on the other hand, includes 20 B in each of three ND_DATA frames and 60 B in the RLY_TX. Since there are three DE stages, the total payload over two hops is

\[3 \times (3 \times 20 + 60) = 360 \text{ B.}\]

The the BRP control overhead ratio in this scenario is 151/360 or 41.9%. Of course a different protocol configuration, in terms of ND_DATA payload size, or the number of DE stages would change this ratio.

Using the scenario presented in the paper of Zorbas et al. we calculate the control overhead of TS-LoRa to 52% [2]. We were unable to compute the control overhead ratio for the
other LoRa protocols listed in Section II due to insufficient information in their papers. Although not a direct comparison, we observe that BRP uses efficient signalling pair with at least one competing LoRa protocol.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented the Beartooth Relay Protocol – a novel MAC protocol for LoRa. BRP provides the flexibility to meet various application performance requirements, notably under 500 ms latency for short message exchanges. BRP also supports real-time streams, specifically that of multiple, simultaneous voice flows, under the same latency bounds. BRP does so by leveraging frequency hopping mechanisms of the Beartooth radio and by making long-lived transmission opportunity reservations. We also demonstrate BRP’s performance under EU duty cycle restrictions that are more stringent than FCC rules. The results indicate BRP’s suitability to a range of IoT applications beyond sensor data collection.

In the future we plan to extend BRP beyond two-hop paths of relayed communications. Extending a single LoRa channel to support multiple hops is challenging due its limited bandwidth. To circumvent that problem we plan on equipping Beartooth relay nodes with additional LoRa radios to enable inter-relay communications. We will link the orthogonal channels used by these radios with data forwarding through the controller, supported by Address Resolution Protocol (ARP) and switched forwarding.

The C++ implementation of BRP also allows us to move the protocols from Raspberry Pi onto the shield board micro controller. We are currently evaluating micro controller options to enable Beartooth radio operation on standalone hardware.

Finally, Semtech introduced a LoRa Frequency Hopping Spread Spectrum (LR-FHSS) extension to LoRa in December of 2020 [26]. The LR-FHSS mechanism implements frequency hopping transmissions at the physical layer without changes to the interface presented to the link layer. The LR-FHSS mechanism does use an additional 3 B in the header and is able to provide added robustness at a modest impact to latency and throughput. We expect that the LR-FHSS will allow BRP retain most of its performance benefits while deployed on a generic LoRa hardware as opposed to a Beartooth radio. To verify that, we will perform an evaluation of BRP on LR-FHSS chipsets and compare it against the results presented in this paper.

The combination of BRP running on standalone hardware (without a paired Raspberry Pi) and a generic radio using the LR-FHSS mechanism will make it easier and cheaper to deploy BRP a variety IoT deployment scenarios.

REFERENCES