ENABLING REAL-TIME APPLICATION STREAMS IN OFF-GRID REGIONS AND MISSION CRITICAL APPLICATIONS THROUGH BRP, RADIOS, AND GATEWAYS WITH COTS COMPONENTS

by

Batuhan Mekiker

A thesis proposal submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

May, 2023

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

The Internet of Things (IoT) applications require flexible and high-performance network solutions, but many IoT solutions can only support single-use case applications, which limits their performance and flexibility for real-time and streaming applications. LoRa offers a flexible physical layer but lacks the power needed in its link layer protocols to support real-time flows. The Beartooth Relay Protocol (BRP) expands the performance envelope of LoRa, making it suitable for a wide range of IoT applications, including those requiring real-time and streaming capabilities. However, the resource-limited nature of LoRa does not allow BRP to support self-healing mesh network capabilities or beyond two hops while maintaining real-time streams. To address the limitations of BRP in supporting mesh network capabilities and real-time streams beyond two hops, we move our focus to the development of the second-generation Beartooth Radios, MKII, and the first-generation Beartooth Gateways. We utilize Commercially-available Off Shelf Components (COTS) in the radios to provide a cost-effective, power-efficient, and compact solution for establishing real-time situational awareness. The self-healing mesh network provided with MKII and Gateways also enhances the reliability of the overall network, ensuring connectivity even in case of node failures. By incorporating military information brokers, such as the Tactical Assault Kit (TAK), the Beartooth Gateway establishes a hybrid network between Beartooth radios, gateways, and other TAK-capable devices, ensuring compatibility with existing IP networks. By combining Beartooth MKII radios, Gateways, and flexible link layer protocol elements in BRP, this research demonstrates a versatile and flexible solution that provides real-time application streams and critical situational awareness capabilities in mission-critical applications.

1

INTRODUCTION

Motivation and Overview

The need for reliable and performant communication is essential in today's world. However, many regions lack the necessary infrastructure due to factors such as difficult terrain, remote locations, or low population densities. These factors often make it economically unviable for service providers to establish connectivity infrastructure, resulting in a digital divide between connected and unconnected regions. Even in areas with connectivity solutions, mission-critical applications may face interruptions or failures because of natural disasters or man-made disruptions, further highlighting the need for robust communication networks.

In addition to providing essential services, reliable communication networks are crucial for disaster response and recovery, as well as enabling economic growth in remote areas. The lack of connectivity infrastructure hampers the development of healthcare, education, and other essential services, perpetuating a cycle of underdevelopment in disconnected regions [1]. As such, addressing the issue of inadequate connectivity is of vital importance for the overall improvement of living standards and socio-economic progress.

Real-time Data Dissemination across Multiple Networks

Peer-to-Peer Networks To address these challenges, peer-to-peer networks have emerged as a viable alternative to traditional communication solutions. These networks are self-organizing, decentralized, and require no infrastructure, making them ideal for use in remote or off-grid regions. By enabling direct communication between

devices, peer-to-peer networks offer increased resilience to failures and improved scalability compared to centralized systems. Furthermore, their decentralized nature makes them less vulnerable to targeted attacks or single points of failure.

Despite their advantages, peer-to-peer networks face certain challenges, such as limited bandwidth, security issues, and lack of standardization. These issues may hinder the widespread adoption and effectiveness of peer-to-peer networks, necessitating further research and development. Moreover, existing IoT technologies designed for sensor networks may not be suitable for real-time user traffic, which calls for new solutions specifically designed for this purpose.

LoRa MAC Layer and Beartooth Relay Protocol The LoRa MAC Layer, a flexible, low-power, and long-range solution, has emerged as a popular radio solution for off-grid communication. This technology is also affordable and easy to develop, making it accessible to a wide range of users. The LoRa MAC Layer's flexibility allows it to be used in various applications, from remote sensing to asset tracking, offering a versatile solution for off-grid communication needs.

However, the limited bandwidth of LoRa poses certain limitations, necessitating the development of new solutions like the Beartooth Relay Protocol (BRP). The BRP doubles the communication distance by utilizing relay nodes, allowing users to extend the range of their LoRa networks beyond the typical two-hop limit. While the BRP has shown promise, it is not without its limitations, and further research is needed to improve its performance and scalability. Developing new solutions that are specifically designed for off-grid regions will be essential in enabling reliable and performant communication in such areas.

Research Design

Problem Statement

The growing importance of real-time situational awareness (SA) in mission-critical applications, particularly for small military units, has been underlined by recent conflicts. Conventional real-time SA solutions tend to be costly, energy-intensive, and cumbersome. This highlights the need for the development of more affordable, energy-efficient, and user-friendly alternatives that works in regions that lacks infrastructure. The second-generation Beartooth Radios MKII, designed using commercially available off-the-shelf (COTS) components, strives to address these issues.

Design Science Framework and Problem Decomposition

The present thesis is a comprehensive exploration of Internet of Things (IoT) technologies. The research is divided into two main parts, each contributing to a deeper understanding of IoT's potential and limitations. The first part of the thesis critically evaluates various IoT technologies, identifying their respective use cases, and highlighting their shortcomings. This evaluation aims to provide a solid foundation for the development of new solutions that address the limitations of existing IoT technologies.

Following the critical evaluation of IoT technologies, the study introduces a novel link layer protocol that enhances connectivity performance and mitigates the identified shortcomings. This new protocol, specifically designed for off-grid regions, is expected to improve the reliability and performance of communication networks in these areas.

The second part of the thesis focuses on real-time Situational Awareness (SA) dissemination through the deployment of radios comprised of Commercial Off-The-Shelf (COTS) components. This section of the research investigates the integration of the aforementioned radios with existing mission-critical applications through

gateways. This analysis contributes to the literature by providing valuable insights into the development and deployment of Peer-to-Peer communication technologies in complex and high-stakes environments.

Overview In Chapter 3, the thesis discusses the wireless connectivity problem and how enabling Device-to-Device connectivity solutions would address the wireless coverage gap. This analysis aims to provide a better understanding of the challenges and potential solutions in the context of wireless communication, especially in remote and off-grid regions.

Chapter 4 tackles the limitations of existing solutions for situational awareness. The chapter also proposes a solution to couple resource-scarce networks with higher capacity networks, creating hybrid networks in which situational awareness data can disseminate seamlessly. This examination offers insights into the potential improvements in real-time SA data dissemination across various networks.

Finally, in Chapter 5, the thesis concludes and discusses future work. This chapter summarizes the key findings of the research and outlines potential avenues for further investigation and development, aiming to contribute to the ongoing evolution of IoT technologies and their applications in mission-critical situations.

5

BEARTOOTH RELAY PROTOCOL: SUPPORTING REAL-TIME APPLICATION STREAMS WITH DYNAMICALLY ALLOCATED DATA RESERVATIONS OVER LORA

Problem Statement

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 [2–7].

One of the recent and exciting physical layer protocols in the ISM band is Long Range (LoRa) from Semtech [8]. 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 [9]. However, link-layer protocols designed for LoRa do not fully take advantage of LoRa's flexibility and cater to specific applications in specific settings [3, 5, 6, 10–13]. 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 [14] 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 [15], 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 [5, 14]. To provide predictable delay in LoRa networks

several solutions proposed time-slotted medium access control (MAC) [3,4,10]. 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 [16]. A few approaches aim to sustain streaming data by abandoning PDR as a primary metric and focusing in improving LoRa throughput instead. Industrial LoRa [6] combines contention and contention-free transmissions to provide sustained throughput of 28 bit/s. DQ-LoRa [13] 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 [17]. 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 chapter 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 [18].

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 [19].

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.

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 [9]. 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 [20]. 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 [14, 21].

The LoRa Alliance publishes a carrier sensing multiple access (CSMA) LoRaWAN protocol, which allows devices to communicate via gateways [22]. LoRaWAN utilizes an ALOHA based approach where end devices transmit contending frames, which leads to error rates as high as 70% depending on the frame size [10]. 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 [23].

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 [10]. 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 [11]. 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 [3]. 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 [12]. 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 [6]. 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) [13]. 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 best-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 [5]. 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 chapter, 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.

Beartooth Radio

The Beartooth radio behind the BRP is a custom LoRa shield paired with a Raspberry Pi 4 controller, as shown in Figure 2.1. The shield includes a SX1276 LoRa chipset, which communicates with the BRP on the controller's CPU via the Pi's

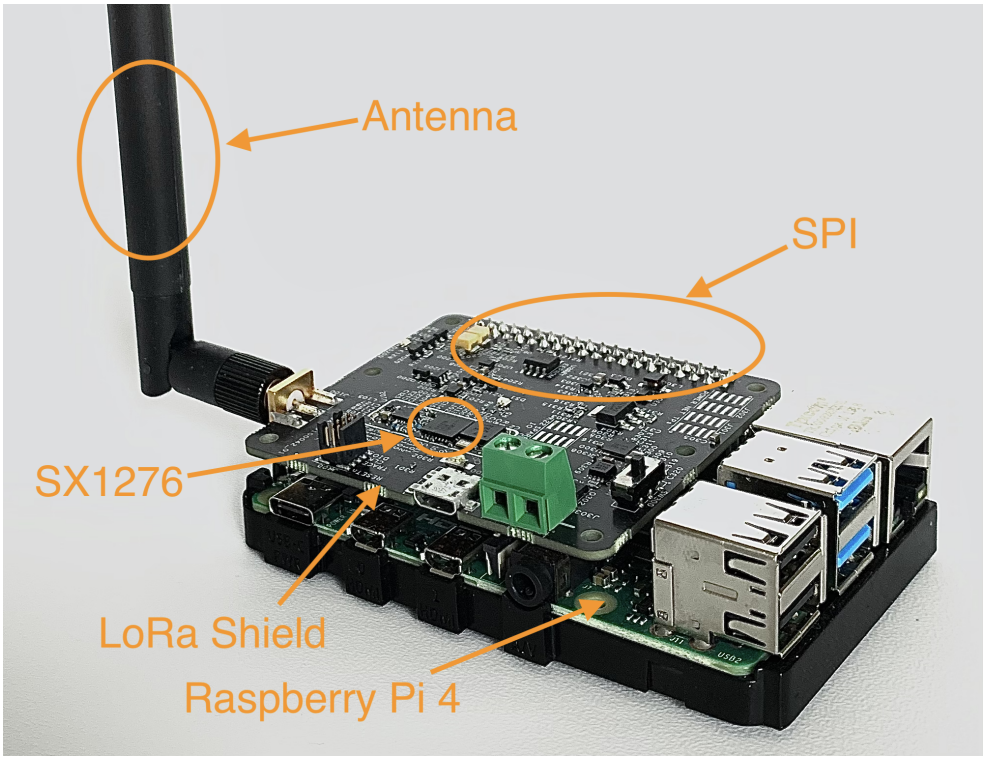


Figure 2.1: Beartooth radio LoRa shield paired with a Raspberry Pi 4.

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 the steepest slope providing the highest data rate, 37.5 kbit/s, and SF12 creating the flattest slope provides the greatest robustness and therefore range with 0.26 kbit/s [7]. 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 [24]. 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 [25]. This regulation limits the throughput LoRa devices can achieve on a single frequency. However, SX1276 chipset supports internally timed frequency hops [20]

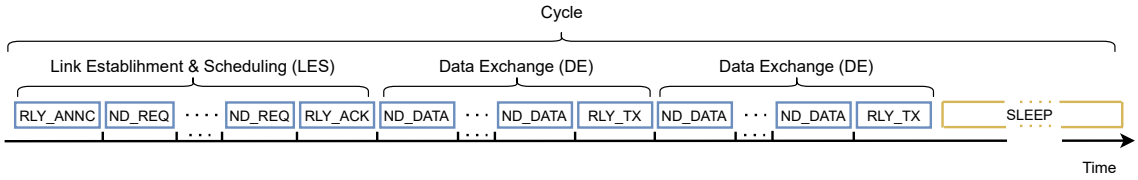


Figure 2.2: BRP cycle, stages, and frames.

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 [24].

Beartooth Relay Protocol (BRP)

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 [19]. 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% [6]. Messages in this network should achieve a PDR above 90% as in existing LoRa protocols [3, 5]. 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 2. 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 2.

Finally, the FCC limits transmit power to 30 dbm [25]. The power limits restrict the communication range of Beartooth radios to 15.2 km as reported in our preliminary work [26]. Our transmissions also comply with FCC' dwell time regulations, as described in Section 2 where a radio cannot occupy one channel more than 400 ms [25].

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 2.1: List of protocol frames with descriptions.

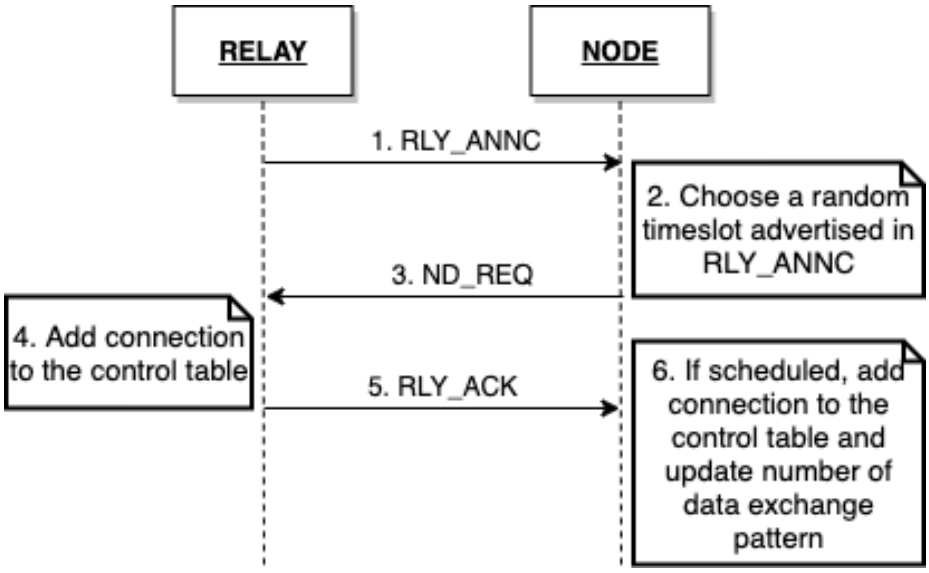


Figure 2.3: Link Establishment and Scheduling message sequence.

Protocol Operation

At a high level BRP operates by repeating a transmission cycle of protocol frames shown in Figure 2.2. We list the details of details of the BRP frames in Table 2.1. 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.

Link Establishment Referring to Figure 2.3, to start a cycle a relay broadcasts the Relay Announce (`RLY_ANNC`) frame (Step 1), which includes configuration ID (`config_id`) and a LES control timeslot table (`ctrl_tbl`) with available timeslots for

nodes to establish connections. The `config_id` specifies network parameters, discussed in Section 2.

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 2.3) 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`.

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 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 2. 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 connected on the LES timeslot it used to sent 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.

Data Exchange Referring to Figure 2.4, after a node, here Node A, receives RLY_ACK (Step 1) it looks up its data time slot(s) shared in traffic_map (Step 2). Then,

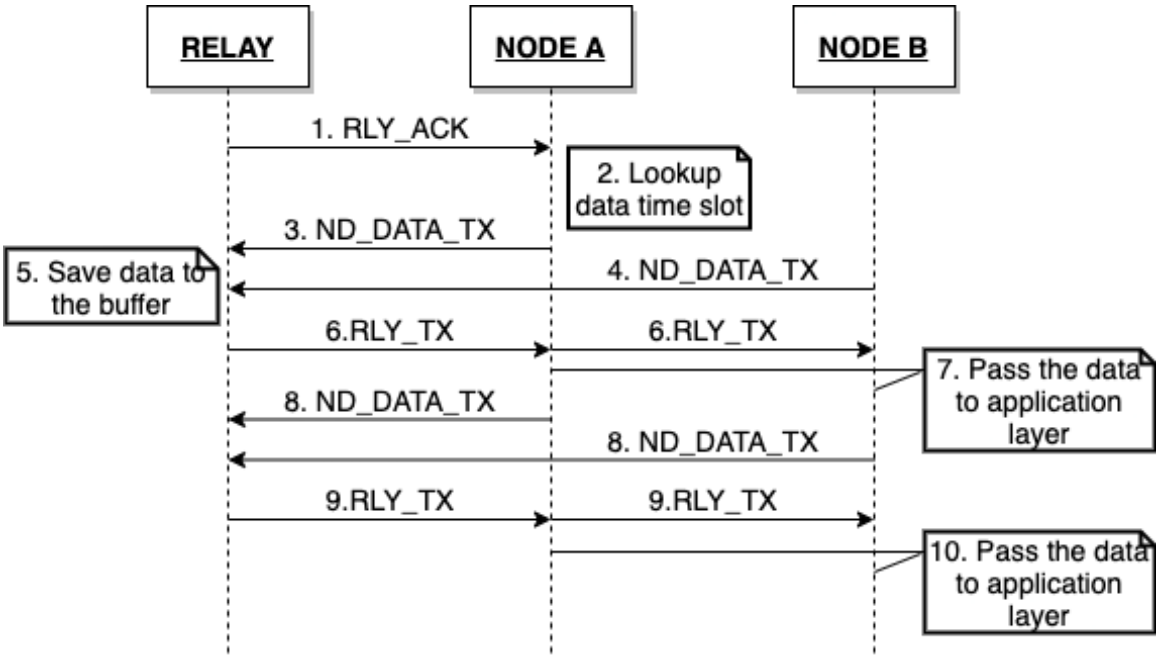


Figure 2.4: Data Exchange message sequence.

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).

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.

Evaluation

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

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 and attractive tradeoff between channel capacity and range [26].

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 500 ms latency. We

encode voice data with Codec 2 at 700 bit/s, which generates 20 B messages every 228 ms [19]. 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 [6]. 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.

Results

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 3.4 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 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

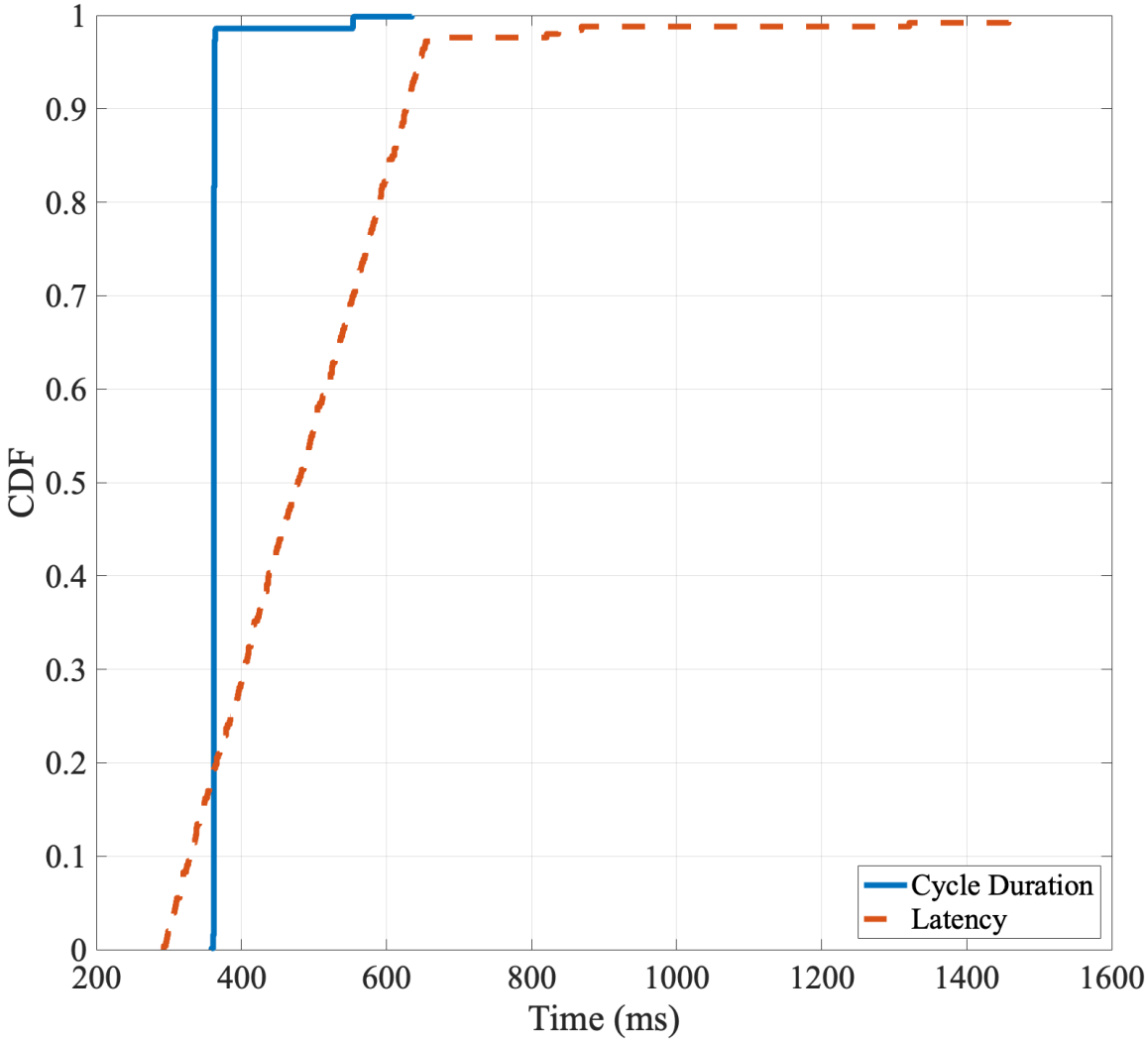


Figure 2.5: Two-hop latency and cycle duration.

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 3.4 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 2.6 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 2.7 shows PDR, obtained in the previous experiment, on the y-axis and

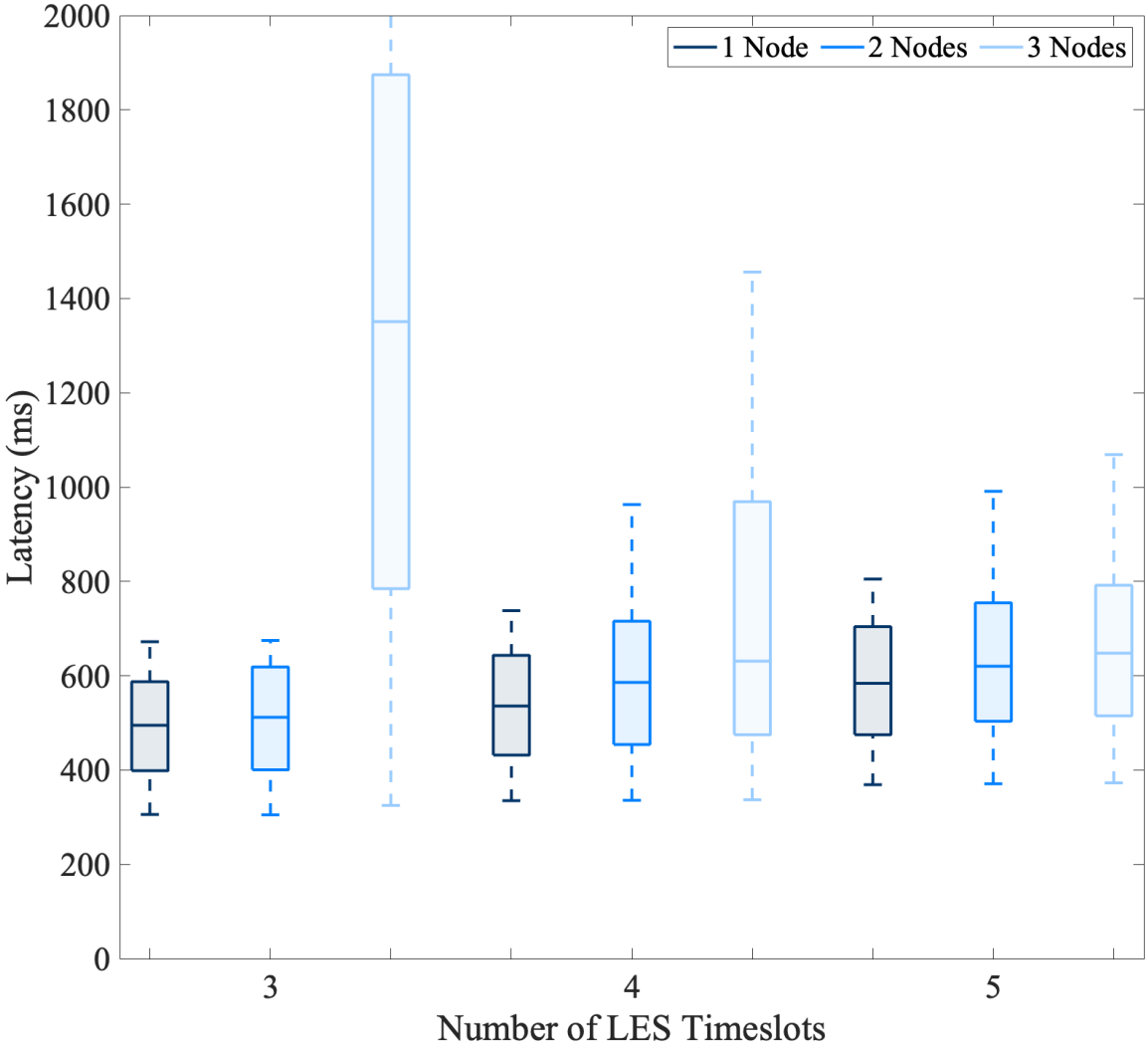


Figure 2.6: Latency versus number of LES timeslots and active senders.

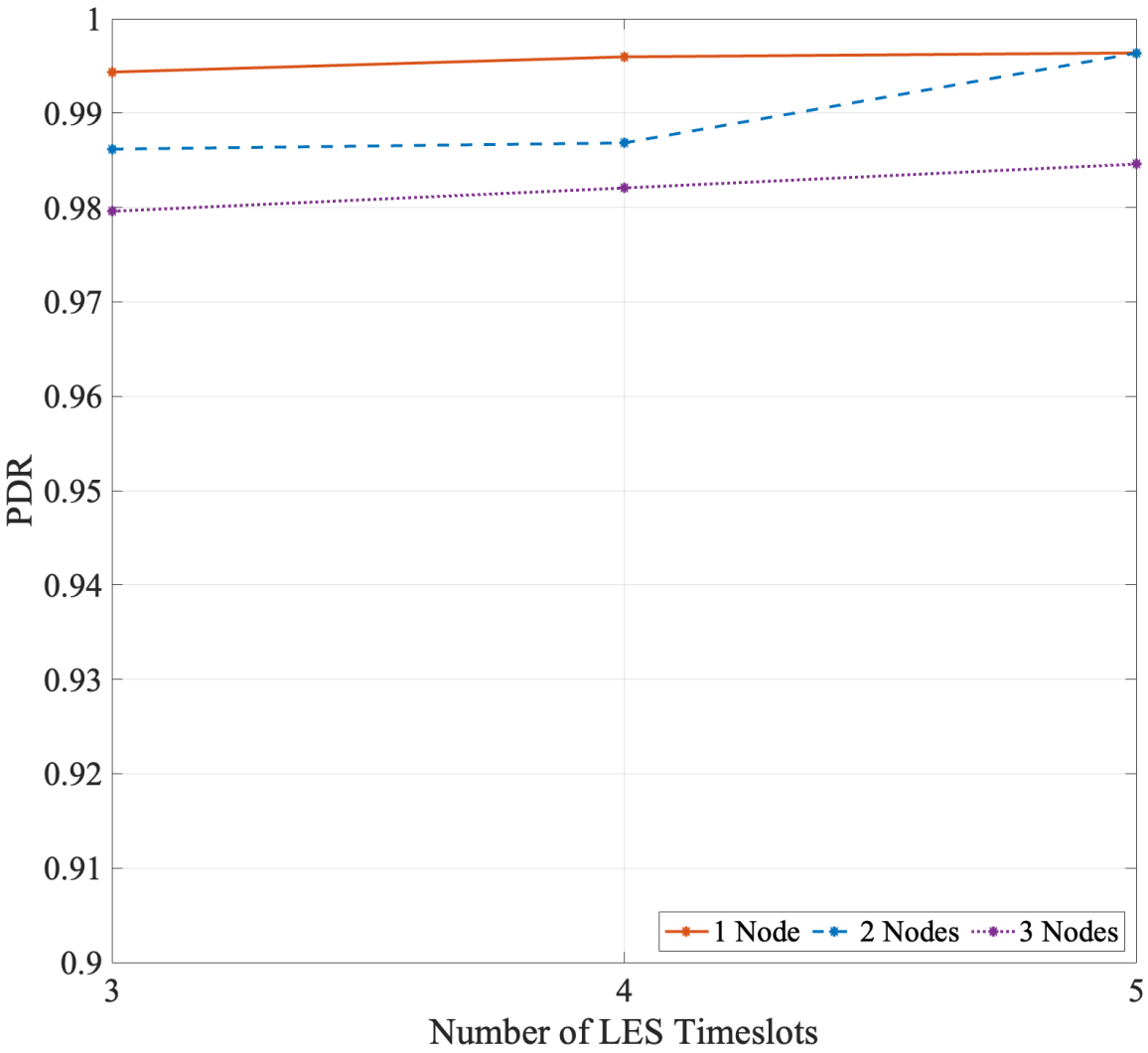


Figure 2.7: PDR versus Number of Control Timeslots.

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.

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 2.8 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 2.9 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

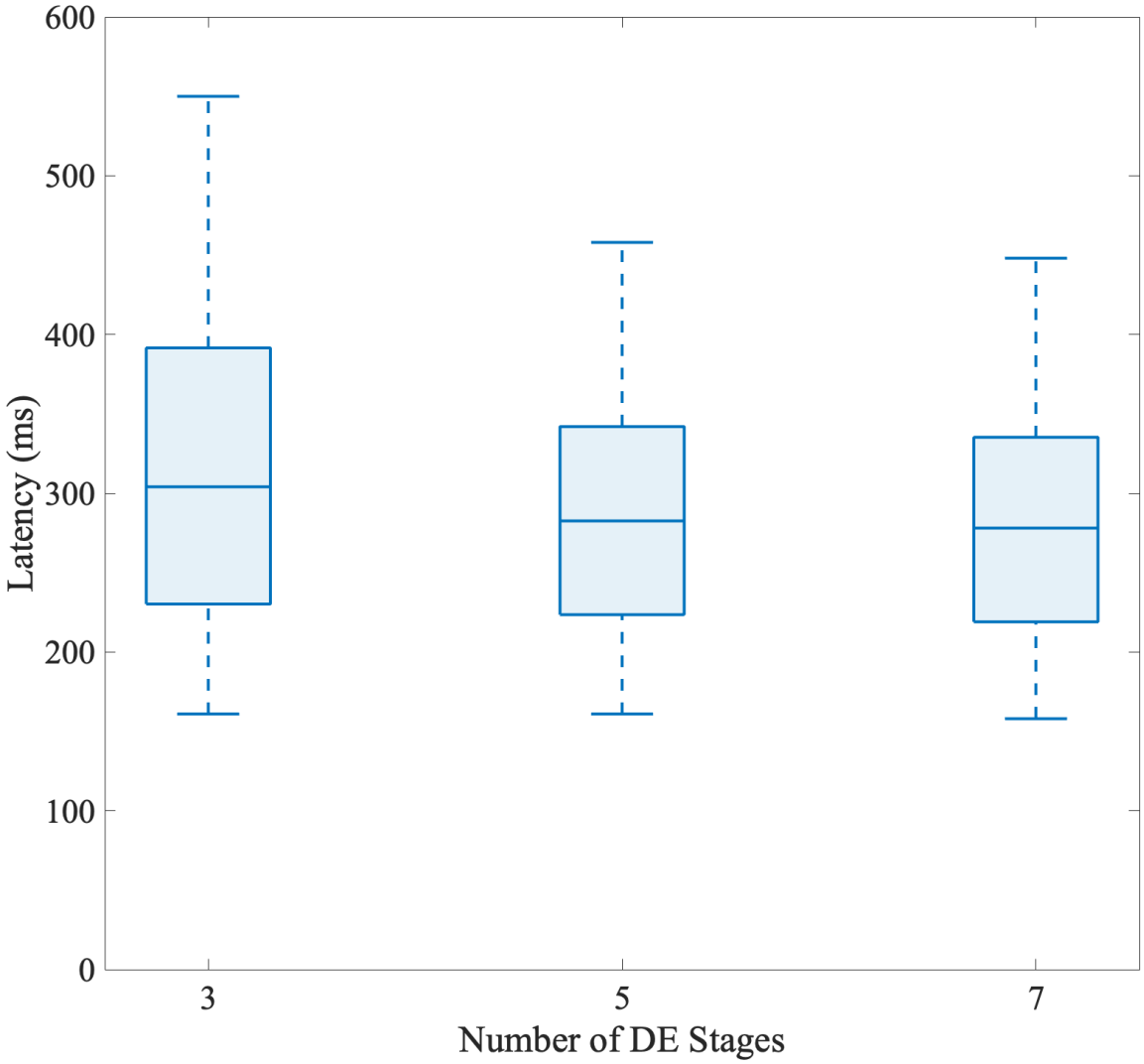


Figure 2.8: Latency versus Number of DE stages.

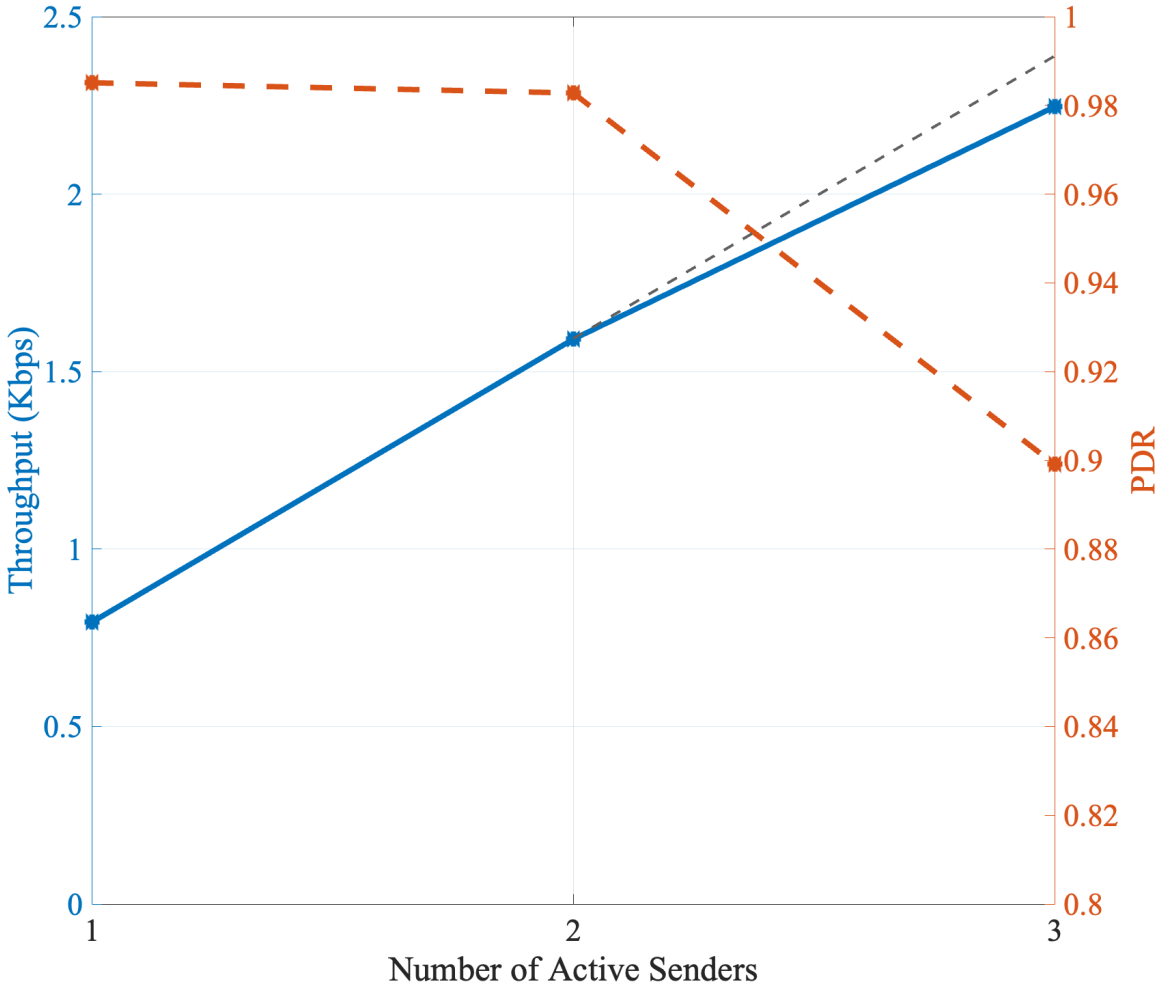


Figure 2.9: Throughput versus Number of Active Nodes.

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 [19].

Figure 2.10 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.

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% [6]. 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 2.11 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,

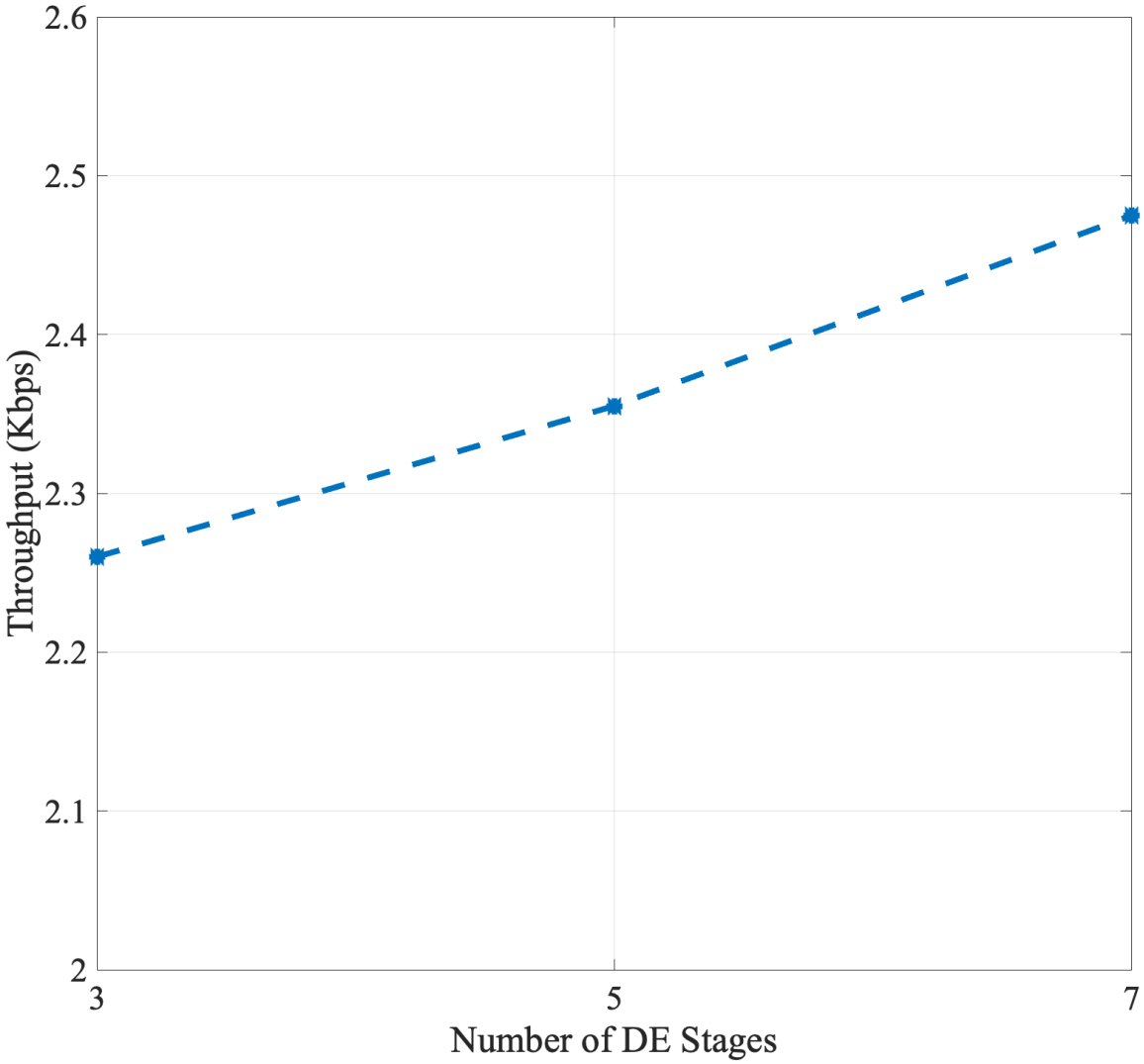


Figure 2.10: Throughput in kbit/s vs. DE stages.

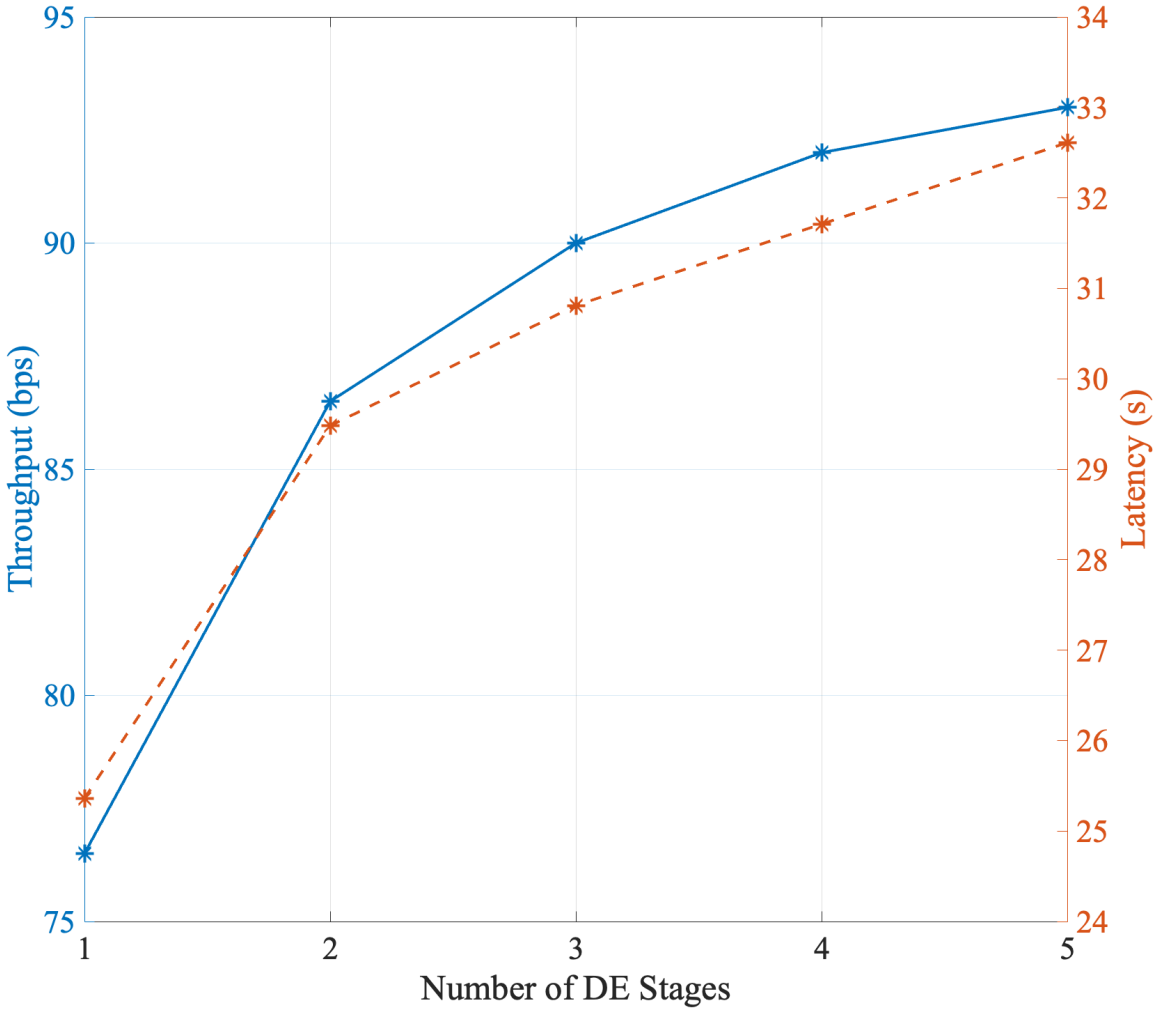


Figure 2.11: Analytical results of throughput and maximum latency observed under varying number of DE within a cycle.

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%.

Comparing BRP to Other LoRa MAC Protocols

In comparing BRP performance to that of other LoRa protocols discussed in Section 3, 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 [13]. 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) [3] and RT-LoRa (0.98) [5], and above that of ST/CA (0.87) [10].

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 2.1, 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% [3]. We were unable to compute the control overhead ratio for the other LoRa protocols listed in Section 3 due to insufficient information in their papers. Although not a direct comparison, we observe that BRP uses efficient signalling par with at least one competing LoRa protocol.

Conclusions and Future Work

In this chapter 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 hop 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.

Additionally, we are working on extensions to connect BRP-like ad-hoc networks to the internet through gateways, bridging P2P connectivity with centralized infrastructure and creating hybrid networks.

The C++ implementation of BRP also allows us to move the protocols from Raspberry Pi onto the shield board micro controller. We evaluated 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 [27]. 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.

COST-EFFECTIVE SITUATIONAL AWARENESS THROUGH COTS BEARTOOTH RADIOS AND GATEWAYS

Introduction

As we have seen with recent conflicts, the nature of warfare is evolving. A lack of encrypted communications creates an operational disadvantage for large military forces and allows small, highly mobile units to disrupt their advance [28]. On the other hand, accurate, up-to-date situational awareness (SA) allows these small units and their command centers to coordinate defensive operations and launch effective counterattacks. Consequently, a low-cost, scalable SA solution becomes an increasingly important force multiplier that military commanders desire to have in their arsenal.

Battlefield SA relies on two integrated technologies: a connectivity layer that provides raw communication links between agents, and an information layer that disseminates location-based data and operational directives. Battlefield radios from Silvus [29], Trellis [30], and WaveRelay [31] can establish direct as well as multi-hop connectivity among agents. These radios also integrate with mobile phones running the Android Tactical Assault Kit (ATAK) [32] to create an SA overlay. Networks built on these radios, however, have several shortcomings. First, these purpose-built radios are expensive [33], which limits their availability. Second, they are power hungry which increases their weight, limits portability, and increases observability [34]. Finally, third, they lack direct integration with the Tactical Assualt Kit (TAK) [35]SA broker, which limits SA to a squad, but does not encompass multiple squads and command centers.

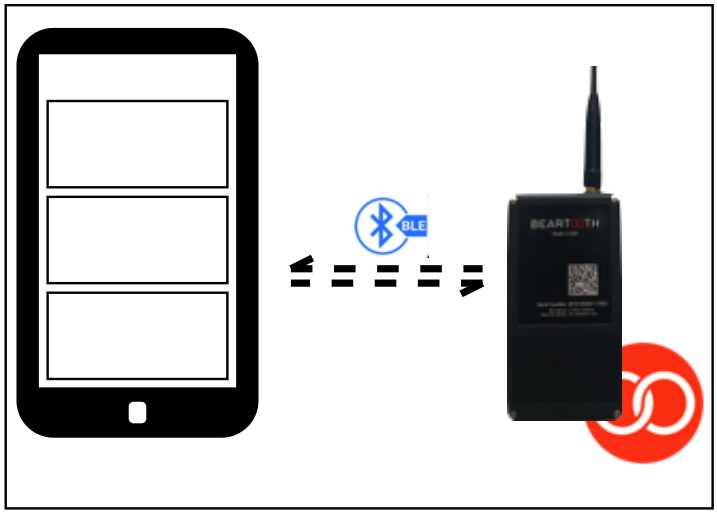
One way to reduce SA system costs and form factor is to build it on commercially available off-the-shelf (COTS) components. Research on internet-of-things (IoT)

communications has produced several multihop connectivity solutions [2, 36–38]. These low-cost radios provide low-bitrate links, which may nevertheless be suitable for SA applications. A key advantage of these radios is their low cost and spectral efficiency, which leads to low power requirements and small, lightweight form factors.

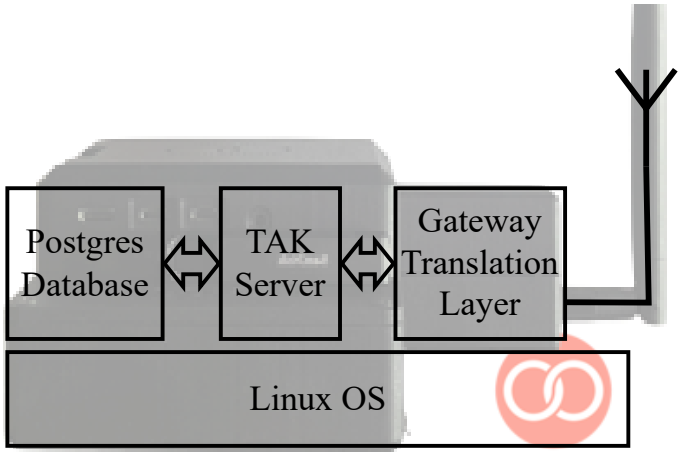
In this chapter, we propose a SA solution based on COTS radios. Specifically, we design the Beartooth MKII radios based on the XBee radio platform [39]. The XBee radio platform provides: more than ten miles of range in urban areas and sixty miles in rural areas with a line-of-sight between transceivers. XBee also supports DigiMesh, a self-healing link layer protocol with built-in frequency hopping and routing among more than a hundred nodes in a Low Power Wide Area Network (LPWAN) [40]. In addition, XBee radios are highly power-efficient and draw less than 3 mA while in standby mode, and 900 mA during data transmission at 30 dBm transmit power [41]. Finally, the XBee platform is inexpensive with modules easily accessible commercially.

The Beartooth MKII radio couples an XBee radio with a Bluetooth module that allows it to speak to a mobile phone running ATAK. To address the bitrate limitation of the XBee platform, we design the Beartooth ATAK Plugin to rearrange the existing TAK data format known as Cursor-on-Target (CoT) events into a more compact representation. We compress larger data types such as sensor readings, images, and bulk data and we split them for transmission over multiple XBee frames. We also leverage unicasting to reduce time-on-air for larger CoT events, which radios route frames to directly to the destination to reduce the number of data and acknowledgement packets traversing on the network with respect to broadcast frames. While Beartooth MKII radios exchange our compact frames, the original CoT events still get recreated at the receiver then published to ATAK, allowing us to share SA in a resource-constrained COTS network. The resulting MKII form factor, shown in Figure 3.1a, is 4.09 x 1.21 x 0.7 inches and weighs in at 6 oz, including a battery. The

Beartooth MKII radios are as much as 54% lighter than existing solutions discussed in Section 3.



(a) Beartooth ATAK Plugin and MKII radio.



(b) Beartooth Gateway and its software modules.

Figure 3.1: Bearthooth system design.

Further, we provide shared SA between squads and commanders through the Beartooth Gateway. The Gateway is a handheld server as seen on Figure 3.1b that consist of a MKII radio and a translation layer to couple an XBee network to an

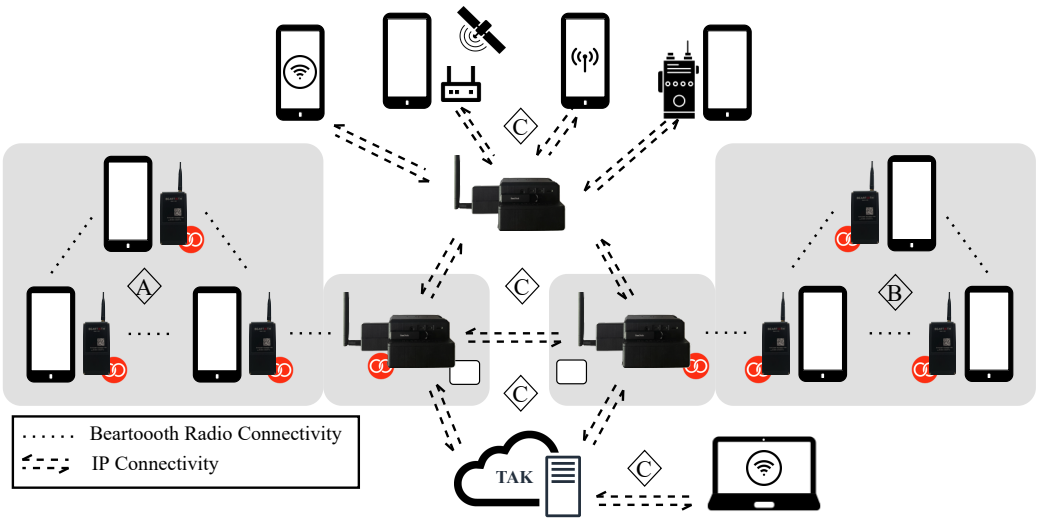


Figure 3.2: An example deployment of Beartooth network with MKII radios and Gateways, connecting many users spanning across WiFi, cellular (5G/LTE), satellite connectivity and battlefield radios.

IP network over cellular, satellite, or other radio technologies, such as the radios from Silvus, Trellis, WaveRelay, or Harris. IP connectivity allows the sharing of encrypted SA data among multiple squads as well as command centers running a TAK server [35]. Figure 3.2 illustrates a possible deployment scenario interconnecting multiple network technologies and operational areas into a shared TAK SA overlay. In Figure 3.2 we show two Beartooth networks, *A* and *B*, bridged together using two Gateways. Both Gateways are connected through an IP network such as the internet with the help of a Virtual Private Network (VPN) denoted as network *C*. An additional Gateway within network *C* serves the TAK information layer to multiple IP-connected devices through various network interfaces such as WiFi, cellular (5G/LTE), and satellite connectivity. All three TAK servers within Gateways are federated, allowing Beartooth SA data to flow into the IP network and SA data generated at network *C* to flow into Beartooth Networks *A* and *B*. A fourth TAK server sits in the cloud computing architecture, is also part of the network *C*, and

is federated to Gateway *A* and *B*. Therefore both Beartooth network members and members of IP network can exchange SA and battlefield intelligence. Furthermore, commanding officers can observe the whole operation and even send orders through the TAK server in the cloud using a WinTAK [42] device, creating an end-to-end encrypted communication links between forces in the field and an operation command center.

This chapter offers the following contributions:

1. A secure, bandwidth-efficient transport layer for TAK messages over XBee frames

2. A routing layer for TAK messages among Beartooth Gateways and TAK servers

A Beartooth network composed of MKII radios and Gateways delivers a capable SA solution to coordinate squads and command centers over large areas of operation. The low cost and small form factor of the Beartooth radios make it possible to deploy SA at a large scale and is a powerful force multiplier.

Related Work

To illustrate the need for cheaper, smaller, and more flexibly deployable radios, we discuss the limitations of existing solutions to create connectivity and situational awareness. We consider commercially available tactical radios as well as COTS technologies from the IoT space.

In the commercial space, there are three predominant solutions Streamcaster, TSM Shadow, and MPU5 radios from Silvus Technologies, TrellisWare Technologies, and Persistent Systems respectively. All were developed to provide tactical connectivity and SA for first responders and military forces. Streamcaster product line from Silvus Technologies uses a proprietary radio module capable of self-forming and

self-healing a mesh IP network [29]. Streamcaster Lite radio has a maximum of 1 W transmit power and 20 Mbps data rate. The radio requires 4.8 W to 17 W power while transmitting at 1 W (30 dBm), which is a few times more than the Beartooth MKII radio. The Streamcaster Lite weighs in at 10.4 oz and takes up more than twice the physical space of a Beartooth MKII.

TrellisWare Technologies is another solution provider for infrastructure-less tactical communication. Similar to Streamcaster, TrellisWare's TSM Shadow handheld radio encloses a proprietary radio module [30]. It has a transmit power of up to 2 W with a maximum data rate of 16 Mbps. Although there is no information about power consumption, it would be safe to assume TSM Shadow would consume similar power as Streamcaster Lite. It weighs around 11.3 oz, an ounce more than Streamcaster Lite however, volume-wise, it sits between the Beartooth MKII and the Streamcaster Lite.

Lastly, we discuss Persistent Systems' WaveRelay MPU5 radios. Compared to Streamcaster Lite and TSM Shadow, MPU5 is a more complete and capable radio system with a full-fledged mobile CPU and Android Operating System on board [31]. Without the battery module, It weighs around 13 oz. MPU5 has a wide variety of radio modules for many frequency bands with data rates up to 150 Mbps. With varying radio modules for different frequency bands, its transmit power varies between 4 W to 10 W, and its power consumption varies between 30 W and 50 W.

All three radios have IP network capability and provide sufficient bandwidth to support hundreds of devices sharing real-time CoT events. However, their physical footprint and cost are significantly higher than Beartooth MKII radios, limiting their adoption. Although all three support interoperability with ATAK and WinTak, they currently do not provide a product with direct integration to an edge TAK server. Therefore, they are physically limited to radios' coverage directly correlated with the

environment and terrain, causing troops in the field to exchange only local SA data and not to and from external sources.

A notable approach adopting COTS radios to provide connectivity and SA cost-effectively through ATAK is the HAMMER ATAK plug-in [43]. Soule et al. discuss the unpredictable availability of IP infrastructure and highlights the abundance of tactical handheld radios among the first responders and military forces [43]. They implement an acoustic modem to encode CoT events into analog signals to exchange SA through rather basic analog radios. Re-utilizing readily available, standard analog radios can provide an alternative solution for search and rescue missions and be vital in emergencies. However, the solution lacks the Medium Access Control (MAC) layer and Quality of Service mechanism, leaving the HAMMER plug-in ineffective for mission-critical as well as military applications where Quality of Service and real-time data delivery are expected.

Another approach to bring cost-effective COTS devices to SA systems is to adopt radio modules developed for a wide variety of IoT applications. Most IoT solutions provide comparatively low-cost and power-efficient solutions. Z-Wave [36], SigFox [37], DASH-7 [2], LoRa [36], and ZigBee [38] are the few solutions the IoT industry commonly utilizes. Z-Wave and SigFox can carry occasional data transmission at a low data rate over long distances. However, they do not meet the Quality of Service SA and TAK traffic requires, disqualifying their use in mobile SA systems. DASH7 and LoRa provide more extensive coverage and sufficient maximum raw data rate of 200 Kbps and 37.5 Kbps respectively for single-hop networks while keeping power consumption low. However, no stock or third-party link layer protocol implements a multi-hop self-forming, self-healing mesh network with throughput SA systems require. For LoRa, in earlier work, we explored a multihop link layer protocol where we show building a mesh network over LoRa is significantly difficult with

available network resources and concluded that we need a second radio to extend coverage beyond two hops [44]. ZigBee's presence in both sub-Ghz and 2.4 Ghz bands and interoperability with other ZigBee compatible devices increases its use cases for IoT applications; however, network deployment requires more effort and planning due to having different radio roles within the network. Thus, extending the mesh network is not as flexible and straightforward as XBee's DigiMesh.

Similar to ZigBee, the DigiMesh protocol developed for XBee radios support both sub-Ghz and 2.4 Ghz unlicensed bands. Both protocols work in similar mechanisms, have similar supported bandwidth, data rate (250 Kbps at 2.4 GHz, 150 Kbps at 900 MHz), number of devices supported within the network, and power consumption [38]. But in contrast to ZigBee, the DigiMesh protocol have only one network role within the protocol compared to ZigBee's slightly more complex deployment with a coordinator, router, and end-device roles. This makes DigiMesh easier to deploy, maintain, and flexibly extend, and it is why we choose to utilize the XBee platform and DigiMesh protocol over other COTS IoT radios.

Beartooth Network

To understand the effectiveness of the Beartooth network, we discuss the software components that both MKII and Gateway require for effective communication. We then elaborate on the types of situational awareness data that the MKII radios and Gateway can serve to the Beartooth Network and the IP-layer network.

Network Elements

The MKIIs are highly mobile and power-efficient handheld radios that forces in the field use to communicate within a local region. Gateways within Beartooth networks are edge servers bringing TAK capability to the field and integrating troops

using MKII radios to a larger TAK SA overlay.

MKII To limit the form factor of MKIIs we design them with limited computational power and place much of the transport layer logic in the Beartooth ATAK plugin. There are a few reasons we opt to use this design decision. First, designing and building a radio without a powerful processing unit is cheaper and less complex. Second, having a power-hungry processing unit affects the power consumption and thus overall battery life of MKII radios. Last but not least, since the mobile phone does all the processing, the protocol design is quite flexible and early upgradable. If and when we decide to change how we encode data into packets, we can do so by modifying the plugin without the necessity update the MKII radio firmware.

One of the key design decision is how to serialize and move data around in the most efficient way within a limited bandwidth DigiMesh network. We use Google's Protocol Buffers (ProtoBuf) library [45] for message serialization. ProtoBuf library lets developers take advantage of efficiently serializing any data structure in a language and platform-neutral manner. ProtoBuf works with schemas – simple data structures that can encode multiple data fields and types. ProtoBuf comes equipped with a few advantages. Enforcing data format among various applications, fine-tuning fields for different application-specific cases, and the flexibility to update schema easily are to name a few..

We define all packet within a ProtoBuf schema where we defined fields such as `sourceUid`, `destinationUid`, `messageType`, `textPayload`, `messageUid` and `locPayload`. While designing the Beartooth packet schema, we were aware of limited network resources. Therefore, we omitted unnecessary or redundant fields in the CoT event message formats, allowing us to downsize our packet length further. Once we have the proper packet format, ProtoBuf encodes necessary data into a series of bytes,

then passes those bytes to the Beartooth radio interface.

The Beartooth MKII plugin for ATAK is a lightweight platform that connects to Beartooth MKII radio using Bluetooth Low Energy (BLE) to send various SA data such as text, location, markers, shapes, pictures, and voice messages. Together with the MKII radio, we design the plugin to be a robust and secure communications platform. As the default option, it auto shares the user location with the entire network and a team leader to monitor the squad members. So that the team leader can take appropriate actions. In addition to the types mentioned above of SA messages, the plugin is also capable of sending casualty evacuation, navigation routes for walking, flying, and driving, icons for a wide variety of first responder missions, elevation, range and bearing. Once the plugin is connected to MKII through the BLE, the plugin configures the radio in the background to user preferences, where the user can set channel and encryption settings.

Contrary to other radios, MKII radios coupled with the Beartooth ATAK plugin do not directly transmit CoT events due to limited network resources. When the user triggers a SA event, such as a marker on the map user interface, the plugin puts necessary data to a Beartooth SA frame using the ProtoBuf schema, serializes it, and sends it to the BLE connection. Then, BLE queues the data for the radio in MKII to be transmitted. End-user devices receiving the Beartooth SA frame relay the data to the plugin. Using the ATAK API, the plugin only then creates CoT events at the receiver device to be shown on the ATAK user interface.

We also want to give the ability to end-user to plan their Beartooth network deployment. Therefore, we implement a network scanning tool where the MKII radios ping all the available devices within a local DigiMesh network. Then, based on their location draw a map on the ATAK app representing the RSSI values, distances, hops between devices, and overall network health. After deployment, the network

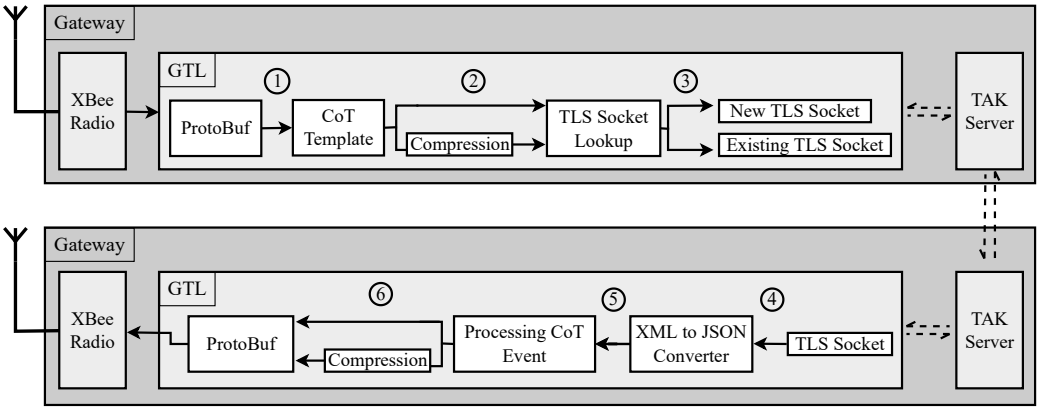


Figure 3.3: Data progression within Gateway and GTL showing various steps and software modules.

scanning tool can also help monitor the overall network, allowing users to devise a better strategy to deploy radios and achieve better, stronger communication links.

Gateway The Gateway, shown in Figure 3.1b, consists of three individual modules: Virtual Private Network (VPN), TAK Server, and Gateway Translation Layer. We use ZeroTier VPN [46] to configure a static IP address for each Gateway, but the VPN also serves as another security layer where we add 256-bit end-to-end encryption on top of TLS. ZeroTier VPN is lightweight, easy to set up, and provides P2P links between TAK server within the Gateway and IP connected end-users. Furthermore, we use TAK server version 4.0 optimized for Single Board Computers (SBC) such as Raspberry Pi. It allows us to bring the TAK server to the edge of the network and connect multiple local networks through the TAK's federation functionality. Federation enables TAK servers to exchange all or selected SA data to be passed on to other authenticated TAK servers securely.

Gateway Translation Layer

Gateway Translation Layer (GTL) is the novel solution that binds Beartooth MKII devices to TAK Server and consequently to IP network. Like the MKII radio, a Gateway has an XBee radio attached to its serial port, allowing SA data to reach the GTL, or GTL to serve incoming CoT events from TAK. There are two data paths from Beartooth to the IP network and from IP to the Beartooth network. Figure 3.3 illustrates data path within the Gateway for bi-directional communication. In the next two section we investigate how GTL processes incoming and outgoing data for both directions.

Data Flow: Beartooth to IP-network We show the data flow from a Beartooth MKII devices to an IP-network through a Gateway in Figure 3.3. The GTL, using the pre-determined ProtoBuf schema, parses the incoming SA data frame and determines the CoT event type. Depending on the type, the GTL process SA data within the DigiMesh frame as shown in *step 1* and forms a valid CoT event out of one of the pre-configured CoT event formats representing the original SA data frames. The pre-configured CoT events are templates that help us generate valid CoT events by pluging in relevant Beartooth SA data.

After forming a valid CoT event, depending on the Cot Event type the GTL can route fully formed CoT event to the compression engine in *step 2* to reduce message time-on-air. Only medium CoT events and CoT events carrying bulk data such as shapes, routes, pictures etc. go through the compression engine. In *step 3* the GTL looks up for active TLS sockets based on sender address. If there is an established TLS socket, the GTL recycles that active TLS socket to submit CoT event to TAK. Otherwise the GTL creates a TLS socket from available client certificates stored at the Gateway. The client certificates are generated and stored for the Gateway in the

configuration process, where we generate many certificates to be used and recycled for TAK server connections. Having multiple certificates helps us keep latency low and TLS connections as sticky as possible for ongoing or more active users rather than closing and opening TLS links for each SA data frame. Furthermore, having multiple certificates readily available will help us bring multi-thread processing to GTL in the future that enables parallel processing of DigiMesh frames and CoT events in queues It also would increase Gateway's capacity of supported devices. Once GTL establishes a TLS connection with the TAK server, it submits the generated CoT event and stores the connection for a brief period. Depending on the type and destination of the event, TAK Server triggers a set of procedures determining how CoT events get processed and routed.

Data Flow: IP to Beartooth Network Conversely, as seen on bottom section of Figure 3.3, when the data path is from the IP network to Beartooth Network, TAK Server on the top creates a TLS connection to TAK server on the bottom using TAK's federation functionality and submits relevant CoT events. Then TAK relays them to GTL. In *step 4* the GTL converts XML-based CoT event into a JSON object for ease of accessibility. GTL then processes the JSON CoT events by categorizing them by the event type on *step 5*. As we discussed, if the CoT event type is what we classify as medium or large, the SA data goes through the compression. Similarly, GTL forms the Beartooth SA data frame from either compressed and uncompressed SA data using ProtoBuf schema in *step 6*. Then ProtoBuf serializes the data into bytes and passes them to the radio module using the serial port. The radio module then transmits data either by unicasting or broadcasting depending on the data type.

Supported Data Types

The Gateway supports two-way communication between the Beartooth network and IP network. Therefore the Gateway support data flows from IP-network to the Beartooth network, the Beartooth network to IP-network, or the Beartooth network to the Beartooth network with Gateways in between. The Gateway also supports various CoT event types through the GTL. The list below shows currently data types Beartooth network and the Gateway supports.

1. *Small Packets (Up to 256 bytes (B)):* are packets formed to carry three distinct types of data; text, geo-location and acknowledgement. Text messages are type of CoT events that TAK compatible devices use to communicate over TAK overlay. TAK overlay also supports acknowledgment CoT events for text messages to show end-user's delivery receipts. Geo-location messages are CoT events containing latitude, longitude and altitude data which end-users use them to ping their location.

2. *Medium Packets (256 B to few KB):* are packets formed to carry what we call generic CoT events. It includes types such as markers, routes, and shapes. DigiMesh protocol limits the payload size to 256 bytes within a frame. For CoT events larger than 256 bytes we form multiple frames with payload size of maximum 256 bytes and transmit SA data in series of frames. To minimize spectrum usage, we utilize a compression engine to deflate XML-based CoT events so that number of frames is as minimal as possible.

3. *Bulk Data Packets (up to 25 KB):* are packets formed to carry bulk data that are in .zip file format. They include a XML-based manifest in the compressed file describing TAK what type of bulk data the compressed file encloses and how TAK needs to process it. So far Beartooth network can only support bulk

data that is an image file. But in the realm of TAK, bulk data can represent not only images but map overlays, set of icons for markers, or configuration for TAK server with certificates and more. We use the same mechanism we implement for Medium Packets where we send data in series of frames.

Designing a User-Friendly Gateway Admin Panel

In the development process of Gateway, our primary objective was to maintain simplicity, prioritizing ease of deployment and usability. We recognized that this technology would be implemented in highly complex and challenging environments, necessitating a user-friendly design. To address this need, we incorporated a Gateway Admin Panel, which presents vital information, such as the Gateway serial number, software version, XBee Network settings, VPN settings, and associated IP addresses. Furthermore, we ensured secure access to the requisite certificates for TAK server connectivity within the Gateway, all protected by robust authentication measures.

As we continue to refine and enhance the functionality of the Gateway Admin Panel, our future development plans include the capacity to modify the aforementioned settings directly from the panel itself. For now, these settings are configured either through our Beartooth Plugin or manual intervention. Additionally, we intend to incorporate features such as tracking online Beartooth MKII devices within the network via location updates displayed on a map, and enabling team communication through text messaging capabilities directly from the Gateway Admin Panel. This will provide users with a comprehensive, efficient, and streamlined management experience for their Gateway system.

Evaluation

Beartooth devices utilize the DigiMesh protocol from the XBee platform. As a result, they inherit its performance. A detailed performance evaluation of DigiMesh is beyond the scope of this chapter (see [38] for an overview).

To demonstrate Beartooth MKII radios and Gateways' capabilities in SA use case, we evaluated their performance on supported data types with latency measurements. In our experiments, we use one MKII radio, a phone with Beartooth ATAK plugin connected to MKII radio, one IP-connected phone with ATAK, and one Gateway. We measure latency with the timestamp created at the start of the transmission till it is received by the IP-connected ATAK phone. The measurement reflects single-hop latency for varying data types.

Figure 3.4 shows latency in seconds on the y-axis and supported data types on the x-axis. We observe the latency for small packets such as text and location data on Figure 3.4a. The average latency hovers around 0.73 s with a slight variance. Figure 3.4b shows latency for medium-sized packets where CoT event types such as markers, routes, and shapes fall under. We see the latency for markers and simple shapes are 2.15 s on average, and with increased detail and complexity, the latency reaches upwards of 2.5 s. Finally Figure 3.4c shows latency for bulk data. In this experiment, we had a compressed file enclosing an image sized 6 KB. We observe that the latency is around 20.1 s.

Conclusion and Future Work

In this chapter we presented a COTS network deployment with Beartooth MKII radios and Gateways to provide SA to first responders and military forces spanning multiple Beartooth and IP networks. We believe extending the usage of COTS radios

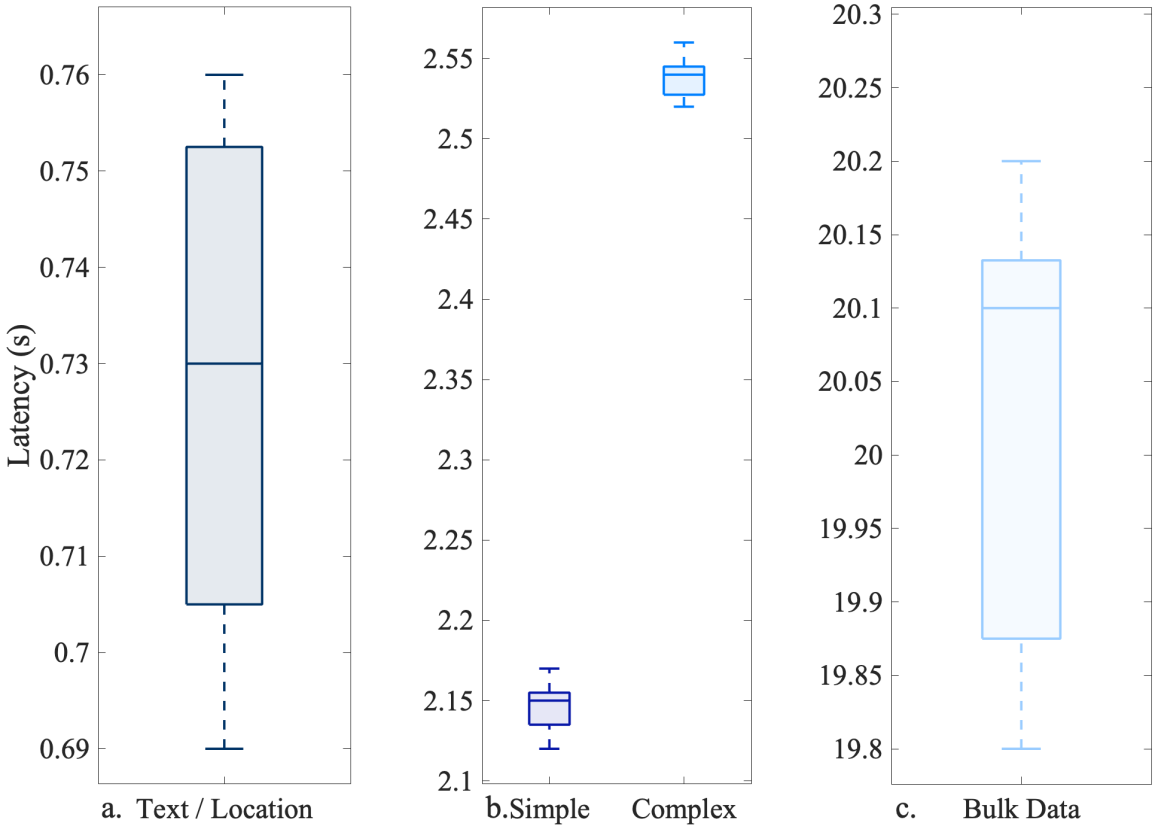


Figure 3.4: Perceived latency of various data types and sizes.

in the form of Beartooth MKII radios and introducing Gateway to interconnect IP networks through TAK can provide scalable SA at low-cost, with a compact form factor. Our experiments show Beartooth devices can support real-time text, location as well as more complex SA data types such as markers, shapes and bulk data with tolerable latency. With several active war zone deployments and military training the MKII radios and Gateways are currently being battle tested further.

In the future, we plan to integrate Beartooth Plugin with different systems and ATAK plugins. Sending various types of sensor data using Beartooth radio over a wide network is the next step toward increasing the system capabilities and growing the market for Beartooth devices. Moreover, on Gateways, we plan to implement parallel processing with multi-threading to increase the number of MKII radios Gateway can support.

CONCLUSION

The integration and advancement of wireless communication technologies have played a crucial role in shaping modern applications, from consumer products to military applications. This thesis presented two innovative approaches to address the challenges in diverse areas of wireless communication by proposing the Beartooth Relay Protocol (BRP) and the Beartooth MKII radio system.

BRP, a flexible MAC protocol designed for LoRa, aims to enhance the performance of real-time and streaming applications. By employing BRP with an advanced LoRa radio, the protocol can address the limitations of previous LoRa MAC protocols, such as constrained flexibility, high latency, and difficulties with real-time data transmission. This development expands LoRa's applicability to a more diverse range of IoT scenarios in a wider range of applications.

Further, the Beartooth MKII radio system, based on the low-cost and power-efficient XBee radio platform, provides a scalable and cost-effective situational awareness (SA) solution for military applications. The Beartooth MKII radio's integration with the Android Tactical Assault Kit (ATAK) and the Beartooth Gateway enable seamless and secure communication among squads, commanders, and command centers. Additionally, the compact form factor of the Beartooth MKII radio offers significant weight reduction compared to existing solutions.

In summary, the proposed solutions demonstrate the potential for advancing the capabilities of wireless communication technologies in both civilian and military contexts. By leveraging the flexibility and configurability of these systems, it is possible to address the challenges associated with real-time data streaming and situational awareness in a wide range of applications. Future research should focus on evaluating on further optimizing these systems for specific use cases and evaluating

their performance in real-world scenarios. For instance, future research could involve examining various voice encoders and neural network voice encoders (neural vocoders) along with their respective performances, as well as analyzing real-time sensor data and exploring diverse methods of compression.

REFERENCES CITED

[1] "2019 Broadband Deployment Report," Federal Communications Commission (FCC), Washington, D.C., Tech. Rep. 19-44, 2019.

[2] P. Mach, Z. Becvar, and T. Vanek, "Internet of Mobile Things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs Standards and Supported Mobility," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, Oct. 2018.

[3] D. Zorbas, K. Abdelfadeel, P. Kotzanikolaou, and D. Pesch, "TS-LoRa: Time-slotted LoRaWAN for the industrial internet of things," *Computer Communications*, vol. 153, Mar. 2020.

[4] M. Bor, J. Vidler, and U. Roedig, "LoRa for the internet of things," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, Feb. 2016.

[5] L. Leonardi, F. Battaglia, and L. L. Bello, "RT-LoRa: A medium access strategy to support real-time flows over LoRa-Based networks for industrial IoT applications," *IEEE Internet of Things J.*, vol. 6, no. 6, pp. 10 812–10 823, Dec. 2019.

[6] L. Leonardi, F. Battaglia, G. Patti, and L. L. Bello, "Industrial LoRa: A novel medium access strategy for LoRa in industry 4.0 applications," in *IEEE Industrial Electronics Society*, Dec. 2018.

[7] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *MDPI Sensors*, vol. 16, no. 9, Sep. 2016.

[8] "LoRa — Platform for IoT," [Online]. Available: https://www.semtech.com/lora.

[9] J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and Unknown Facts of LoRa: Experiences from a Large Scale Measurement Study," *ACM Trans. Sensor Netw.*, vol. 15, no. 2, Feb. 2019.

[10] Q. L. Hoang, H. P. Tran, W.-S. Jung, S. H. Hoang, and H. Oh, "A slotted transmission with collision avoidance for LoRa networks," *Procedia Computer Science*, vol. 177, Nov. 2020.

[11] R. Piyare, A. L. Murphy, M. Magno, and L. Benini, "On-demand tdma for energy efficient data collection with LoRa and wake-up receiver," in *Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2018.

[12] R. K. Singh, R. Berkvens, and M. Weyn, "Time synchronization with channel hopping scheme for LoRa networks," in *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, Oct. 2020.

[13] W. Wu, Y. Li, Y. Zhang, B. Wang, and W. Wang, "Distributed queueing-based random access protocol for LoRa networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, Oct. 2020.

[14] R. E. Chall, S. Lahoud, and M. E. Helou, "LoRaWAN Network: Radio Propagation Models and Performance Evaluation in Various Environments in Lebanon," *IEEE Internet of Things J.*, vol. 6, no. 2, pp. 2366–2378, Mar. 2019.

[15] N. Abramson, "Development of the ALOHANET," *IEEE Transactions on Information Theory*, vol. 31, no. 2, Mar. 1985.

[16] H. T. Yew, M. F. Ng, S. Z. Ping, S. K. Chung, A. Chekima, and J. A. Dargham, "IoT based real-time remote patient monitoring system," in *IEEE International Colloquium on Signal Processing Its Applications (CSPA)*, Apr. 2020.

[17] M. Hadded, P. Muhlethaler, A. Laouiti, and L. A. Saidane, "A centralized TDMA based scheduling algorithm for real-time communications in vehicular ad hoc networks," in *Software, Telecommunications and Computer Networks (SoftCOM)*, Dec. 2016.

[18] "Team Awareness Kit: Tactical Situational Awareness Solution," Fact Sheet, Department of Homeland Security, Mar. 2020, [Online]. Available: https://tinyurl.com/TAKInfoSheet.

[19] "Codec 2," Jul. 2019. [Online]. Available: http://www.rowetel.com/?page_id=452

[20] "AN1200.22 LoRa™ Modulation Basics," Application Note, Semtech Corporation, May 2015, [Online]. Available: https://tinyurl.com/LoRaModBasics.

[21] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of LPWANs: range evaluation and channel attenuation model for LoRa technology," in *IEEE ITS Telecommunications (ITST)*, Dec. 2015.

[22] LoRa Alliance Technical Committee, "LoRaWAN™ 1.1 Specification," Contribution, LoRa Alliance™, Oct. 2017, [Online]. Available: https://tinyurl.com/LoRaWAN11Specs.

[23] "Symphony Link ™ vs. LoRaWAN ™: A Guide for Engineers and Decision Makers," White Paper, LinkLabs, 2016, [Online]. Available: https://tinyurl.com/SymphonyVsLoRaWAN.

[24] "Certification Test Report For Bluetooth and LoRa Cellphone Connection Assistant," Apr. 2017. [Online]. Available: https://tinyurl.com/FCC-BTR-REP

[25] "FCC Code of Federal Regulations Title 47 Part 15 Radio Frequency Devices," Jun. 2020, [Online]. Available: https://tinyurl.com/FCCTitle47P15.

[26] B. Mekiker, M. P. Wittie, J. Jones, and M. Monaghan, "Beartooth relay protocol: Supporting real-time application streams over LoRa," Jul. 2020, [Online]. Available: https://arxiv.org/abs/2008.00021.

[27] G. Boquet, P. Tuset-Peiro, F. Adelantado, T. Watteyne, and X. Vilajosana, "LR-FHSS: Overview and performance analysis," Dec. 2020.

[28] "Startegy for the Long Haul: The US Marine Corps Fleet Marine Forces for the 21st Century," Report, CSBA Online, 2008. [Online]. Available: https://tinyurl.com/CSBA-Report

[29] "STREAMCASTER RADIOS ," Spec. Sheet, Silvus Technologies, 2021, https://tinyurl.com/silvus-scaster-specs.

[30] "TSM Shadow Product Datasheet," Spec. Sheet, TrellisWare Technologies, 2021. [Online]. Available: https://tinyurl.com/trellis-tsm-shadow

[31] "MPU5 Spec. Sheet," Spec. Sheet, Persistent Systems, 2021. [Online]. Available: https://tinyurl.com/mpu5-specs

[32] K. Usbeck, M. Gillen, J. Loyall, A. Gronosky, J. Sterling, R. Kohler, K. Hanlon, A. Scally, R. Newkirk, and D. Canestrare, "Improving situation awareness with the Android Team Awareness Kit (ATAK)," in *(C3I) Technologies for Homeland Security, Defense, and Law Enforcement XIV*. SPIE, 2015.

[33] "How to blow $6 billion on a tech project," News Article, ArsTechinca, June 2012. [Online]. Available: https://tinyurl.com/six-billion-dollars

[34] "Lighter, faster, smaller equipment needed for soldiers to win, says Gen. Townsend," News Article, US Army, 2018. [Online]. Available: https://tinyurl.com/army-Gen-Townsend

[35] "Team Awareness Kit: Tactical Situational Awareness Solution," Fact Sheet, DHS, 2020. [Online]. Available: https://tinyurl.com/tak-fact-sheet

[36] S. Al-Sarawi, M. Anbar *et al.*, "Internet of Things (IoT) communication protocols: Review," in *IEEE Information Technology (ICIT)*, May 2017.

[37] "Sigfox Technology Overview," Jul. 2019. [Online]. Available: https://tinyurl.com/what-sigfox

[38] A. Khalifeh, H. Salah, S. Alouneh, A. Al-Assaf, and K. Darabkh, "Performance evaluation of DigiMesh and ZigBee wireless mesh networks," in *Wireless Communications, Signal Processing and Networking*, 2018.

[39] "Digi XBee Ecosystem," Online Resource, DIGI. [Online]. Available: https://tinyurl.com/digi-xbee

[40] "Wireless Mesh Networking: ZigBee vs. DigiMesh," White Paper, DIGI, 2018. [Online]. Available: https://tinyurl.com/zigbee-vs-digimesh

[41] "Digi XBee SX 900 RF Module," 2022. [Online]. Available: https://tinyurl.com/xbee-specs

[42] "Team Awareness Kit (TAK) - Enhancing Homeland Security Enterprise Collaboration on the Mobile Edge," White Paper, DHS and S&T, 2019. [Online]. Available: https://tinyurl.com/tak-dhs

[43] N. Soule, B. Kalashian, C. Rock, and C. L. Tomcho, "Software acoustic modem for tak communications with analog radios at the tactical edge," in *MILCOM*, 2021.

[44] B. Mekiker, M. P. Wittie, J. Jones, and M. Monaghan, "Beartooth relay protocol: Supporting real-time application streams with dynamically allocated data reservations over LoRa," in *Computer Communications and Networks (ICCCN)*, 2021.

[45] "Protocol Buffers - Google's data interchange format," GitHub Repo, Google. [Online]. Available: https://tinyurl.com/protobuf-github

[46] "Protocol Design Whitepaper," White Paper, ZeroTier, 2018. [Online]. Available: https://tinyurl.com/zerotier-protocol