Mobile Device-to-Device Communication for Crisis Scenarios Using Low-Cost LoRa Modems

Jonas Höchst, Lars Baumgärtner, Franz Kuntke, Alvar Penning, Artur Sterz, Markus Sommer, and Bernd Freisleben

Abstract We present an approach to enable long-range device-to-device communication between smartphones in crisis situations. Our approach is based on inexpensive and readily available microcontrollers with integrated LoRa hardware that we empower to receive and forward messages via Bluetooth, Wi-Fi, or a serial connection by means of a dedicated firmware, called *rf95modem*. The developed firmware cannot only be used in crisis scenarios but also in a variety of other applications, such as providing a communication fallback during outdoor activities, geolocation-based games or broadcasting of local information. We present two applications to show the benefits of our approach. First, we introduce a novel device-to-device LoRa chat application that works on both Android and iOS as well as on traditional computers like notebooks using a console-based interface. Second, we demonstrate how other infrastructure-less technology can benefit from our approach by integrating it into the DTN7 delay-tolerant networking software. Furthermore, we present the results of an in-depth experimental evaluation of approach consisting of (i) real-world device-to-device LoRa transmissions in urban and rural areas and (ii) scalability tests based on simulations of LoRa device-to-device usage in a medium-sized city with up to 1000 active users. The firmware, our device-to-device chat application, our integration into DTN7, as well as our code fragments of the experimental evaluation and the experimental results are available under permissive open-source licenses.

Keywords LoRa · Disaster communication · Device-to-device communication

J. Höchst (✉) · A. Penning · A. Sterz · M. Sommer · B. Freisleben
Department of Mathematics and Computer Science, University of Marburg, Marburg, Germany
e-mail: hoechst@mathematik.uni-marburg.de

L. Baumgärtner · F. Kuntke
Department of Computer Science, Technical University of Darmstadt, Darmstadt, Germany

H. J. Scholl et al. (eds.), *Disaster Management and Information Technology*,
Public Administration and Information Technology 40,
https://doi.org/10.1007/978-3-031-20939-0_12

Introduction

The functionality of today's smartphones and other mobile devices highly depends on the availability of telecommunication infrastructures, such as Wi-Fi or cellular technology (e.g., 3G, 4G, 5G). However, there are situations in which no communication infrastructure is available, e.g., in remote areas (Gardner-Stephen, 2011), in the agricultural sector (Elijah et al., 2018), as a result of disasters (Manoj & Baker, 2007), or due to political censorship (Liu et al., 2015). Furthermore, in countries with less evolved infrastructures, e.g., due to low population densities or due to economic reasons, cellular networks often cannot be used at all or cannot be established in an economically feasible manner. In this case, low-cost communication technologies would give people the possibility to communicate with each other (Kayisire & Wei, 2016). However, while modern infrastructure-independent technologies do exist, these are often only accessible to advanced users due to regulations, high costs, or technical complexity. To make these technologies accessible to a broad user base, they need to be integrated into devices already known to users.

We propose to use LoRa wireless technology as a communication enabler in such situations. LoRa (long range) is a long-range and low-power network protocol designed for the Internet of Things (IoT) to support low data rate applications (Hornbuckle, 2010). It consists of a proprietary physical layer, using the chirp spread spectrum (CSS) in the freely usable ISM bands at 433, 868, or 915 MHz, depending on the global region. The additional MAC layer protocol LoRaWAN is designed as a hierarchical topology. A set of gateways is receiving and forwarding messages of end devices to a central server that processes the data. While LoRa itself has to be licensed by the Semtech company and implemented in specific hardware, it is independent of LoRaWAN and can thus be used in a device-to-device manner.

In this chapter, we present an approach to equip existing mobile devices with LoRa technology, by distributing small System-on-a-Chip (SoC) devices supporting multiple Radio Access Technologies (RATs). There are several commercially off-the-shelf microcontroller units (MCUs) available that support Wi-Fi, Bluetooth, and LoRa. We propose to use these low-cost devices to upgrade existing smartphones, laptops, and other mobile devices for long-range infrastructure-less communication. To reach this goal, we present a custom firmware for Arduino-SDK compatible boards, called *rf95modem*. Existing mobile devices can be connected to a board through a serial connection, Wi-Fi, or Bluetooth. As a general solution, we propose to use modem AT commands as an interface for application software. This interface can then be exposed through different communication channels and used by application software without requiring LoRa-specific device drivers. Since these boards are cheap and do not require laying new cables or setting up communication towers, these boards can either be distributed to people living in high-risk areas beforehand or handed out by first responders during the event of a crisis. We further formulate possible applications for the daily usage of these devices to incentivize people buying and using these devices during nonemergency times, such as geolocation-based games or broadcasting of local information, so that they are available and ready to use when an emergency occurs.

To demonstrate the functionality of our implementation, we first present a cross-platform mobile application for device-to-device messaging. This re-enables basic infrastructure-less communication capabilities in disasters. Second, we present an integration of our implementation into a disruption-tolerant networking (DTN) software. Although the low data rates of LoRa are not sufficient to support multimedia applications, sensor data, e.g., in agricultural applications or environmental monitoring, as well as context information for further DTN routing decisions can be transmitted through the LoRa channel. To illustrate the benefits of our approach, the developed device-to-device messaging app as well as our DTN integration are tested through experimental evaluations in an urban and a rural area. Furthermore, to demonstrate the feasibility but also the limitations of this approach, we simulated and evaluated a scenario with up to 1000 users.

To summarize, we make the following contributions:

- We present a novel free and open-source modem firmware implementation for LoRa-enabled MCUs, featuring a device-driver-independent way of using LoRa via serial, Bluetooth LE, and Wi-Fi interfaces.
- We present a novel device-to-device LoRa chat application for (i) Android and iOS smartphones and (ii) traditional computers.
- We present a freely available and open-source integration of long-range communication into a delay-tolerant networking software.
- We experimentally evaluate the proposed approach by conducting field tests in an urban environment as well as in a rural area and perform energy measurements of multiple devices.
- We demonstrate the scalability of our approach by simulating and evaluating large application scenarios with up to 1000 users.
- The presented rf95modem software,[1] the device-to-device chat application,[2] the integration into DTN7,[3] the experimental evaluation code fragments,[4] and the results as well as the evaluation code of the scalability test[5] are freely available.

This chapter is organized as follows. Section "Related Work" discusses related work. Section "Design" presents the design of approach to enable device-to-device communication between smartphones using LoRa. Implementation issues are described in section "Implementation". Section "Experimental Evaluation" presents the results of our experimental evaluation. Section "Conclusion" concludes this chapter and outlines topics for future work.

[1] https://github.com/gh0st42/rf95modem/, MIT License.

[2] https://github.com/umr-ds/BlueRa, MIT License.

[3] https://github.com/dtn7/dtn7-go, GNU General Public License v3.0.

[4] https://github.com/umr-ds/hoechst2020lora

[5] Will be released with the final version of the chapter.

Related Work

Augustin et al. (2016) experimentally evaluated the foundations of LoRa. The authors built a LoRa testbed and conducted different tests including receiver sensitivity and network coverage. LoRa's Chirp Spread Spectrum (CSS) modulation technique allows users to decode the received signals from -120 to -125 dBm, depending on the spreading factor (SF). The network coverage was examined in a suburb of Paris using SFs of 7, 9, and 12, based on different test locations. With SF7 and SF9, distances of 2.3 km were reached with less than 50% packet loss. Using SF12, the packet delivery ratio at the highest distance of 3.4 km was 38%.

Bor et al. (2016) investigated the current LoRaWAN protocol and proposed an alternative MAC layer to be used with LoRa, making use of multi-hop communication. Wixted et al. (2016) evaluated the properties of LoRaWAN for wireless sensor networks, demonstrating reliable usage of LoRa up to 2.2 km in an urban scenario.

Baumgärtner et al. (2018) proposed to use LoRa for environmental monitoring. In the included LoRa evaluation, ranges of 4.6–6.5 km with the base station placed on a high building were achieved depending on the antenna and the frequencies in use. Furthermore, the concept of a unified radio firmware was introduced, but only limited functionality was implemented and evaluated.

Long-range peer-to-peer links were investigated by Callebaut et al. (2019). The authors showed experimentally that with an increased SF, the received signal strength (RSS) did not change, but the signal-to-noise ratio (SNR) was increased, proving the better decoding ability. Distances of up to 4 km in line of sight and 1 km in forested terrain were achieved.

Deepak et al. (2019) created an overview of wireless technologies for post-disaster emergency communication. They identified three disaster network scenarios: congested network, partial network, and isolated network. In isolated networks, the user devices have to deploy a new network to provide temporal wireless coverage. This could be achieved with drone-assisted communication or mobile ad hoc networks (MANETs). The advantage of the latter is high redundancy: a failure of individual nodes is not necessarily mission critical.

Lieser et al. (2017) analyzed multiple disaster scenarios to highlight the main communication issues that occurred. The depicted scenarios are based on unavailable or broken communication infrastructures. In particular, the authors proposed an architecture that incorporates delay-tolerant MANETs to be independent of any fixed infrastructure. Additionally, the authors focused on communication tools that ordinary civilians can use, since civilians typically do not possess their own dedicated communication facilities, in contrast to disaster relief organizations.

By analyzing 49 crisis technology articles that focus on mobile apps in disaster situations, Tan et al. (2017) illustrated that disaster communication is shifting away from authority-centric approaches toward approaches that integrate and engage the public. The authors argued that supporting on-site collaboration (e.g., by chatting) is the main purpose of mobile apps for disaster situations.

According to Kaufhold et al. (2018), the widespread use of smartphones provides opportunities for bidirectional communication between authorities and citizens. The authors developed the app 112.social for communication between authorities and citizens during emergencies. The authors argued that further research in the area of infrastructure-less technologies for emergency communication apps is required to provide new opportunities.

Sciullo et al. (2018) presented an infrastructure-less solution for emergency communication by combining LoRa modules with smartphones. In their approach, the LoRa transceiver was hooked directly to the smartphone via USB to achieve higher communication ranges compared to conventional wireless transmission technologies (e.g., Wi-Fi). Thus, only Android devices work with this approach, and the solution is tightly coupled to the emergency communication app provided by the authors. Olteanu et al. (2013) used an USB dongle to access ZigBee nodes through an Android app. These USB-connected devices were later also identified by Sciullo et al. (2020) as being problematic and tackled through the addition of an extra Bluetooth bridge. This setup is still tailored to the provided emergency application of the authors and has higher complexity, bill of materials, and energy consumption compared to our approach.

Berto et al. (2021) are investigating LoRa-based mesh networks that implement peer-to-peer communication between nodes and extend node reachability through multi-hop communication. The evaluation is based on a hardware/software prototype in a real-world scenario, and the scaling of the approach is not investigated further.

Mekiker et al. (2021) propose a LoRa-based radio and relay protocol allowing real-time application traffic on point-to-point and multi-hop connections. The proposed Beartooth Relay Protocol (BRP) aims to extend mobile applications functionality beyond infrastructure coverage areas. An evaluation of the approach is performed using a real scenario, but the crucial test of the scalability of the approach is not part of the work.

Design

In this section, the design goals of the proposed approach are discussed. First, general principles of using LoRa on smartphones are covered. Second, design goals of a generic LoRa modem firmware are presented. Third, requirements for a device-to-device chat application are examined. Finally, thoughts on integrating LoRa into disruption-tolerant networking are presented.

Enabling LoRa on Smartphones

To extend smartphones and other common devices by infrastructure-less communication technologies, a generic interface must be designed. While these devices offer a variety of communication technologies, only few are shared across different categories of devices. Ethernet and USB may be available on most devices including laptops and routers, but smartphones can utilize these connections only using special adapters, if at all. However, all of the mentioned devices offer Wi-Fi and/or Bluetooth interfaces. Furthermore, the used approach should be based on low-energy solutions, since in the described scenarios power supply may be limited or not available. In the following, we present a modem firmware, called rf95modem, for LoRa MCUs that can enable access to the LoRa hardware through other communication channels.

Modem Firmware

Figure 1 shows how different devices can be connected to a modem board. There are several commercial off-the-shelf microcontroller boards available that include a LoRa transceiver and thus can be used for the proposed functionality. With our approach, we aim to support the majority of these boards by providing a hardware abstraction layer across all of them. Thus, the provided implementation supports a wide variety of available boards, e.g., the LilyGO TTGO LoRa series,[6] Adafruit's Feather 32u4 and M0 boards[7] or the Heltec Automation Wi-Fi LoRa 32, and Wireless Stick (Lite).[8] Some of these boards only provide LoRa and a serial interface via USB, but others also provide Wi-Fi and Bluetooth. The modem firmware is supposed to be controllable through AT commands similar to classic modems or various smartphones. Thus, no specific device drivers are needed to send and receive data via the rf95modem firmware. Finally, the firmware should be flexible enough and easily configurable to only ship the code actually needed for the device and the scenario in which it is used.

[6] http://www.lilygo.cn/pro.aspx?FId=t3:50003:3, Xing Yuan Electronic Technology Co., Ltd., LongGang, Shenzhen, China.

[7] https://www.adafruit.com/product/3178, Adafruit Industries, LLC, 150 Varick Street, New York 10013, USA.

[8] https://heltec.org/proudct_center/lora/lora-node/, Heltec Automation, Longtan Industrial Park, Chengdu, China.

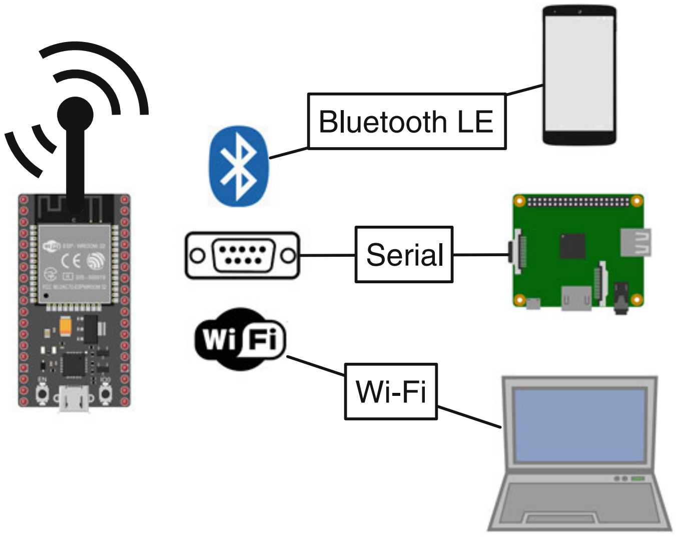


Fig. 1 ESP32-based modem board and its connection options for smartphones, single-board computers, and laptops

Incentivizing LoRa Usage

An important challenge in establishing emergency networks is the availability of hardware and software to users when an emergency happens. If users find themselves in an emergency situation with an infrastructure failure, they either have to wait until the infrastructure is restored; they are equipped with new technology, e.g., from the emergency services; or they can use existing and already known devices and technologies. The latter case has the clear advantage that rudimentary communication and, in particular, emergency calls are possible without involving third parties. For the technology presented here to have an impact, it is necessary for users to be able to use it meaningfully outside of a crisis, to gain experience with it, and to avoid the need for elaborate steps in the event of a crisis. In this section, we outline some use cases where LoRa-based communication is helpful in everyday life and can get users to familiarize themselves with LoRa technology.

A strong use case for LoRa outside of emergency communication is outdoor activities, in which people are in areas of bad or completely missing cellular coverage. While in skiing areas cellular networks are built due to commercial interest, many activities depending on less infrastructure suffer from a missing communications infrastructure. Using LoRa communication among the participants of a group or even among different groups in the same area can be very useful for coordination, e.g., if a part of the group separates and looks for food, water, or firewood. Even in the case of unintentional separation, e.g., if the group gets lost while canoeing, this infrastructure-free communication can be helpful to find each other again. There are several commercially available products that support the case for a companion device offering infrastructure-free communication, such as GoTenna[9] MeshTastic,[10] or Beartooth.[11]

A second incentive for using LoRa is gamification, e.g., by deploying beacons through volunteers in certain locations. Beaconing can be implemented as a service on already existing LoRa infrastructure, such as on LoRa gateways or even on weather stations or other IoT devices. The goal of the game would be to collect as many beacons as possible and thus prove that a player has actually visited the locations. To make cheating in the game more difficult and to introduce another component for the competition of different players, the beacons are generated and signed based on a timestamp.

A third use case for LoRa in everyday life is a public message board that is enhanced by local information. Important information of the city, e.g., for visitors but also for people who live in the city can be announced via LoRa, e.g., local weather recordings, traffic information, or information in potentially dangerous situations, such as power outages, fires, or terrorist attacks. The inherent property of a location-based limitation allows effective and efficient distribution of location-based information and can also be used for marketing purposes.

A Device-to-Device Messaging Application

To enable communications in rural areas or in situations after disasters, mobile applications play an outstanding role for various reasons and support a variety of communication technologies like cellular, Wi-Fi, and Bluetooth. Therefore, we designed a mobile application to support off-grid communication in the scenarios mentioned above. In particular, in crisis situations, it is important that users do not first have to familiarize themselves with new paradigms or UI/UX concepts and are not confronted with technical terms that are incomprehensible to laypersons. Therefore, our application should use Bluetooth Low Energy (BLE) as the primary connection technology. Bluetooth is widely accepted as a technology to create one-to-one connections and exchange data between the involved peers, whereas Wi-Fi is usually used to access information from a central place. Therefore, the Bluetooth paradigm fits better to the given scenario. Additionally, Bluetooth is more energy efficient compared to Wi-Fi, which makes it the appropriate technology to use in this case. To further reduce barriers in app usage, our app should automatically connect to nearby modem devices without any further actions required by the user. This increases the chances of instant access to the communication infrastructure in cases of emergencies. The app should also be able to receive messages in the background,

[9] https://gotenna.com

[10] https://meshtastic.org

[11] https://beartooth.com

e.g., when the user leaves the application. Additionally, the user should not worry about using a specific mobile device. It is therefore crucial to provide a platform-independent application that is usable on the most popular mobile platforms iOS and Android.

To get people in contact as fast as possible without prior exchange of IDs, usernames, or alike, the application should follow a public message board paradigm, similar to Twitter, where users can post short messages to a publicly visible channel. Here, users can send messages visible for all and ask for help or provide status information. This approach gives users easy and fast access to a communications method. Users should have an easy and fast way to find new channels and create channels for specific topics. Finally, users should be presented with a common and familiar look and feel including accessibility features so that no one is excluded.

Disruption-Tolerant Networking

For crisis scenarios, DTN is a technology to enable infrastructure-less communication using an emergency infrastructure in conjunction with existing devices of users (Baumgärtner et al., 2016; Lieser et al., 2017). DTNs benefit from a large number of devices storing and forwarding messages to other devices when they become available. Today, end user-focused DTNs are mostly based on ad hoc Wi-Fi and Bluetooth, since these are available in the mobile devices used by the users. Due to slow data rates and duty cycle restrictions introduced by regulation, LoRa in DTNs is not suited for larger data transmissions, such as multimedia content, but is helpful to transmit context information or small messages. LoRa can be used to connect different local clouds of people, where smaller messages available inside the cloud can be transmitted to another cloud. Modern DTN routing algorithms use context information to reduce overheads introduced by unnecessary transmission (Graubner et al., 2018). We propose to add LoRa to existing delay- and disruption-tolerant networks to enable larger spatial low-bandwidth coverage, in order to propagate small messages and context information. To facilitate the use of LoRa in DTN networks, an exemplary integration should be implemented that can use LoRa via Bluetooth, Wi-Fi, or a serial connection and thus is available on mobile devices and static nodes added in crisis scenarios.

Implementation

In this section, our implementations of the rf95modem firmware, the device-to-device messaging application, and the integration into disruption-tolerant networking software are presented.

Modem Firmware

Since a rf95modem should be controlled by AT commands over all of its available connection mechanisms, handling such commands is an essential part of the implementation. Therefore, this functionality is shared across all supported hardware platforms and connection mechanisms, as shown in Fig. 2. Here, the software components are displayed in blue, while the rest represents the underlying hardware modules. For interaction with users and software, the serial device interface, usually accessible via USB, is always active. Furthermore, the in-/output functions may also be hooked to the Bluetooth Low Energy or Wi-Fi modules we developed, if enabled at compile time. Any output is mirrored to all enabled interfaces that can be used simultaneously.

To achieve optimal results on various hardware platforms and configurations, all features and hardware configurations can be set using build flags. For example, the SPI pin configuration and the underlying CPU architecture must be configured, as well as the base LoRa frequency. Currently, we support ESP32-based boards with RF95-compatible LoRa transceivers as well as some Cortex M0 and ATmega32u4-based boards, such as the ones produced by Adafruit for the Feather line of devices.

For the ESP32 boards, we provide a Wi-Fi mode featuring two different ways of communication that can be used in parallel. In both cases, an access point is opened by the device itself for modem users to connect to. The first mode is UDP-based and just broadcasts the modem output to the local network and interprets incoming AT commands via datagram packets. This is especially useful if many local devices want to listen on incoming transmissions. The second mode is the TCP exclusive mode. Here, a single TCP connection is accepted that can then control the model similarly to a serial interface. Since the ESP32 boards also feature Bluetooth, they can be used to announce a BLE characteristic for interaction with the rf95modem. This interface acts similarly to the others by interpreting strings received via a write characteristic as AT modem commands. The output is shared via a notify characteristic to which devices can subscribe. BLE is supposed to have a payload limit of 20 bytes, and thus splitting the serial output into smaller chunks is necessary.

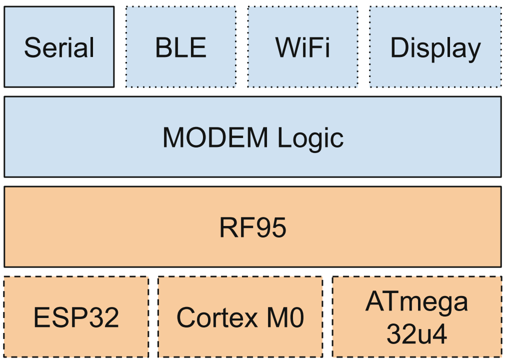


Fig. 2 Overview of the rf95modem architecture

Our tests on various platforms, e.g., iPhones and Raspberry Pis, have shown that sending much larger packets via BLE is often also possible and much more efficient. Therefore, sending overlong frames via BLE can optionally be activated during runtime via a specific AT command. Finally, there is also a software module to support OLED displays as they are pretty common on TTGOs and Heltecs ESP32 devices. If enabled at compile time, these can be used to display status information such as the current frequency, packets received, and number of packets transmitted, which can be used for debugging or providing statistical information at a glance without the need for special hardware or software.

Since all board-specific features can be configured at compile time, the firmware can be custom-tailored to fit even very resource-limited devices. Enabling all features at once results in a large firmware, which requires more flash memory and a custom partition layout, but still works on the most common ESP32 boards. Due to the fact that all output is mirrored between the interfaces, one can easily use two interfaces in parallel, e.g., debugging the BLE communication via an attached serial cable. The firmware is completely written in C/C++ using the Arduino SDK and PlatformIO as a build system.

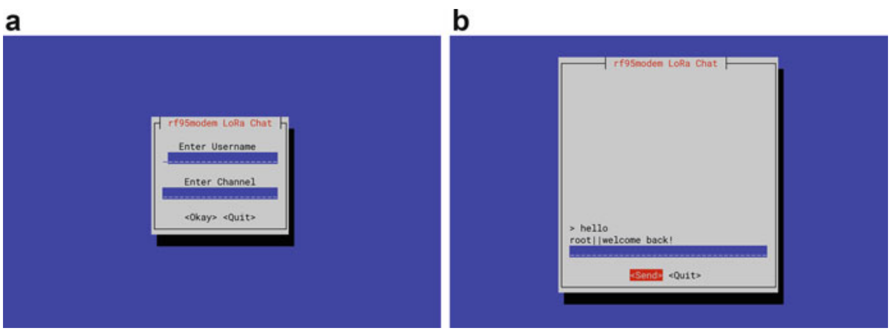
A Device-to-Device Messaging Application

To satisfy the requirements of the messaging application, we provide two different approaches. First, we provide a console-based user interface for traditional computers, as shown in Fig. 3.[12] Second, for the mobile version of the application (BlueRa), we used the Flutter UI toolkit.[13] Flutter allows developers to create platform-independent apps for both major mobile operating systems, iOS and Android, using the same code base.

Figure 4 gives a simplified overview of the components of the app. The top block shows the UI classes. The application starts at the home screen, which contains a path to the settings, a list view of the available channels and a path for joining to new channels or to create channels. On the left, users can change their usernames or manage the app's Bluetooth connection, each in their own screens (the username settings screen is not shown in the figure). When the app's route heads over to the JoinChannelScreen, a list of available channels that the user has not joined yet is presented. Additionally, this screen enables the user to create new channels. The final screen is the chat screen itself, where the user can see a history of the messages in this particular channel as well as a text field for creating and sending new messages.

[12] https://github.com/gh0st42/rf95modem-rs

[13] https://flutter.dev



> hello
root||welcome back!

Fig. 3 Console-based rf95modem LoRa chat example. (**a**) Login screen. (**b**) Chat interface

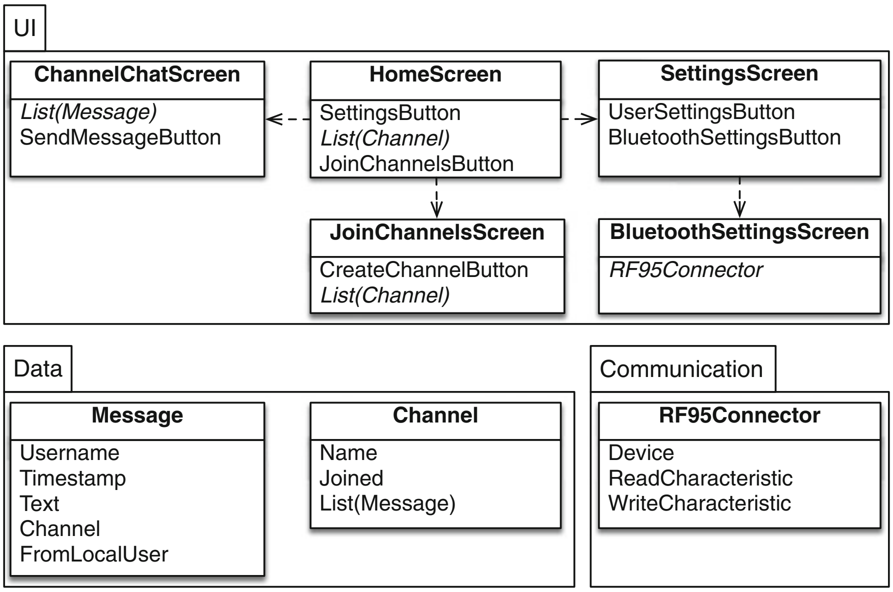
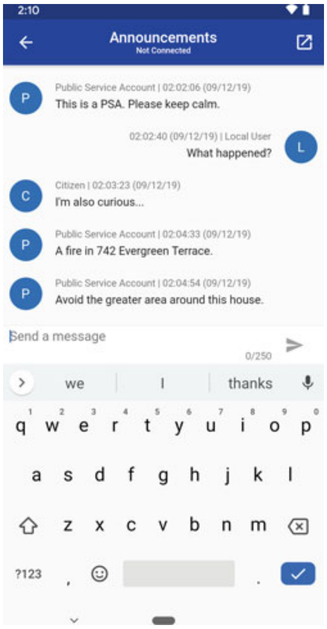


Fig. 4 Overview of the components of the app

Figure 5 shows the chat screen for the announcements channel. Using this common chat UI/UX, the user gets a familiar look and can start messaging immediately, without the need of getting familiar with a special UI.

As indicated by the Channel module in Fig. 4, a channel has a name, an indicator whether the local user has joined this channel and a list of messages. A message, on the other hand, contains the name of the user who sent this message, a timestamp, the text itself, the channel name, and an indicator whether the message was sent from the local user.

The connection to the rf95modem device is implemented in its own module, RF95Connector. This module holds the device ID and Bluetooth connection state,


2:10

Announcements
Not Connected

Public Service Account | 02:02:06 (09/12/19)
This is a PSA. Please keep calm.

02:02:40 (09/12/19) | Local User
What happened?

Citizen | 02:03:23 (09/12/19)
I'm also curious...

Public Service Account | 02:04:33 (09/12/19)
A fire in 742 Evergreen Terrace.

Public Service Account | 02:04:54 (09/12/19)
Avoid the greater area around this house.

Send a message
0/250

Fig. 5 Screenshot of the chat screen for the announcements channel

as well as the read and write characteristics for the serial communication service. Additionally, this module also implements data and message handling. When sending a new message, all required data are serialized to the appropriate format and sent to the modem using the write characteristic. Furthermore, a receive listener gets notified, as soon as new data is available in the read characteristic. The received data is parsed, and the internal channel and message database is updated. If the channel of the received message is already present, the message is appended to the channel's message list. Otherwise, a new channel in the local database is created with the received message. This new channel will be presented in the JoinChannelScreen, so that users can join this channel if they want to.

We use a simple communication protocol for sending and receiving messages. The message is encoded using the Concise Binary Object Representation (CBOR) format, which allows only a small overhead for encoding the message. A message encoded according to this scheme requires an additional five byte overhead for the CBOR encoding, which is comparatively low. Every message sent contains the channel name the message belongs to, the sending user's name, the message itself, and, optionally, the position of the sending user. Channel and username and the message are encoded as strings, and the location is encoded in the form of two single precision floats. Using single precision floats reduces GPS accuracy to roughly 3 m,[14] which is sufficient for estimating a user's location. To make use of the location being sent with every message, we added a map view showing the last

[14] https://sites.google.com/site/trescopter/Home/concepts/required-precision-for-gps-calculations

received message of every user in a channel on the map. This gives an overview of where the communication partners were last seen, e.g., so that first responders can quickly use this information to coordinate a mission.

Disruption-Tolerant Networking

To use LoRa in disruption-tolerant networking, we have extended the DTN7 implementation[15] introduced by Penning et al. (2019). Within the DTN context, the communication interface for bundle exchange between nodes is called convergence layer. We have implemented the convergence layer interface provided by DTN7 to achieve LoRa support.

To integrate rf95modem's serial link into DTN7, we have first developed a library,[16] written in the Go programming language. This library's main task is to provide Golang typical interfaces for writing and reading data streams through rf95modem. Furthermore, status information of the modem can be read and reconfigured.

Until now, DTN7 only had support for unicast convergence layers, while the transmission of LoRa packets corresponds to a broadcast. Since most broadcast technologies are similar in structure, we first developed a generic broadcasting convergence layer, the Bundle Broadcasting Connector (BBC). Its simplified implementation model is shown in Fig. 6.

The main component of the BBC package is the connector that implements DTN7's convergence layer interfaces for both sending and receiving bundles. The connector itself communicates with a modem, which is an interface implemented in rf95modem-go and a mock object for testing. Each modem reports its MTU such that transmissions can be fragmented accordingly.

With regard to transmissions, the BBC makes a distinction between incoming and outgoing ones. Both types have an identifier and can determine whether they have finished. If a bundle should be sent via our BBC, an outgoing transmission with a new identifier will be generated. This identifier is derived from the node. Every node is initialized with a random identifier, which is then incremented for each transmission. The payload is the xz-compressed bundle. As long as the transfer is not completed, the connector requests a new fragment. Its length including headers must not exceed the modem's MTU. This is then handed to the modem, which broadcasts it via LoRa in our case.

The network protocol specification of a fragment is shown in Fig. 7. A fragment itself consists of a header of two bytes, followed by the payload. In the header, the identifier of the transmission is referenced next to a sequence number. Each fragment contains the incremental sequence number of its predecessor. Thus, lost

[15] https://github.com/dtn7/dtn7-go

[16] https://github.com/dtn7/rf95modem-go

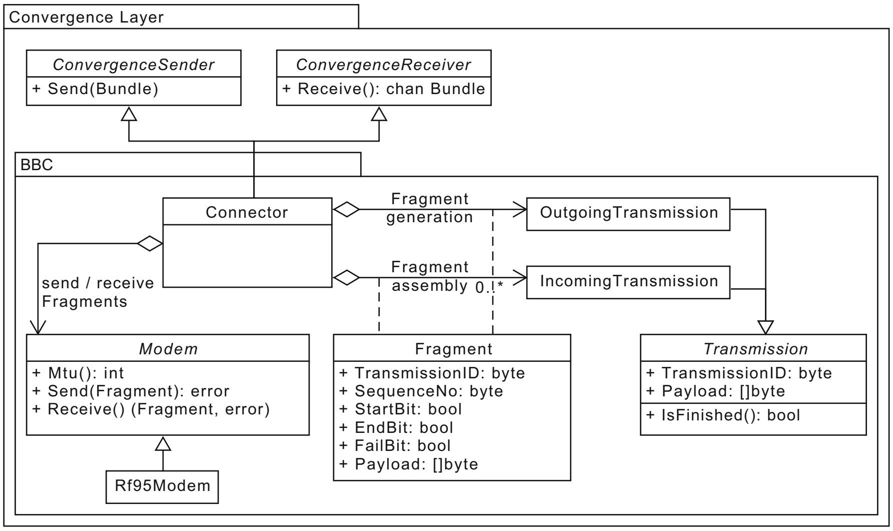


Fig. 6 Simplified implementation model of the Bundle Broadcasting Connector

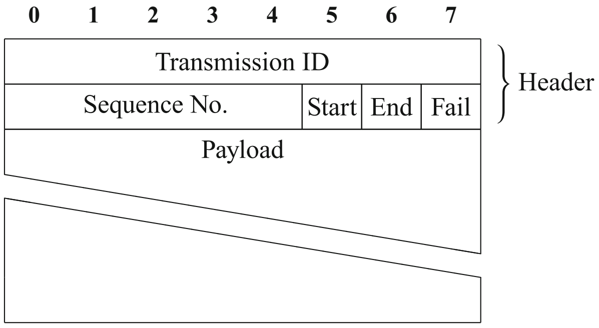


Fig. 7 Protocol specification of a fragment

fragments can be detected in advance. In addition, the header has three flags. The start bit indicates the beginning of a new transmission, while the end bit indicates its end. A failure bit is set for status packets that imply the absence of fragments.

When receiving fragments, the modem forwards them to the connector. This checks whether the transmission identifier is already known. If this is the case, the fragment is added to the incoming transmission. Otherwise, a new incoming transmission is created. Once the transmission is finished, the entire payload is extracted and decompressed. The resulting bundle will be passed back to DTN7's logic. However, if a reception error occurred, e.g., due to a skipped sequence number, a status packet is sent. This packet is equal to the last fragment, except that the failure bit is set and the payload is empty. Reception of such a packet by the sender marks the transmission as faulty. As a result, DTN7 will re-trigger the transmission at a later time.

Experimental Evaluation

In this section, LoRa protocol properties are discussed, and the presented implementations are evaluated through experiments.

LoRa in Device-to-Device Scenarios

LoRa as a long-range protocol is limited in terms of bandwidth, since the resilient encoding scheme introduces some overhead and a duty cycle needs to be followed to fairly use the shared medium. To understand the limitations of LoRa communication, some application-oriented examples are discussed.

Figure 8 shows the payload sizes compared to the airtime required for sending with different spreading factors (SF), where the coding rate is set to 4/5. The presented SF and channel bandwidth examples are taken from the EU standards (LoRa Alliance, 2018). The message length of LoRa is limited depending on the SF to limit the airtime each individual message requires. The highest SFs are limited to a payload of 51 bytes. Using SF9, the payload can go up to 115 bytes, and in the fastest SFs 8 and 7, messages can contain up to 222 bytes. SF12 packets, with the maximum payload of 51 bytes, take up to 1.92 seconds airtime, while 222

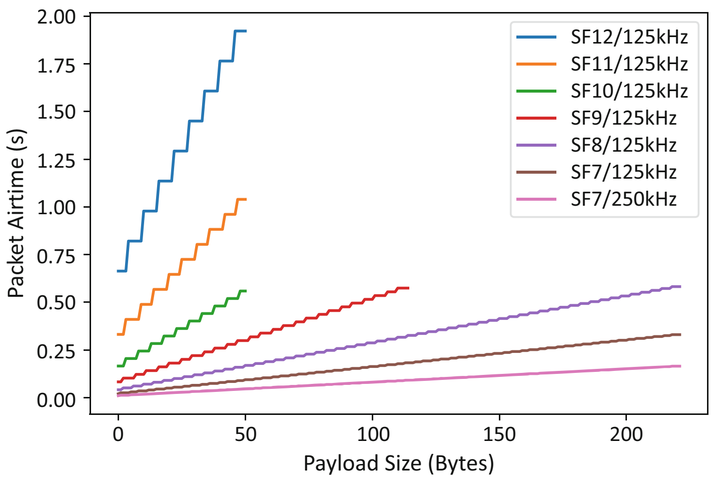


Fig. 8 Exemplary packet airtime in different LoRa profiles

bytes in SF7/250 kHz only take 0.16 seconds. When using LoRa for emergency communication, different profiles can be used to model, e.g., the importance of messages. Public service announcements of governmental institutions, including messages of rescuers, can be sent in more resilient configurations, while chats of users helping each other in emergency situations can be limited to smaller areas, to cope with the limitations of the protocol.

Device-to-Device Smartphone Communication

We evaluated our proposed infrastructure-less LoRa communication via real-world tests that cover two scenarios: (a) city area communication and (b) rural area communication.

The motivation for scenario (a) is communication demands in disaster situations. By having a low-cost companion device that extends the infrastructure-less communication range of our everyday devices could be a real benefit for such scenarios. However, the inherent characteristics of cities, e.g., the high density of buildings, are a major problem for each wireless technology.

Scenario (b) is motivated by the fact that some rural areas, also in industrial countries, are still not covered by mobile networks (GSM, 3G, 4G, 5G). The expectations of the tests in the rural areas therefore differ, since regions without obstacles might easily get good coverage, while areas with many trees might suffer from worse connections.

Experimental Setup

For the conducted tests, we used one fixed and one mobile station. The fixed station consists of a laptop logging the incoming messages. Figure 9 shows the mobile station, consisting of a smartphone in combination with a Heltec Wireless Stick driven by a powerbank. The default antenna was replaced by a +3dBi model, connected via SubMiniature version A (SMA). The antennas of each station were 1.5 m above the ground, in order to model realistic usage in device-to-device scenarios.

First, we selected one exemplary region for each of our two considered scenarios. The fixed station was then placed in the middle of the selected area and started listening for incoming messages. For reproducibility and accuracy, we scripted message generation and sending on the mobile station, such that every 15 seconds one message including a GPS position was sent via Bluetooth LE and broadcasted by the companion device. The mobile station was then moved away from the static station until no message could reach its counterpart anymore. To observe a realistic model of device-to-device communication, the mobile station was moved in multiple directions. The tests in both scenarios were repeated using two LoRa profiles provided by rf95modem: (a) medium range: bandwidth: 125 kHz, Cr: 4/5,

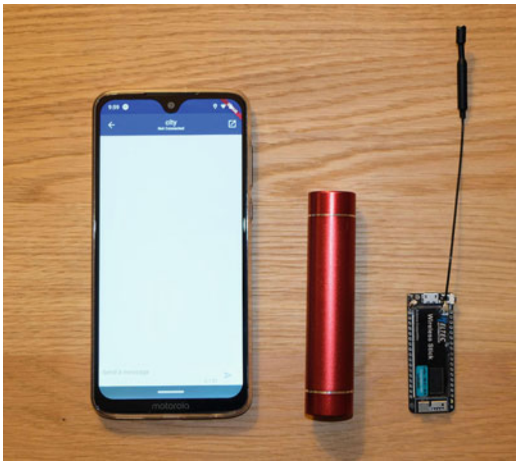


Fig. 9 Mobile station: smartphone, powerbank, and Heltec wireless stick

Table 1 Maximum distances achieved in the different areas and tested LoRa profiles in the conducted experiments

Scenario	Mode	Maximum distance (km)
City	(a) Medium range	1.09
Rural area	(b) Long range	2.89
	(a) Medium range	1.31
	(b) Long range	1.64

SF7 and (b) long range: bandwidth: 125 kHz, Cr: 4/8, SF12. Due to the simplicity of our test procedure, we did not get the maximum possible distances of our exemplary regions, but two real-world setups, with distances that work even with the simple out-of-box experience of the rather low-cost Heltec wireless sticks.

Results

By analyzing the logs of the smartphone applications that transmit GPS locations of each sent message, we were able to calculate the distances of reliable communication setups between all participants for each scenario.

Table 1 shows the maximum distances of the conducted tests. For the medium-range configuration, 1.09 km in the city area and 1.31 km in the rural area could be achieved. With the rather high data rate of 5.47 kbps, the mode is a good choice in dense areas, where a larger amount of messages might occur, and airtime is limited. In the long-range profile, 1.64 km could be achieved in the rural area, while in the city scenario, some messages could be transmitted from 2.89 km range.

Figure 10 shows the results of the conducted tests in the city area. The orange dots denote the medium-range profile, while the red dots show the successful transmissions in the long-range profile. In the size of the markers, the received signal strength indicator (RSSI) is visualized. The larger the marker, the better the RSSI is. Note that in LoRa, a higher SF enables a higher chance of successful decoding

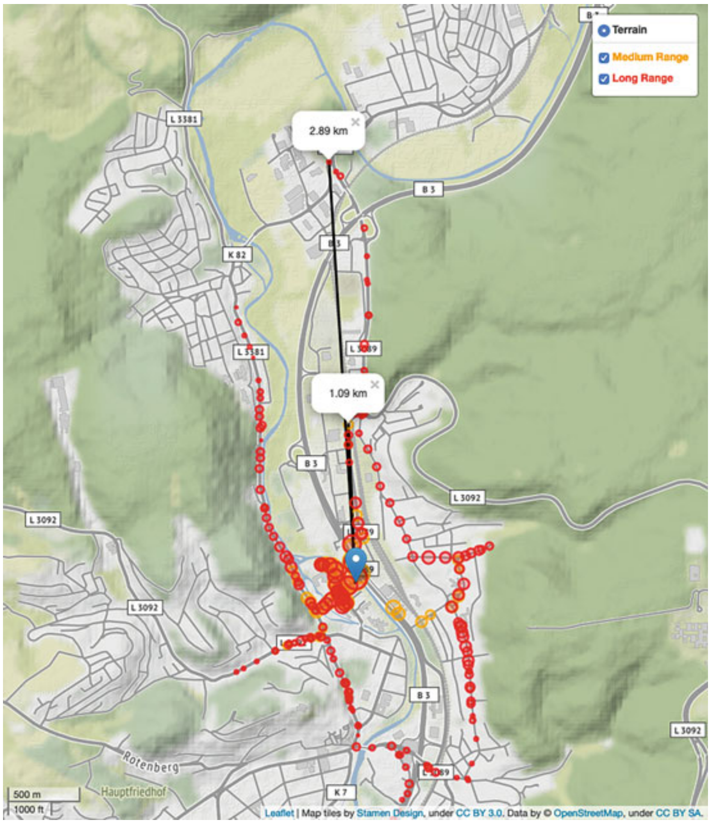


Fig. 10 Successful LoRa transmissions in the city area

under worse RSSI values. In the presented results, it is evident that LoRa works well as long as no obstacles are in the way. The maximum distance in the city area was achieved in the valley going through the city. Even though obstacles, such as buildings, were in the way, the signal could reach the other peer well. When moving behind a hill, such as in the western or eastern parts of the presented map, the signal was not able to penetrate the obstacle.

In Fig. 11, the successful LoRa transmissions of the rural area are presented. As expected, the transmission range in the forested area is worse compared to the unforested area. In the presented example, the northern part of the map consists of a forested area, while the southern part is mostly not forested. From the plot, it can be observed that in the non-forested valley area, RSSI is high, and all LoRa messages are successfully transmitted in both modes. When forested areas and hills are in the line of sight, the RSSI worsens and quickly becomes unavailable. In long-range mode, transmission in forested places improves and messages are successfully transmitted through up to 600 m of forested area.

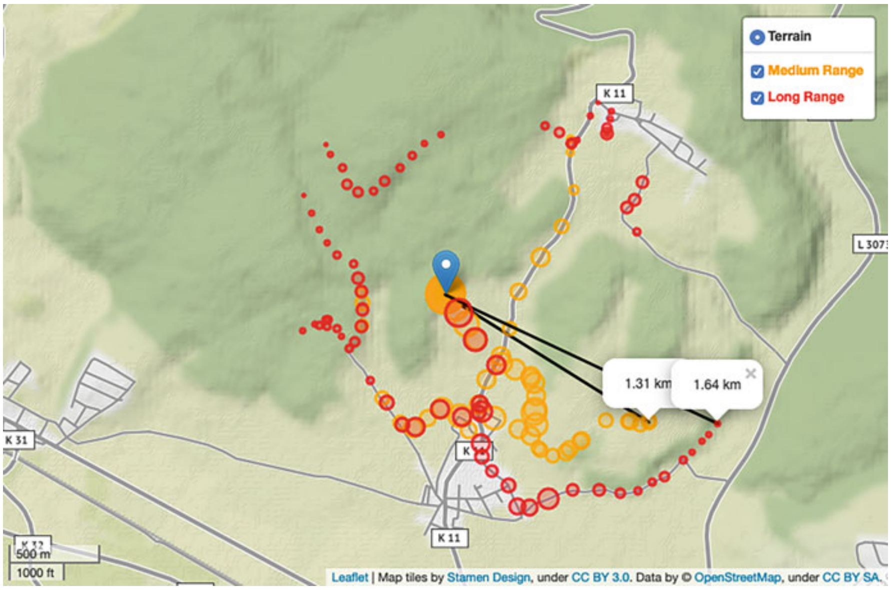

Terrain
Medium Range
Long Range
1.31 km
1.64 km
K 11
K 31
K 32
L 3073
500 m
1000 ft
Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under CC BY SA.

Fig. 11 Geo-positions of successful LoRa transmissions in a rural area

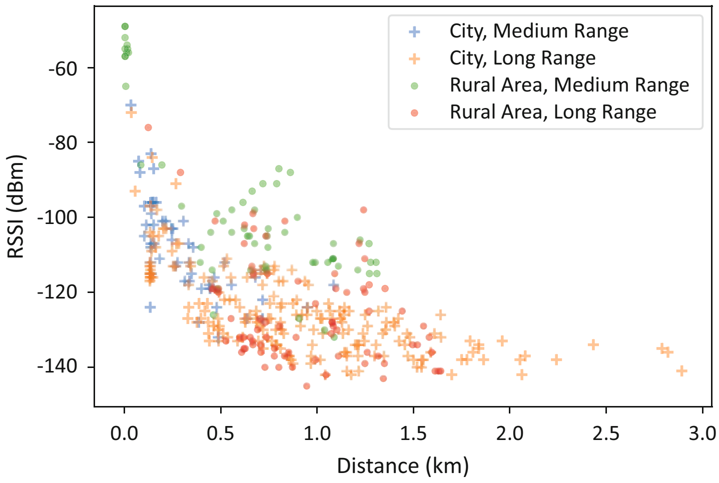


Fig. 12 Received signal strength indicator in relation to transmission distance in the proposed device-to-device scenario

In Fig. 12, the observed RSSI values in relation to the distances are presented. With the long-range profile, signals with RSSI values of up to −140 dBm can be decoded successfully, while in the medium-range profile, the limit is around −130 dBm.

In general, this shows that LoRa is a viable option to enable device-to-device communication in crisis scenarios, where infrastructure is destroyed or temporarily not available. The different profiles of LoRa can be used to limit communication to a certain area and therefore allow higher data rates or cover a larger area and therefore reach out to more people.

Interfacing Emergency Networks

When transmitting data over a disruption-tolerant network, an overhead is generated. This is caused by the additional metadata that a DTN bundle carries, e.g., the sender, receiver, or other blocks of information. In addition, there is now a second overhead for the fragmentation header of the BBC. Due to the small size of a LoRa packet, it is advisable to examine the total size of a transmission and the number of fragments. The benefits or costs of the xz compression should also be considered.

For our evaluation, we created two types of payload data: randomly distributed data and the lorem ipsum placeholder text. The respective payloads were generated in the sizes of the power of two, from 21 to 211. For this purpose, the 445 byte long lorem ipsum text was shortened or repeated accordingly. This payload was wrapped into a DTN packet, sent from dtn://source/ to dtn://destination/ with an additional age block to set the lifetime to 1 hour. The LoRa maximum payload can be up to 251 bytes in size, as instructed by rf95modem in our test configuration.

The overhead of a DTN bundle is 77 bytes without compression. In Fig. 13, the final transmission size and the number of required fragments are shown for the two characteristics of the payload data and its size. It is noticeable that for a random payload, the transmission size is slightly larger. However, the number of fragments is almost always the same. Furthermore, user data is usually not randomly distributed. This is where the advantage of the compression comes into effect, as

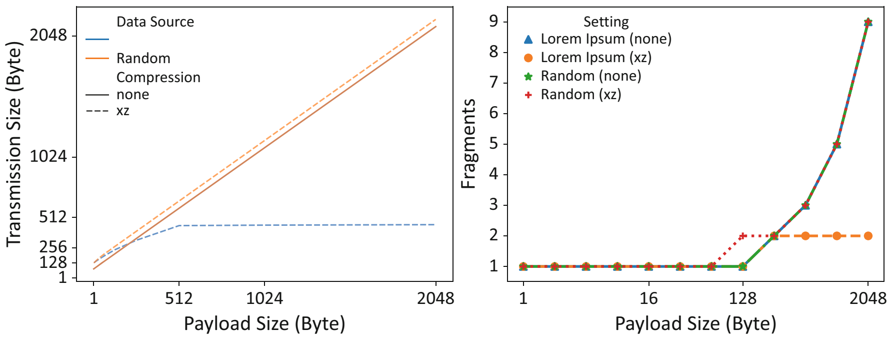


Fig. 13 Total transmission size and amount of fragments for different payloads

it becomes evident especially in the low number of fragments with compressed payloads.

We also carried out a small field test. For this purpose, three DTN nodes were installed, each equipped with a rf95modem for 868 MHz in the short-range profile: 500 kHz, Cr: 4/5, SF7. The nodes were positioned so that only one node had direct radio contact with the other two. Every time a packet is forwarded in a DTN, some metadata is updated, e.g., the previous node to specify the last relaying node. To verify the packet forwarding, we inspected the previous node from the received packet. If this value does not match the packet's sender, it was successfully forwarded. To perform this evaluation, we prepared three nodes, n0, n1, and n2. n1 was positioned in the midpoint; further n0 and n2 were not supposed to have direct contact. Outgoing from n0, packets were sent addressed to n2. These should be transferred from n0 to n1 first and forwarded from n1 to n2 afterward. We then sent DTN packets with a small payload so that they fit into a single LoRa packet. As a result, we observed situations where the previous node was adjusted accordingly. In such a case, the roundtrip time took 1.7 seconds from initiating the transmission to receiving the acknowledgment of reception.

Energy Considerations

While the energy consumption of smartphones is a well-studied field and battery lifetimes of these devices are up to some days, the companion devices studied in this chapter are not evaluated that well. Thus, we measured multiple devices targeted by the proposed firmware in terms of energy usage in different energy states, namely, receiving, sending, and deep sleep. From these measurements, the required battery capacities can be inferred.

The energy consumption was measured using an ODROID Smart Power Meter[17] connected to the microUSB connector of the board and supplied 5 V.

In Table 2, the average energy consumption of the listed boards is presented. Since the boards need to be online to receive messages from other boards, the receiving mode has the highest impact on energy consumption.

The power consumption of the measured boards when receiving data shows a broad variance, e.g., from about 72 mW for the Adafruit Feather 32u4 LoRa board up to 723 mW for the TTGO T-Beam v0.7 board. While sending data, the required power differences become more balanced. When deployed in sensor networks, the deep sleep power consumption becomes important. Four of the tested boards require 49–76 mW in this mode, while one board requires below 1 mW. The values for deep sleep are likely caused by powering the boards through the micro-USB connection, which requires a transformation to the voltage required by the microprocessors.

[17] https://www.hardkernel.com/shop/smart-power/

Table 2 Energy consumption in receiving, sending, and deep sleep modes of rf95modem compatible boards

Board	Receiving (mW)	Sending (mW)	Deep sleep (mW)	Additional features	Price (€)
TTGO T-Fox 27 dB	392	1771	61	Wi-Fi, BLE, OLED, RTC	20
TTGO T-Fox 20 dB	400	902	61	Wi-Fi, BLE, OLED, RTC	20
TTGO LORA ESP32	404	782	68	Wi-Fi, BLE	15
TTGO LORA32 V2.0	393	689	57	Wi-Fi, BLE, OLED, SD	20
TTGO LORA32 V2.1_1.6	387	785	57	Wi-Fi, BLE, OLED, SD	20
Heltec Wireless Stick	391	923	76	Wi-Fi, BLE, OLED	12
Adafruit Feather 32u4 LoRa	72	648	49	–	35
Adafruit Feather M0 LoRa	95	697	<1	–	35
TTGO T-Beam v0.7	723	1125	393	Wi-Fi, BLE, GPS	20

Also, most boards contain a serial to USB converter, which cannot be turned off when powered via USB.

To put these numbers into perspective, we assume a powerbank with a capacity of up to 20,000 mAh. Such powerbanks are widespread and used by smartphone users to recharge their phones. This capacity at 3.3 volts relates to 66 Wh and thus can power the TTGO and Heltec hardware for more than 160 hours. The maximum receiving time can be achieved using the Feather 32u4 LoRa board with more than 900 hours of receiving time.

Scalability

When speaking of LoRa, the question regarding usability with respect to large networks and radio interference arises. As mentioned earlier, LoRa, or more precisely any protocol in the same frequency band, must follow a strict duty cycle of 1% with respect to time. This is mainly to allow for fair use of the radio spectrum and to reduce collisions of packets leading to data loss. In an emergency scenario with users sending messages in an uncontrolled and unrestricted manner, however, it is hardly possible to enforce any such limitation. Thus, in this section, we investigate the limitations of LoRa with respect to three aspects: (a) how many active users can be in the network before rendering it unusable due to too many collisions, (b) how many people can be reached in which distance (i.e., how far do LoRa packets travel), and (c) what can be done to circumvent saturated networks with respect to practical applicability.

Experimental Setup

To perform large-scale tests with a high number of devices sending data using LoRa, we rely on the NS-3 network simulator (Henderson et al., 2008). NS-3 allows us to simulate a high number of users using different physical layer implementations as well as a variety of path loss and propagation models. However, currently, NS-3 does neither support LoRa as the physical layer nor the LoRaWAN data link layer. Thus, we use the LoRaWAN plugin for NS-3 presented by Magrin et al. (2017). By omitting the data link layer implementation and sending data directly to the physical layer, the used LoRaWAN plugin can also simulate the LoRa physical layer without the LoRaWAN data link layer, which emulates the usage of our proposed application.

Due to the duty cycle requirements of LoRa, one main goal of this test is to explore how our system performs under different amounts of network traffic. Furthermore, it is more likely that such a LoRa communication application as proposed in this chapter is reaching its limits in urban environments than in rural areas due to the different population densities. Thus, we modeled a city including suburban areas of 10 km $\times$ 10 km. We assume a population of 100,000 inhabitants,

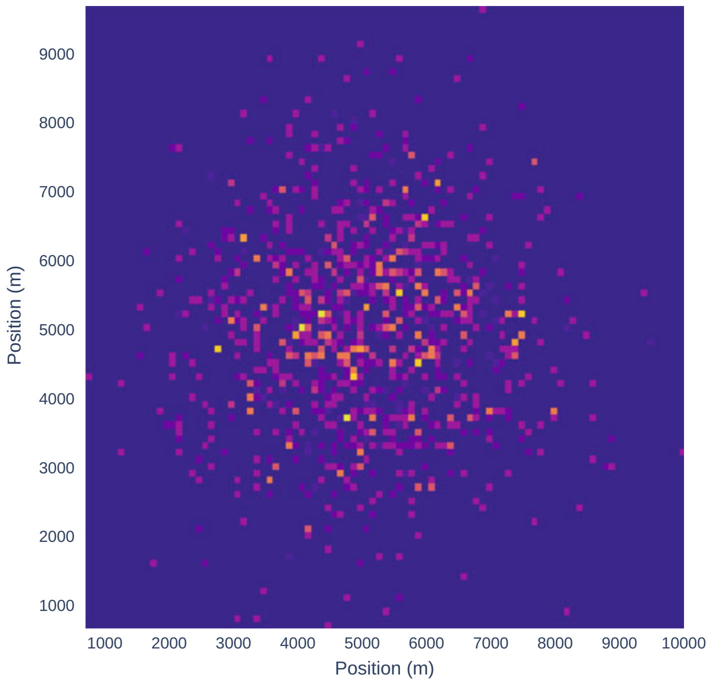


Fig. 14 User distribution

whereas their distribution follows roughly a normal distribution on both sides of the square, where the mean is set to the center (5000) and the standard deviation to 1000. This results in a population distribution that is densest at the center of our hypothetical city and decreases with higher distances. Figure 14 visualizes this distribution. Each cell in the grid represents a 100 by 100 m^2, where an empty cell is blue and a more crowded cell gets brighter. We used this distribution since it roughly resembles the distribution of a city: many people live and spend their time in the city center, while the outer areas of a city, i.e., the suburbs, are populated less densely. Regarding the number of users, we assume that realistically at most 1% of the population would use such an application. Thus, we simulated scenarios of 100 (0.1% of the population), 500 (0.5% of the population), and 1000 users (1% of the population). Finally, we modeled three different user behaviors: users sending only a few messages (3), e.g., because they are currently helping others or because they are busy doing other things during an emergency. On the other end, we modeled users sending many messages (50), e.g., because they are actively searching for people. Finally, an average user was modeled to send 10 messages. During a simulation, each user sends the specified number of messages during the simulation period of 1 hour, where the time a user sends its messages is uniformly distributed across the entire simulation time.

Table 3 Experimental configurations

Parameter	Values
Simulation time	1 hour
Area	10 km × 10 km
Seed	35,039
Repetitions per configuration	5
Base frequency	868.0 MHz
Users	100, 500, 1000
Messages per user	3, 10, 50
LoRa configurations (SF, BW, Payload)	1. SF7, 250 kHz, 222 bytes 2. SF7, 125 kHz, 222 bytes 3. SF7, 125 kHz, 51 bytes 4. SF9, 125 kHz, 51 bytes 5. SF12, 125 kHz, 51 bytes

The next parameter set refers to the LoRa parameters. For the base frequency, we used 868.0 MHz since it is predominantly and almost exclusively used in Europe. Furthermore, to test the capabilities of different LoRa settings and their effect on both the maximum transmission distance and interferences under high loads, we used five different configurations: (1) SF7 with a bandwidth of 250 kHz using a payload size of 222 bytes; (2) SF7, 125 kHz, and 222 bytes payload; (3) SF7, 125 kHz, and 51 bytes payload; (4) SF9, 125 kHz, and 51 bytes payload; and (5) SF12, 125 kHz, and 51 bytes payload. These payload sizes were chosen as they are both the maximum payload size of a LoRa packet for the given configuration (except configuration 4.) and, at the same time, also provide a good reference size for typical short messages. Table 3 summarizes these parameters.

Furthermore, each experiment was repeated five times, to cope with side effects due to unfortunate randomness, e.g., during user positioning. Finally, as the experiments require randomness for distributing the users within the simulation area and selecting sending times within the simulation time, we used a starting seed of 35,039 and incremented this number for every iteration of the 5 simulation repetitions resulting in 805 overall simulation runs.

Applicability and Limitations

Since LoRa is intended to be used as a low-bandwidth technology, one of the primary questions one needs to ask when considering feasibility is at what point will traffic saturate the network.

In Fig. 15, each plot represents one distinct simulation parameter set, as described in the previous section. Each row containing three figures shows results for different messages per user, and each column represents a different number of users. Within each figure, each bar on the *x*-axis shows a different LoRa configuration with respect to SF, bandwidth, and payload, and the *y*-axis shows the percentage of attempted

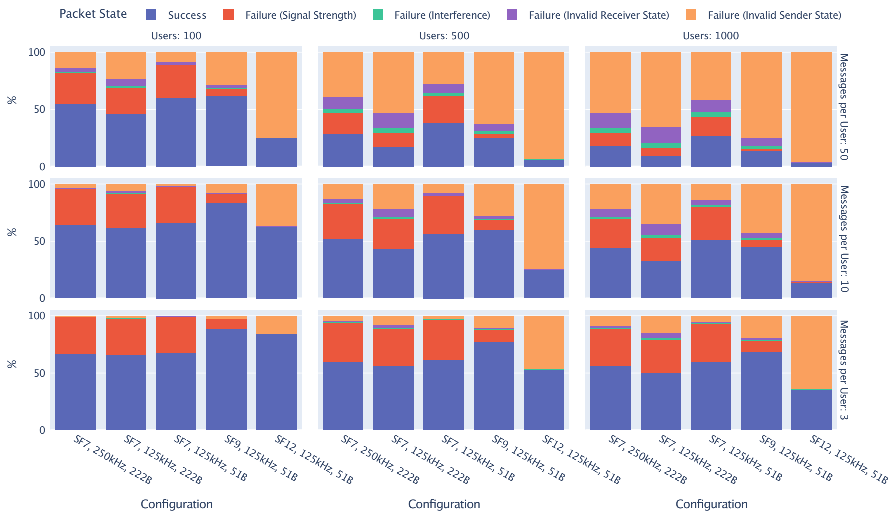


Fig. 15 Transmission results

transmissions which resulted in one of five states. Note that due to the broadcast nature of LoRa, a single transmission will lead to $n - 1$ reception events (where n is the number of users) with potentially different results:

- *Success* represents successful transmissions, i.e., a user received the packet and was able to successfully decode it.
- *Failure* (*signal strength*) represents users being unable to receive a packet because the signal attenuation due to path loss was too high.
- *Failure* (*interference*) is an unsuccessful reception due to multiple, simultaneous transmissions interfering with each other.
- *Failure* (*invalid receiver state*) means that the receiving LoRa module was in a state in which it was unable to receive the packet. This occurs since LoRa modules cannot simultaneously transmit and receive data.
- *Failure* (*invalid sender state*) is a packet reception that did not occur because the packet was not sent at all. This can be due to either one of two reasons. One possible reason is the sender being in receive mode when the packet was meant to be sent. Since LoRa modules cannot send data while they are receiving, this results in a failure. The other possible reason for this failure mode is that a LoRa module can only send a single packet at once; thus, if one tries to send a second packet while the first is still being transmitted, the sending fails.

It can be observed that with increasing load, be it due to a higher number of users, or more packets sent per user, or both, the probability that a packet will be received successfully decreases. While an increase in messages seems to impact delivery somewhat more strongly than an increase in users, the difference seems comparatively small. However, as either, or both, load factors increase, the delivery probability quickly drops to or below 50%.

Interference between senders plays virtually no part in the measured decrease of the delivery ratio. Rather, the majority of unsuccessful transmissions are due to invalid hardware states, with an invalid sender state quickly becoming the vast majority of failure conditions, followed by an invalid receiver state.

Failures due to signal attenuation do not change in any meaningful way in between load scenarios, which is to be expected since greater or smaller loads do not change physical constraints that are responsible for these failures. Only the most congested scenarios result in a significant decrease of this type of failure, but this is most likely simply due to the overwhelming effect of the invalid state failures that overpower all other conditions.

The only parameter that has a major effect on the delivery range is the spreading factor, with SF9 having already greatly decreased signal strength failures, and SF12 effectively eliminating them altogether. However, the tradeoff of higher spreading factors is obvious, since they are the most susceptible to state failures as the load increases. Comparing the three rightmost columns, which are identical except for different SFs, it becomes obvious that any spreading factor greater than seven is infeasible for our use case with respect to the ratio of successfully received packets compared to failed transmissions. Higher spreading factors increase the time it takes to send the same amount of data. Transmitting a 222 bytes large packet using SF7 and 125 kHz bandwidth takes roughly 370 ms, whereas sending 51 bytes using SF12 requires about 2800 ms airtime, i.e., 7.6 times more. This also increases the likelihood of the LoRa module being occupied in the sending state where it can neither receive packets nor accept new packets for transmission. The fact that sending takes longer in SF12 explains this observation.

Figure 16 is generated from the same data as Fig. 15 but shows the total number of events rather than the percentage-based normalized values in Fig. 15. The differences in load that are separating the simulation scenarios can be best understood when having this view on the data.

To summarize, it can be seen that the probability of successfully delivering a packet is highly susceptible to network congestion. To cope with this challenge, we need to find mitigations that allow us to prevent saturating the LoRa band, one of which we are going to present in the following section.

While congestion may be the principal issue in the way of real-world feasibility, transmission distance is another. Since LoRa messages are single-hop broadcasts, if two users are too far apart for direct transmission, they can effectively not communicate. Therefore, we have to answer the question of how far LoRa packets get depending on the experimental configuration.

Figure 17 shows the reception events in their spatial distribution, where the meaning of the colors of the packet states is the same as in the above figures. Note that in Fig. 15 (transmission ranges), the failure (invalid sender state) is not shown because packets that could not be sent do not have any location information and thus no distance associated. Furthermore, the x-axis denotes the LoRa configuration, where every group of boxes is associated with one LoRa configuration, and the

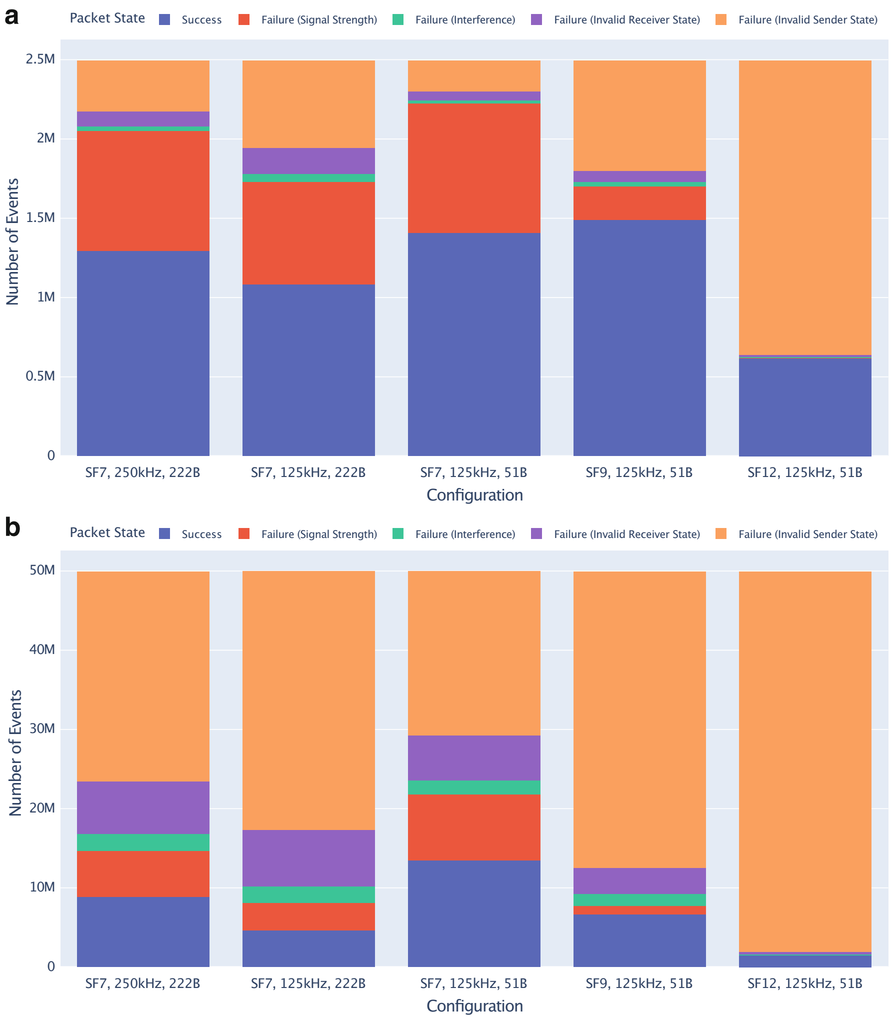


Fig. 16 Transmission results (absolute). (**a**) 10 messages per user, 500 users. (**b**) 50 messages per user, 1000 users

y-axis denotes the distance in meters that a packet traveled between sender and receiver.

One insight of the evaluation is that the general results of the distance evaluation do not depend on the load of the network, i.e., they are largely independent of how many users are sending in the network and how many packets each user sends. Thus, Fig. 17 only shows distances of packets for a single number of users (500) and a single configuration for the messages per user (10). SF, bandwidth, and payload are set as discussed previously. As can be seen in Fig. 17, the configured bandwidth

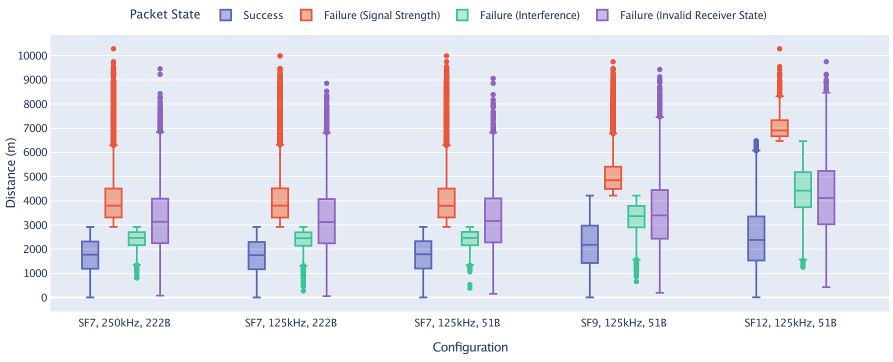


Fig. 17 Transmission ranges for 500 users and 10 messages per user

and packet payload do not affect the distance of a transmitted packet. The first three groups show SF7 but with different bandwidths and payloads. The distance of successful transmissions, however, does not change. With a lower bandwidth of 125 kHz interferences occur after a slightly shorter distance, but only visible in outliers, while the quartiles and medians do not differ significantly. The difference between the payload with SF7 and 125 kHz bandwidth with respect to interferences can also be seen in the outliers of the green boxes of groups 2 and 3. Here we can see that a smaller payload results in less interference, which is validated by the above evaluation of the transmission results. The most influential factor with respect to transmission distances is the SF. Higher SFs result in farther successful transmissions and fewer failures due to low signal strength. However, this increase in transmission range also leads to a bigger area where interference can occur, as evident in the last group of boxes representing SF12. This is explainable by the fact that higher SFs result in an increased airtime. Thus, for SF12, it is more likely that interferences occur, which is also reflected in the increased distance of interferences. The same argument also applies for an increased range of failures due to invalid receiver states. The longer the transmission takes, the higher is the probability that a user is currently in the sending state and not able to receive an incoming packet across all distances.

In summary, these results show that LoRa is able to cover a large area of a city. Users are able to reach other users within a radius of up to 2.9 km for SF7, 4.2 km for SF9, and 6.4 km for SF12. Due to interferences in the ranges around 2.4 km (SF7), 3.3 km (SF9), and 4.4 km (SF12) on the average, the usable radius is about 1.7 km, 2.2 km, and 2.5 km, respectively. However, it must be noted that the average transmission range is a result of our user distribution. With a different geographic distribution of users, the mean transmission range also changes due to interferences in different distances, but not the maximum. These results show that LoRa and especially our approach are suitable to provide emergency communications with respect to the communication range. With this transmission range, people in affected areas can communicate, coordinate themselves, and ask for help with a high chance

Table 4 Experimental configurations for additional edge-case tests

Parameter	Values
Users	100
Messages per user	1, 10, 20, . . . , 200
LoRa configurations (SF, BW, payload)	SF7, 125 kHz, 222 bytes (cf. 2.) SF9, 125 kHz, 115 bytes SF12, 125 kHz, 51 bytes (cf. 5)

to reach first responders that might not be in proximity but are still able to receive messages due to the high LoRa transmission range.

Building LoRa Communities

As a result of the issues discussed in the previous section, a single LoRa channel is of limited use to a city public, such as for the distribution of public information or emergency messages. Luckily, the different international frequency bands available for LoRa provide us with a way to implement multiple, non-interfering channels, which can be used for better practical usability in a scenario as ours. In addition to these separately usable frequencies, chirp spread spectrum modulation has the advantage that the spreading factors are orthogonal, which means that messages sent with one spreading factor do not interfere with the transmission of messages from another spreading factor.[18] Following the LoRa Alliance's definition of 8 channels in the 868 MHz band and the common spreading factors 7–12, a total of 48 independent channels are available. These channels can be used by different communities and institutions, whereby the channel distribution is either agreed upon in advance or negotiated among the users at a central coordination channel.

To get an impression of the usability of a single channel, we performed additional simulations in which the channel was used by 100 users with different message rates (1–200 messages per user and hour). For this experiment series, we used spreading factors 7, 9, and 12, and their respective maximum message lengths. A summary of the updated values used for these tests can be found in Table 4.

Figure 18 shows the experimental results of the proposed experiment in a community of 100 people. As already indicated in the previous experiments, the rate of successfully delivered messages is limited by the range, especially in smaller spreading factors. For SF7, 27.6–33.2% of the messages are lost due to low signal strength, while SF9 incurs 6.3–9.9% loss. When using spreading factor 12, the transmission time of the packets is so high that a successful transmission is no longer possible even with a low number of messages per user. The capacity limit of the channel can be derived by determining the intersection of the successful

[18] https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrp oKFf-vaF_Fkpgp5kzjiNyiAbqcpqh9qSjE

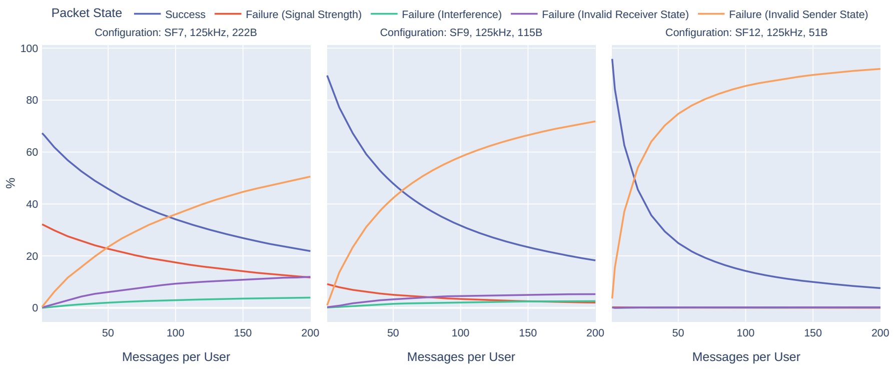


Fig. 18 Message receiving performance for different spreading factors and variable messages per user for a community of 100 users

deliveries and the transmission prevented by simultaneous reception, i.e., invalid sender state. Following this scheme for 100 users, a SF7 channel has a capacity of 90,222-bytes messages and a SF9 a capacity of around 60,115-bytes messages, and a SF12 channel is limited to around 2051-bytes messages. These metrics, alongside with the range benefits and drawbacks of individual spreading factors, can help communities to decide about a configuration to establish useful communication.

Conclusion

In this chapter, we presented an approach to facilitate long-range device-to-device communication between smartphones in crisis situations. Our approach relies on inexpensive microcontrollers with integrated LoRa hardware that we enabled to receive and forward messages via Bluetooth, Wi-Fi, or a serial connection. We developed a dedicated firmware, called *rf95modem*, to provide this functionality not only in crisis situations but also in several other applications, such as providing a communication fallback during outdoor activities, geolocation-based games, or broadcasting of local information. To illustrate the practical relevance of our approach, we implemented a novel device-to-device LoRa chat application for iOS, Android, and laptop/desktop computers. Furthermore, we integrated LoRa using *rf95modem* into the disruption-tolerant networking software DTN7. Our experimental evaluation based on real-world device-to-device LoRa transmissions in urban and rural areas, as well as scalability tests based on simulations of LoRa device-to-device usage with up to 1000 active users, showed that our approach is technically feasible and enables low-cost, low-energy, and infrastructure-less communication. All software implemented as part of our work and the results of the experimental evaluation are released with this chapter under permissive open-source licenses.

There are several areas of future work. For example, to efficiently use LoRa and its limited bandwidth in crisis scenarios, a frequency plan for users and first responders should be created. Such a plan can be integrated into the emergency communication app, and the plan could be presented to the user. Furthermore, while the presented energy evaluation provides a basic model, further measurements with the board-specific connection options should be conducted and evaluated in field tests.

Acknowledgments This research work has been funded by the Deutsche Forschungsgemeinschaft (DFG), the German Federal Ministry of Education and Research (BMBF), and the Hessian State Ministry for Higher Education, Research and the Arts (HMWK) in the National Research Center for Applied Cybersecurity ATHENE (BMBF, HMWK), the Collaborative Research Center (SFB) 1053 – MAKI (DFG), the LOEWE Research Center emergenCITY (HMWK), and the LOEWE Project Nature 4.0 (HMWK).

References

Augustin, A., Yi, J., Clausen, T., & Townsley, W. (2016). A study of LoRa: Long range & low power networks for the internet of things. *Sensors, 16*(9), 1466.

Baumgärtner, L., Gardner-Stephen, P., Graubner, P., Lakeman, J., Höchst, J., Lampe, P., Schmidt, N., Schulz, S., Sterz, A., & Freisleben, B. (2016). An experimental evaluation of delay-tolerant networking with serval. In *2016 IEEE Global Humanitarian Technology Conference (GHTC)* (pp. 70–79). IEEE.

Baumgärtner, L., Penning, A., Lampe, P., Richerzhagen, B., Steinmetz, R., & Freisleben, B. (2018). Environmental monitoring using low-cost hardware and infrastructureless wireless communication. In *2018 IEEE Global Humanitarian Technology Conference (GHTC)* (pp. 1–8). IEEE.

Berto, R., Napoletano, P., & Savi, M. (2021). A LoRa-based mesh network for peer-to-peer long-range communication. *Sensors, 21*(13), 4314.

Bor, M., Vidler, J., & Roedig, U. (2016). LoRa for the internet of things. In *Proceedings of the 2016 international conference on embedded wireless systems and networks. EWSN '16* (pp. 361–366). Junction Publishing.

Callebaut, G., Leenders, G., Buyle, C., Crul, S., & Van der Perre, L. (2019). LoRa physical layer evaluation for point-to-point links and coverage measurements in diverse environments. *arXiv preprint arXiv:1909.08300.*

Deepak, D. C., Ladas, A., Sambo, Y. A., Pervaiz, H., Politis, C., & Imran, M. A. (2019). An overview of post-disaster emergency communication systems in the future networks. *IEEE Wireless Communications, 26*(6), 132–139.

Elijah, O., Rahman, T. A., Orikumhi, I., Leow, C. Y., & Hindia, M. N. (2018). An overview of Internet of Things (IoT) and data analytics in agriculture: Benefits and challenges. *IEEE Internet of Things Journal, 5*(5), 3758–3773.

Gardner-Stephen, P. (2011). *The serval project: Practical wireless ad-hoc mobile telecommunications*. Technical report. Flinders University, Adelaide, South Australia.

Graubner, P., Lampe, P., Höchst, J., Baumgärtner, L., Mezini, M., & Freisleben, B. (2018). Opportunistic named functions in disruption-tolerant emergency networks. In *ACM international conference on computing frontiers 2018 (ACM CF 2018).* ACM.

Henderson, T. R., Lacage, M., Riley, G. F., Dowell, C., & Kopena, J. (2008). Network simulations with the ns-3 simulator. *SIGCOMM Demonstration, 14*(14), 527.

Hornbuckle, C. A. (2010). *Fractional-N synthesized chirp generator*. United States Patent US7791415B2, Semtech Corp (May 2007).

Kaufhold, M. A., Rupp, N., Reuter, C., Amelunxen, C., & Cristaldi, M. (2018). 112.Social: Design and evaluation of a mobile crisis app for bidirectional communication between emergency services and citizens. In *26th European conference on information systems: Beyond digitization – Facets of socio-technical change, ECIS 2018*.

Kayisire, D., & Wei, J. (2016). ICT adoption and usage in Africa: Towards an efficiency assessment. *Information Technology for Development, 22*(4), 630–653.

Lieser, P., Alvarez, F., Gardner-Stephen, P., Hollick, M., & Boehnstedt, D. (2017). Architecture for responsive emergency communications networks. In *2017 IEEE Global Humanitarian Technology Conference (GHTC)* (pp. 1–9). IEEE.

Liu, Y., Bild, D. R., Adrian, D., Singh, G., Dick, R. P., Wallach, D. S., & Mao, Z. M. (2015). Performance and energy consumption analysis of a delay-tolerant network for censorship-resistant communication. In *Proceedings of the 16th ACM international symposium on mobile ad hoc networking and computing* (pp. 257–266). ACM.

LoRa Alliance. (2018). *LoRaWAN regional parameters v1.0.3*. LoRa Alliance.

Magrin, D., Centenaro, M., & Vangelista, L. (2017). Performance evaluation of LoRa networks in a smart city scenario. In *2017 IEEE international conference on communications (ICC)* (pp. 1–7). IEEE.

Manoj, B. S., & Baker, A. H. (2007). Communication challenges in emergency response. *Communications of the ACM, 50*(3), 51–53.

Mekiker, B., Wittie, M. P., Jones, J., & Monaghan, M. (2021). Beartooth relay protocol: Supporting real-time application streams with dynamically allocated data reservations over LoRa. In *2021 International conference on computer communications and networks (ICCCN)* (pp. 1–9). IEEE.

Olteanu, A.-C., Oprina, G.-D., Tapus, N., & Zeisberg, S. (2013). Enabling mobile devices for home automation using ZigBee. In *2013 19th international conference on control systems and computer science* (pp. 189–195). IEEE.

Penning, A., Baumgärtner, L., Höchst, J., Sterz, A., Mezini, M., & Freisleben, B. (2019). DTN7: An open-source disruption-tolerant networking implementation of Bundle Protocol 7. In *International conference on ad-hoc networks and wireless* (pp. 196–209). Springer.

Sciullo, L., Fossemo, F., Trotta, A., & Di Felice, M. (2018). LOCATE: A LoRa-based mObile emergenCy mAnagement sysTEm. In *2018 IEEE global communications conference (GLOBECOM)* (pp. 1–7). IEEE.

Sciullo, L., Trotta, A., & Di Felice, M. (2020). Design and performance evaluation of a LoRa-based mObile emergenCy mAnagement sysTEm (LOCATE). *Ad Hoc Networks, 96*, 101993.

Tan, M. L., Prasanna, R., Stock, K., Hudson-Doyle, E., Leonard, G., & Johnston, D. (2017). Mobile applications in crisis informatics literature: A systematic review. *International Journal of Disaster Risk Reduction, 24*, 297–311.

Wixted, A. J., Kinnaird, P., Larijani, H., Tait, A., Ahmadinia, A., & Strachan, N. (2016). Evaluation of LoRa and LoRaWAN for wireless sensor networks. In *2016 IEEE SENSORS* (pp. 1–3). IEEE.