LoRa-based Device-to-Device Smartphone Communication for Crisis Scenarios

Jonas Höchst
University of Marburg, Germany
Technical University of Darmstadt, Germany
hoechst@informatik.uni-marburg.de

Lars Baumgärtner
Technical University of Darmstadt, Germany
baumgaertner@cs.tu-darmstadt.de

Franz Kuntke
Technical University of Darmstadt, Germany
kuntke@peasec.tu-darmstadt.de

Alvar Penning
University of Marburg, Germany
penning@informatik.uni-marburg.de

Artur Sterz
University of Marburg, Germany
Technical University of Darmstadt, Germany
sterz@informatik.uni-marburg.de

Bernd Freisleben
University of Marburg, Germany
Technical University of Darmstadt, Germany
freisleb@informatik.uni-marburg.de

ABSTRACT

In this paper, we present an approach to facilitate long-range device-to-device communication via smartphones in crisis scenarios. Through a custom firmware for low-cost LoRa capable micro-controller boards, called *rf95modem*, common devices for end users can be enabled to use LoRa through a Bluetooth, Wi-Fi, or serial connection. We present two applications utilizing the flexibility provided by the proposed firmware. First, we introduce a novel device-to-device LoRa chat application that works a) on the two major mobile platforms Android and iOS and b) 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. The firmware, the device-to-device chat application, the integration into DTN7, as well as the experimental evaluation code fragments are available under permissive open-source licenses.

Keywords

LoRa, Disaster Communication, Device-To-Device Communication,

INTRODUCTION

The communication technologies developed and deployed in the last decades are integral parts of our daily life and are used by mobile phones, computers, or smart applications in homes and cities. Today's smartphones, however, highly depend on the availability of telecommunication infrastructures, such as Wi-Fi or cellular technology (e.g., 3G, LTE, or the upcoming 5G standard). However, there are situations in which either no communication infrastructure is available or only at a high cost, e.g., in remote areas (Gardner-Stephen 2011), in the agricultural sector (Elijah et al. 2018), as a result of disasters (Manoj and 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 and 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.

CoRe Paper – Usability and Universal Design of ICT for Emergency Management
Proceedings of the 17th ISCRAM Conference – Blacksburg, VA, USA May 2020
Amanda Lee Hughes, Fiona McNeill and Christopher Zobel, eds.

We propose to use LoRa wireless technology as a communication enabler in such situations. LoRa (*Lo*ng-*Ra*nge) is a long range and low power network protocol designed for the Internet of Things 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 paper, 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 supporting 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.

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.

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 a) Android and iOS smartphones and b) 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 performing energy measurements of multiple devices.
- The presented *rf95modem* software[1], the device-to-device chat application[2], the integration into DTN7[3] and the experimental evaluation code fragments[4] are freely available.

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 to decode 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%.

[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

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, Penning, et al. 2018 proposed to use LoRa for environmental monitoring. In the included LoRa evaluation, ranges of 4.6 to 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 a line-of-sight and 1 km in a 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 towards 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, Fossemo, 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, Trotta, 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.

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.

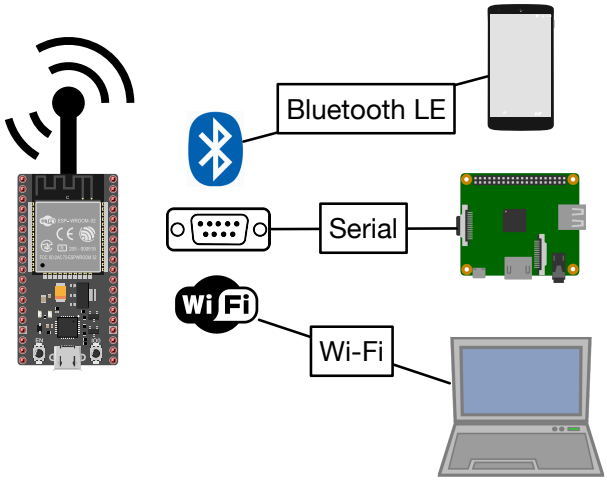


Figure 1. ESP32-based modem board and its connection options for smartphones, single-board computers, and laptops.

Modem Firmware

Figure 1 shows how different devices can be connected to a modem board. There are several commercial off-the-shelf micro-controller 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[5], Adafruit's Feather 32u4 and M0 boards [6] or the Heltec Automation WiFi LoRa 32 and Wireless Stick (Lite)[7]. 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.

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, 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.

[5] http://www.lilygo.cn/pro.aspx?FId=t3:50003:3, Xing Yuan Electronic Technology Co., Ltd., LongGang, Shenzhen, China
[6] https://www.adafruit.com/product/3178, Adafruit Industries, LLC, 150 Varick Street, New York 10013, USA
[7] https://heltec.org/proudct_center/lora/lora-node/, Heltec Automation, Longtan Industrial Park, Chengdu, China

Disruption-tolerant Networking

For crisis scenarios, DTN is a technology to enable infrastructureless communication using an emergency infrastructure in conjunction with existing devices of users (Baumgärtner, Gardner-Stephen, 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, the implementations of the *rf95modem* firmware, the device-to-device messaging application, and the integration into disruption-tolerant networking software are presented.

Modem Firmware

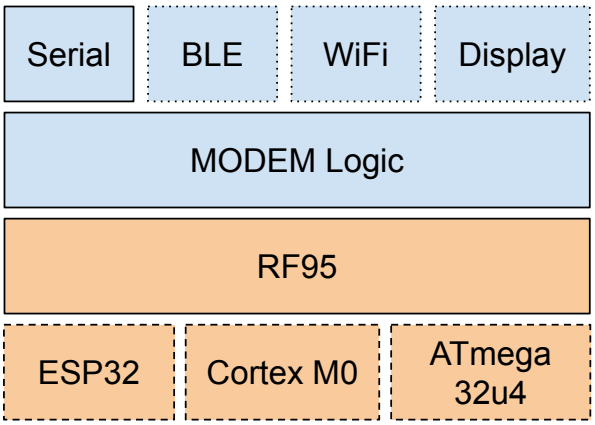
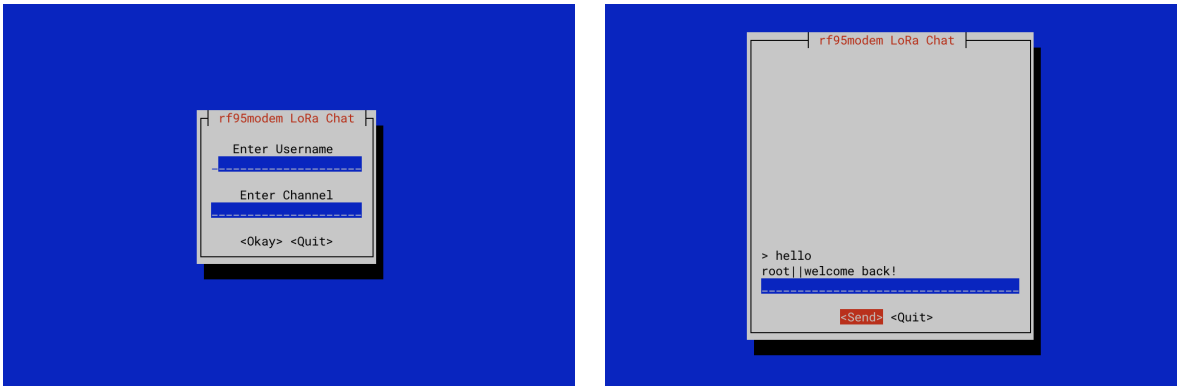


Figure 2. Overview of the *rf95modem* architecture.

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 Figure 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 WiFi 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 WiFi 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



(a) Login screen **(b) Chat interface**

Figure 3. Console-based *rf95modem* LoRa chat example.

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. 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 Figure 3[8]. Second, for the mobile version of the application (BlueRa), we used the Flutter UI toolkit[9]. 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.

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 Figure 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, as well as the read and write characteristics for the serial communication

[8] https://github.com/gh0st42/rf95modem-rs
[9] https://flutter.dev

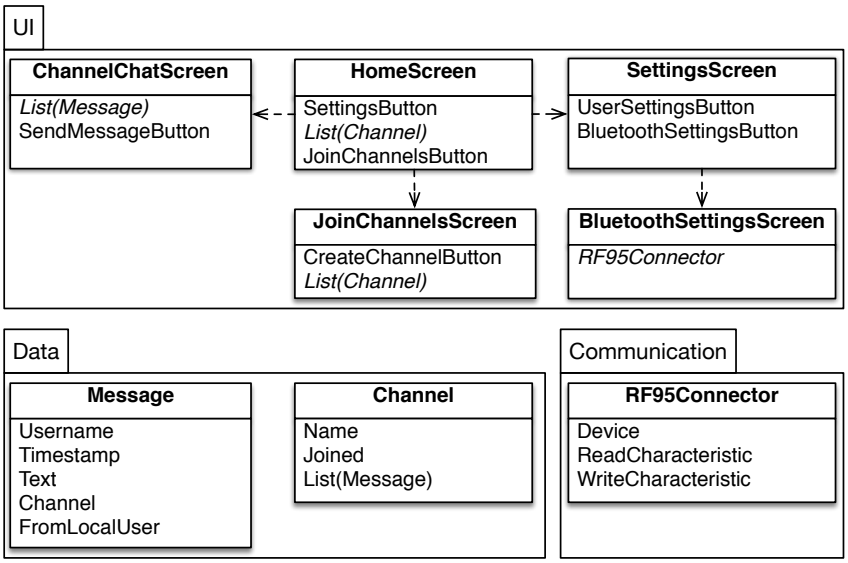


Figure 4. Overview of the components of the app.

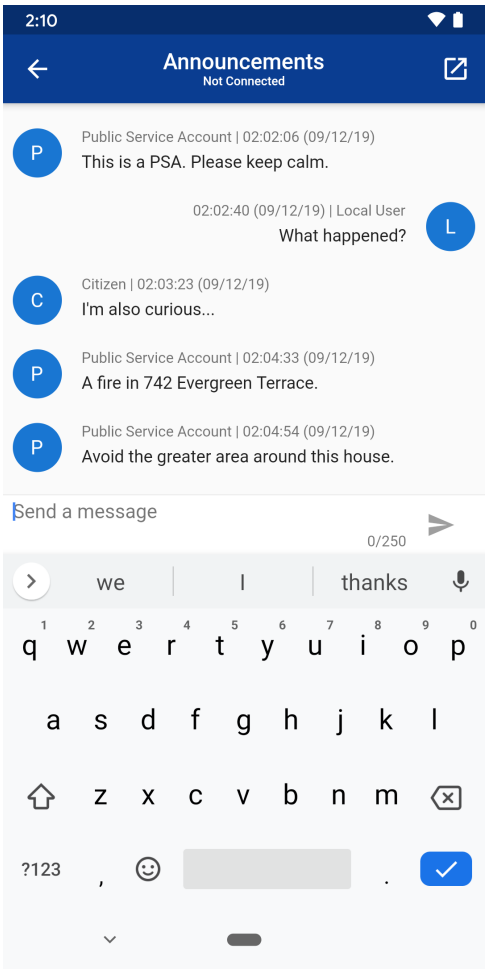


Figure 5. Screenshot of the chat screen for the announcements channel.

service. Additionally, this module also implements data and message handling. When sending a new message, all required data is 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 used a simple communication protocol for sending and receiving messages. It consists of the channel name, a user name, an optional location and the actual message separated by vertical bars. Following this simple protocol design, it is possible for any communication device to communicate with the app, regardless of the capabilities and available serialization libraries like JSON, CBOR, or Protocol Buffers.

Disruption-tolerant Networking

To use LoRa in disruption-tolerant networking, we have extended the DTN7 implementation[10] 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[11], 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.

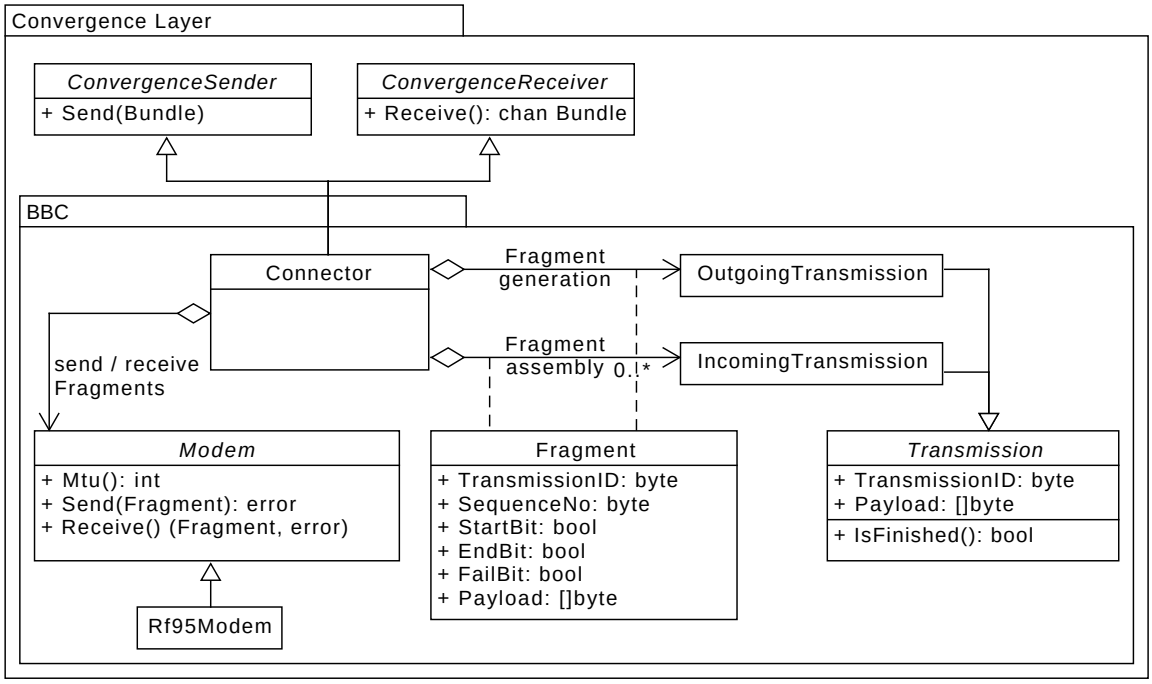


Figure 6. Simplified implementation model of the Bundle Broadcasting Connector.

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 Figure 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

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

is initialized with a random identifier, which is then incremented for each transmission. The payload is the `xz`[12] 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.

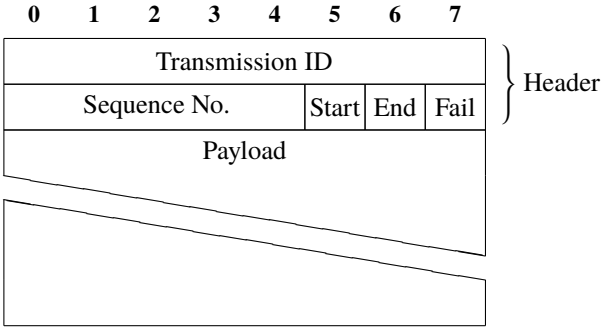


Figure 7. Protocol specification of a fragment.

The network protocol specification of a fragment is shown in Figure 7. A fragment itself consist 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 incremented sequence number of its predecessor. Thus, lost 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 fail 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 fail 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 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 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 announcement 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.

[12] https://tukaani.org/xz/format.html

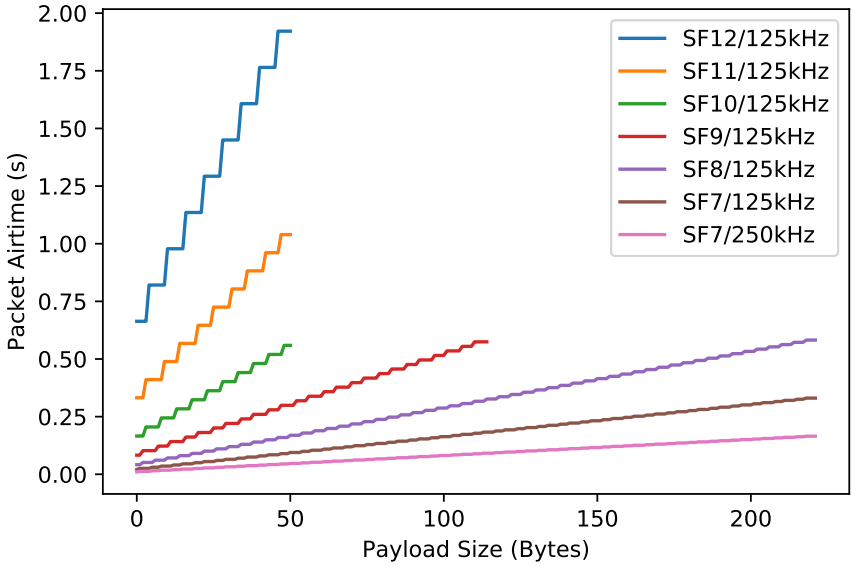


Figure 8. Exemplary packet airtime in different LoRa profiles.

The motivation for scenario (a) are 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 meter above the ground, in order to model realistic usage in device-to-device scenarios.

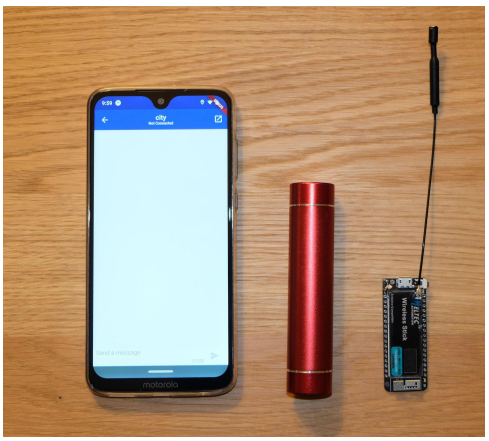


Figure 9. Mobile station: smartphone, power bank, and Heltec wireless stick.

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, 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.

Scenario	Mode	Maximum Distance
City	(a) Medium Range	1.09 km
	(b) Long Range	2.89 km
Rural Area	(a) Medium Range	1.31 km
	(b) Long Range	1.64 km

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

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 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 Figure 11, the successful LoRa transmission of the rural area are presented. As excepted, 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 meters of forested area.

In Figure 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 meta-data 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 2^1 to 2^{11}. For this purpose, the

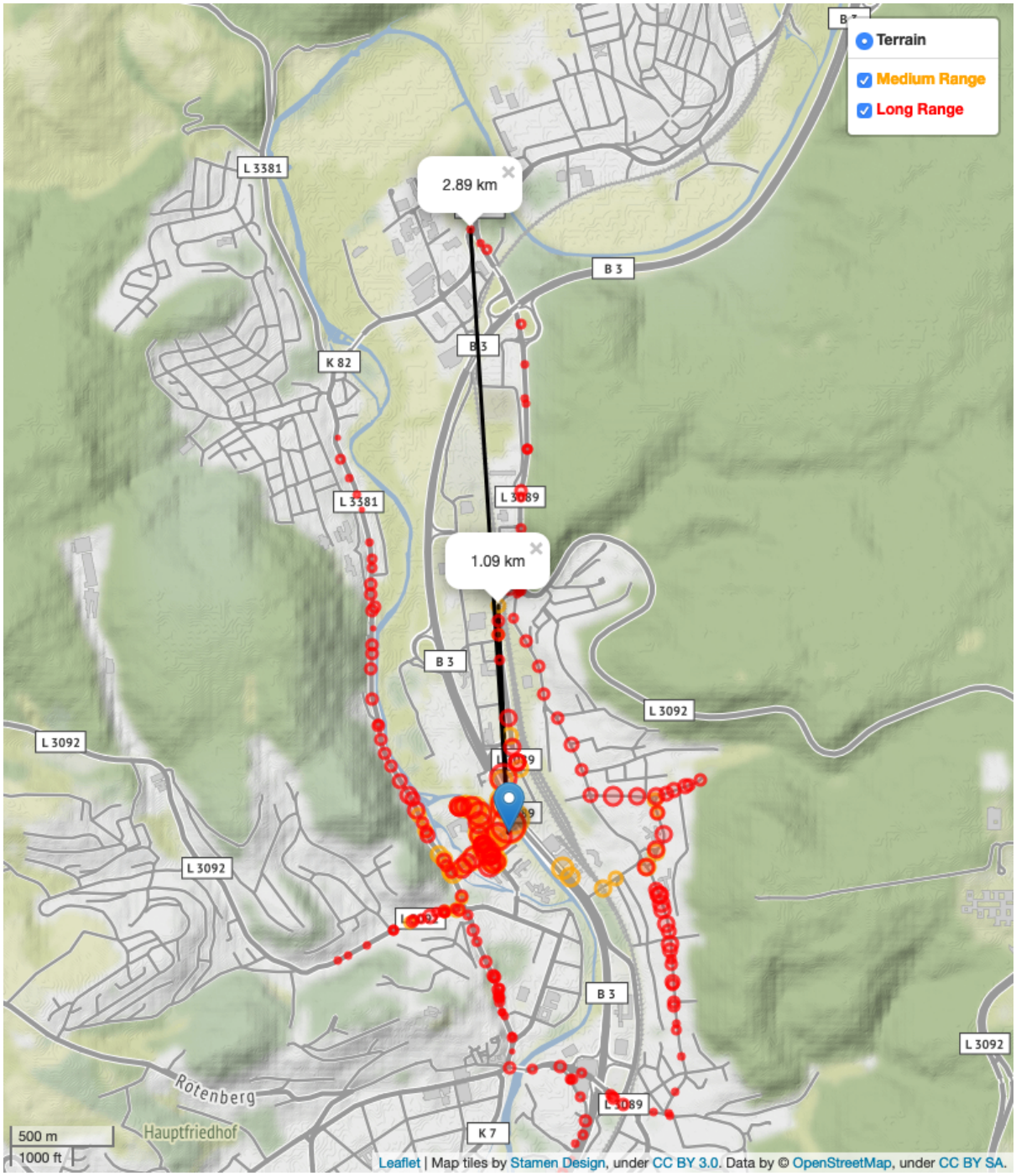


Figure 10. Successful LoRa transmissions in the city area.

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

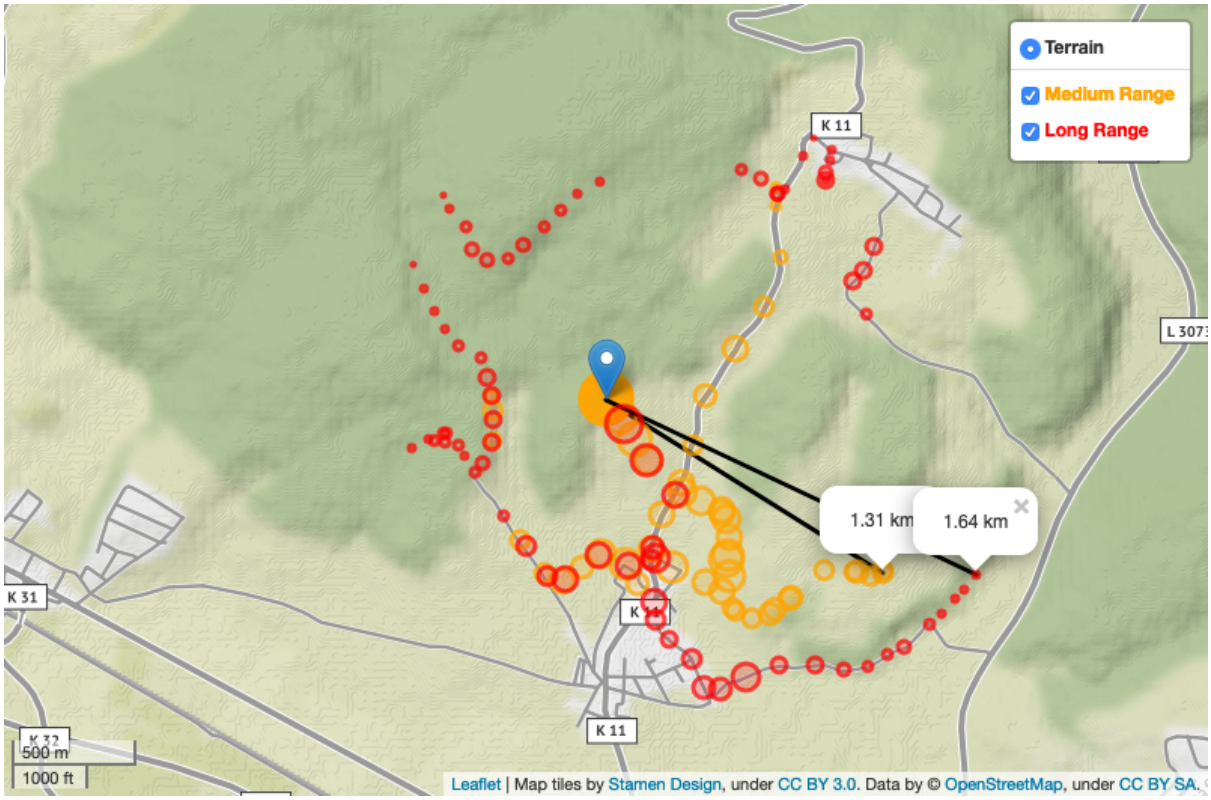


Figure 11. Geo-positions of successful LoRa transmissions in a rural area.

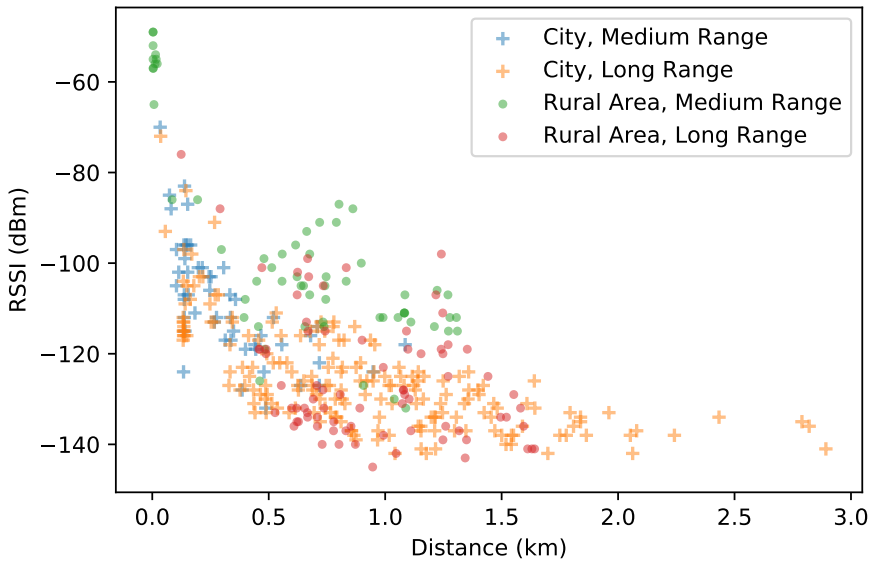


Figure 12. Received Signal Strength Indicator in relation to transmission distance in the proposed device-to-device scenario.

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 meta-data 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, n_0, n_1, and n_2. n_1 was positioned in the midpoint,

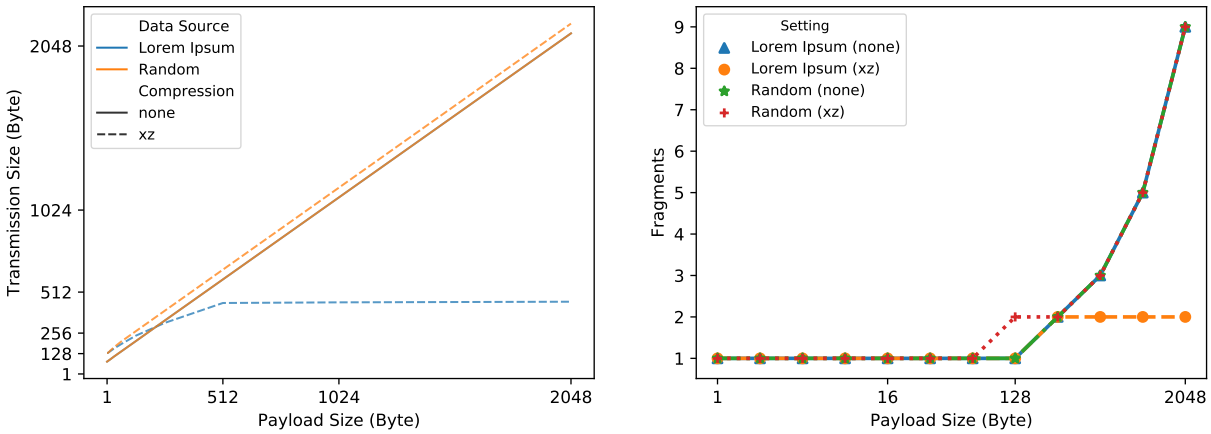


Figure 13. Total transmission size and amount of fragments for different payloads.

further n_0 and n_2 were not supposed to have direct contact. Outgoing from n_0, packets were sent addressed to n_2. These should be transferred from n_0 to n_1 first and forwarded from n_1 to n_2 afterwards. 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 round trip time took 1.7 seconds from initiating the transmission to receiving the acknowledgement 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 paper 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.

Board	Receiving	Sending	Deep Sleep	Additional Features	Price
TTGO T-Fox 27 dB	392 mW	1,771 mW	61 mW	WiFi, BLE, OLED, RTC	20 €
TTGO T-Fox 20 dB	400 mW	902 mW	61 mW	WiFi, BLE, OLED, RTC	20 €
TTGO LORA ESP32	404 mW	782 mW	68 mW	WiFi, BLE	15 €
TTGO LORA32 V2.0	393 mW	689 mW	57 mW	WiFi, BLE, OLED, SD	20 €
TTGO LORA32 V2.1_1.6	387 mW	785 mW	57 mW	WiFi, BLE, OLED, SD	20 €
Heltec Wireless Stick	391 mW	923 mW	76 mW	WiFi, BLE, OLED	12 €
Adafruit Feather 32u4 LoRa	72 mW	648 mW	49 mW	-	35 €
Adafruit Feather M0 LoRa	95 mW	697 mW	<1 mW	-	35 €
TTGO T-Beam v0.7	723 mW	1,125 mW	393 mW	WiFi, BLE, GPS	20 €

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

The energy consumption was measured using an ODROID Smart Power Meter[13] 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 to 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 microUSB connection, which requires a transformation to the voltage required by the microprocessors. 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

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

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.

CONCLUSION

In this paper, we presented an approach to integrate modern emergency and rural communications technology into existing devices. In particular, we presented a novel, freely available and open source modem firmware for LoRa-enabled MCUs, called *rf95modem*. Using such a companion device with our firmware, a novel device-to-device LoRa chat application for iOS, Android, and laptop/desktop computers was created. An integration of LoRa into the disruption-tolerant networking software DTN7 was presented. LoRa as a protocol for such use cases was discussed, and the device-to-device chat application, as well as the DTN7 integration were experimentally evaluated. The evaluation showed that our approach is technically feasible and enables low-cost, low-energy, and infrastructure-less communication. All software implemented for this paper and the results of the experimental evaluation are released with this paper 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.

ACKNOWLEDGEMENT

This research work has been funded by the Deutsche Forschungsgemeinschaft (DFG), the German Federal Ministry of Education and Research (BMBF), and the Hessen 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., and Townsley, W. (2016). "A study of LoRa: xLong range & low power networks for the internet of things". In: *Sensors* 16.9, p. 1466.

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

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

Bor, M., Vidler, J., and Roedig, U. (2016). "LoRa for the Internet of Things". In: *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*. EWSN '16. Graz, Austria: Junction Publishing, pp. 361–366.

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

Deepak, D. C., Ladas, A., Sambo, Y. A., Pervaiz, H., Politis, C., and Imran, M. A. (2019). "An overview of post-disaster emergency communication systems in the future networks". In: *IEEE Wireless Communications* 26.6, pp. 132–139.

Elijah, O., Rahman, T. A., Orikumhi, I., Leow, C. Y., and Hindia, M. N. (2018). "An overview of Internet of Things (IoT) and data analytics in agriculture: Benefits and challenges". In: *IEEE Internet of Things Journal* 5.5, pp. 3758–3773.

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

Graubner, P., Lampe, P., Höchst, J., Baumgärtner, L., Mezini, M., and Freisleben, B. (May 2018). "Opportunistic named functions in disruption-tolerant emergency networks". In: *ACM International Conference on Computing Frontiers 2018 (ACM CF 2018)*. Ischia, Italy: ACM.

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

Kaufhold, M. A., Rupp, N., Reuter, C., Amelunxen, C., and 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. and Wei, J. (2016). "ICT adoption and usage in Africa: Towards an efficiency assessment". In: *Information Technology for Development* 22.4, pp. 630–653.

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

Liu, Y., Bild, D. R., Adrian, D., Singh, G., Dick, R. P., Wallach, D. S., and 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.

LoRa Alliance (2018). "LoRaWAN Regional Parameters v1.0.3". In: *LoRa Alliance: Fremont, CA, USA*.

Manoj, B. S. and Baker, A. H. (2007). "Communication challenges in emergency response". In: *Communications of the ACM* 50.3, pp. 51–53.

Olteanu, A.-C., Oprina, G.-D., Tapus, N., and Zeisberg, S. (May 2013). "Enabling mobile devices for home automation using ZigBee". In: *2013 19th International Conference on Control Systems and Computer Science*, pp. 189–195.

Penning, A., Baumgärtner, L., Höchst, J., Sterz, A., Mezini, M., and Freisleben, B. (2019). "DTN7: An open-source disruption-tolerant networking implementation of Bundle Protocol 7". In: *International Conference on Ad-Hoc Networks and Wireless*. Springer, pp. 196–209.

Sciullo, L., Fossemo, F., Trotta, A., and Di Felice, M. (Dec. 2018). "LOCATE: A LoRa-based mObile emergenCy mAnagement sysTEm". In: *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7.

Sciullo, L., Trotta, A., and Di Felice, M. (2020). "Design and performance evaluation of a LoRa-based mobile emergency management system (LOCATE)". In: *Ad Hoc Networks* 96, p. 101993.

Tan, M. L., Prasanna, R., Stock, K., Hudson-Doyle, E., Leonard, G., and Johnston, D. (2017). "Mobile applications in crisis informatics literature: A systematic review". In: *International Journal of Disaster Risk Reduction* 24.November 2016, pp. 297–311.

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