Demonstration of a library prototype to build LoRa mesh networks for the IoT

Joan Miquel Solé, Sergi Miralles Nogués, Roger Pueyo Centelles, Felix Freitag
Department of Computer Architecture. Technical University of Catalonia. Barcelona, Spain
{j.miquel.sole, s.miralles}@estudiantat.upc.edu, {r.pueyo, f.freitag}@upc.edu

Abstract—LoRa has become popular in the Internet of Things (IoT) domain as a Low Power, Wide Area Network (LPWAN) radio technology providing low-power and long-range communication. In a typical IoT application, the LoRaWAN architecture is applied, where LoRa-enabled sensor nodes communicate their data to a gateway, which over the Internet sends these data to a cloud-based service for further processing. LoRa however can also be used standalone for the communication between LoRa end nodes. In this demo paper we show the implementation of a library called LoRaMesher, which enables an application running on a LoRa-enabled IoT device to communicate over a mesh network through a distance vector routing protocol with other LoRa nodes.

Index Terms—LoRa, Mesh networks.

I. INTRODUCTION

LoRa is a wireless communication technology designed for the Internet of Things (IoT). It provides long range communication, with links of several km between devices, low power consumption and low data rate. Its available configuration parameters, like the Spreading Factor (SF), can extend its range, although at the expense of lower data rates [1].

LoRaWAN is the dominant architecture for IoT deployments using LoRa [2]. There, LoRa-equipped sensor nodes (i.e., end devices) transmit their data to nearby gateways. The gateways, in turn, forward these messages to a network server over an Internet connection. This way, data from the nodes reach an application server, where they can be further processed by higher level IoT services. An example of a large, public LoRaWAN deployment is The Things Network [1].

Researchers have made several proposals to extend the LoRaWAN architecture by means of multi-hop, mesh and routing protocols [3] and envisioning new application scenarios if the communication between end nodes were available [4]. A few works are deployed in real field environments, such as Ebi et al.’s [5], which proposed a synchronous LoRa mesh network, and Meshtastic [6], an open-source project to establish LoRa networks for short text messages. Neither of both, however, uses a routing protocol to forward messages.

In this paper, we demonstrate a library which implements a routing protocol for LoRa mesh networks. The implementation is based on the distance-vector routing protocol designed in [7], where each LoRa node maintains a routing table about the network that is periodically updated among the directly connected neighbours.

II. PROTOTYPE IMPLEMENTATION

The LoRaMesher library 2 implements a distance-vector routing protocol for exchanging messages among LoRa nodes. For the interaction with the LoRa radio chip, we leverage RadioLib 3, a versatile communication library which supports the SX1276 LoRa series module available on the hardware we used, among others.

In our LoRa mesh network, nodes examine all the received LoRa messages. A specific task determines whether a message is a data or a routing packet. Besides, if the data packet’s destination not the node itself and it has to be routed to another node, a path towards the destination is determined. We use FreeRTOS 4 to implement task handlers for the receiver, sender, packet processing and user process tasks.

The receiver task has to be occupied the least time. Otherwise, if two packets arrive consecutively, one may be lost because of the other being processed in the reception routine. Therefore, we implemented a packet queue class. This allows the receiver routing to encapsulate the packet and add it to the received packets queue, to be processed when the microcontroller is available to do it.

Data packets include a small header of 8 bytes added by the LoRaMesher library to the packet (source: 2 bytes, destination: 2 bytes, via: 2 bytes, type: 1 byte, payload size: 1 byte). Therefore, the maximum payload in the LoRa packet is 248 bytes (256 - 8 bytes). Routing packets require only 6 bytes (source: 2 bytes, destination (broadcast address): 2 bytes, type: 1 byte, payload size: 1 byte) such that 250 bytes are available to send the content of the node’s routing table.

III. EXPERIMENTATION WITH THE LIBRARY

In the following, we use four TTGO T-Beam ESP32 boards flashed with the LoRaMesher library. These devices are based on the ESP32 System on a Chip (SoC) and feature an SX1276 LoRa transceiver (Fig. 1). We run an experiment in controlled laboratory conditions, where the boards were placed close to each other (i.e., within a 1 hop distance).

1https://www.thethingsnetwork.org/
2https://github.com/LoraMesher/LoraMesher
3https://github.com/gromes/RadioLib
4https://www.freertos.org/
For the experiment, each of the boards runs an application that sends and receives data packets. In addition, routing packets are sent by the library between neighbours, keeping the routing tables updated. The application randomly chooses one of the other devices in the network and sends a data message holding a numeric counter value.

Figure 2 shows a trace obtained from the serial port monitor connected to a LoRa node with id 0x63AC. We have enabled log information, both from the application and the LoRaMesher library, to show the operation of the nodes on both layers. First, we can see the construction of the routing table in the node. Moreover, in the second packet received, which is a routing packet, the library finds a better route to node 0xC5FC with a metric of 1. After receiving the last hello packet, the node sends to the device with id 0xDE9C a data packet of 12 bytes with a numeric value. The last packet received by node 0x63AC is a data packet from node 0xDE9C. It contains the counter value 1 in the payload (see the last line of the trace). Moreover, the trace shows another data packet that has been received by the node 0x63AC with a destination 0xC5FC, telling that this data packet is not for this node. It can be seen that the data packet is deleted by the node. This also means that node 0x63AC is not in the data packet’s via field (i.e., node 0x63AC is not expected to route this packet towards its destination).

IV. DEMONSTRATION

The demonstration of the experimentation with the LoRaMesher library will show three aspects:

1) Application development: we will show how LoRaMesher is used by applications. For this, we will explain and analyse an application running on several nodes.

2) Library development: we will look deeper at the internal implementation of the library and will explain the decisions made, and their effect on the performance of the library.

3) Future work: we will point to our on-going research for extending the LoRaMesher library.

REFERENCES


