A system for monitoring and querying LoRa mesh network nodes

Jack Griffiths*, Joan Miquel Solé*, Felix Freitag*, Leandro Navarro*, Mennan Selimi§,
* Computer Architecture Department, UPC BarcelonaTech. Spain
§ Max van der Stoel Institute, South East European University. North Macedonia

***Abstract*—LoRa mesh networks have recently gained an increasing interest to extend traditional LoRaWAN-based IoT applications. The mesh technology introduces the capacity of multi-hop and node-to-node communication in LoRa networks. This more sophisticated network service delivered by LoRa mesh networks compared to LoRaWAN raises the need for network management, to understand the situation of the different nodes that operate as routers and hosts of applications. In this paper, we present a monitoring solution for a LoRa mesh network that not only passively collects data about the nodes for visualization, but also allows Internet services to actively send queries to any network node to ask for its values. We show a proof of feasibility of the monitoring system with the deployment of real IoT boards and operational services.**

I. Introduction

LoRa is a popular technology for Low Power Wide Area Network (LPWAN) applications. Often, these Internet of Things (IoT) applications are implemented using the LoRaWAN architecture [1]. Typically, the LoRa end nodes acquire sensor values which, using the LoRaWAN architecture, are sent to a one-hop distant gateway node with an Internet connection. Although LoRaWAN has a downlink capacity, it is not possible to make a symmetric communication between uplink and downlink. Thus, data transfer in LoRaWAN in practice is primarily unidirectional, from the sensor nodes to the gateway and from there to the cloud.

LoRa mesh networks have been proposed in recent years [2], including routing protocols for LoRa mesh networks such as in [3]. Nodes can connect with each other and data can be sent over multiple hops to its destination. Also, nodes can have the role of routers or focusing on the running of applications [4]. Different scenarios are possible where an IoT application can run only within the LoRa mesh networks, or span over gateways to Internet services.

The network monitoring needs of LoRa mesh networks and LoRaWAN are different. In LoRaWAN, a typical end node does not provide a network service to other end nodes. The connection between the end node and the gateway is only active in the short time slots when the end nodes transmit a sensor value. Therefore, from the perspective of monitoring, the LoRaWAN end node shows its status to an Internet service by the expected periodic sending of sensor values.

Differently, in LoRa mesh networks, there are monitoring needs since a network service is established which enables sending packets from a source to a destination within the LoRa mesh network. The performance of that network service, such as latency and throughput, can be affected by the status of each node, e.g. the number of packets accumulated in the node's internal queues or link quality. Therefore, for LoRa mesh networks, a monitoring system can be useful both to validate the network's correct performance under permanent operation, and to be able, from experimental data, to adjust the configuration of parameters, e.g. Spreading Factor (SF), amount of traffic, or maximum LoRa packet size.

We consider a Lora mesh network as illustrated in Figure 1. Nodes can forward messages to their neighboring nodes and a message can reach any destination in the LoRa mesh network. Two types of nodes are possible, nodes acting as routers only and nodes which in addition host applications. Applications on a node may communicate with instances on other nodes. If a node is connected to a Wifi access point, it also operates as a gateway, and bidirectional communication between the LoRa mesh network nodes and the Internet can be enabled.

In this paper, we present a system to monitor LoRa mesh network nodes from an Internet service. The application consists of two parts: One part is integrated into the LoRa mesh nodes. That component reads the node data, responds to queries, and sends monitoring data. The other part is the Internet-based service. It does data visualization and the data query.

The contribution consists of a proof of feasibility. We detail our implementation to facilitate the reproduction by others. We show in some examples how the chosen solution is applied to understand the network situation and node changes. The provided tool can not only be useful to monitor deployed networks but also to collect metrics for research of systems in real environments.

II. Solution design

The monitoring spans from Internet-based services to pieces of code on the IoT nodes. There must be a communication capacity between LoRa mesh network nodes to Internet services to transmit monitoring data.

Microcontroller-based IoT nodes typically contain the code of only one application, such as the stack for measuring a sensor value and sending it in a LoRa packet. In our case, we consider two applications or apps to be hosted on the node. One is for the periodic sending of monitoring data, and the

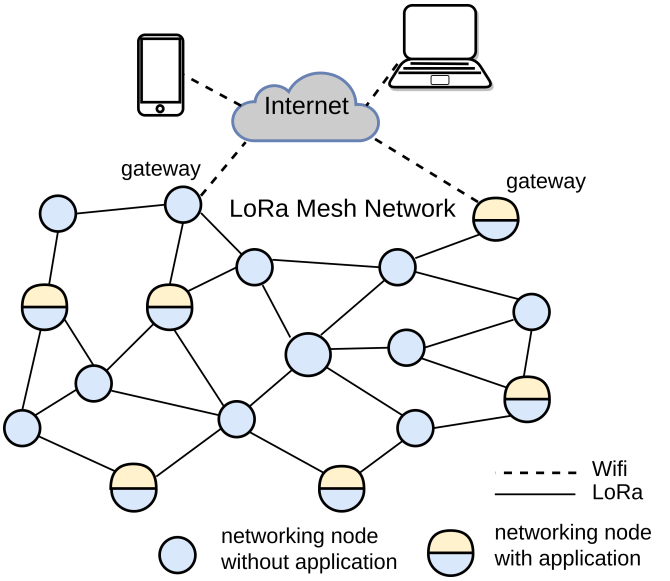

Figure 1. LoRa mesh network with router and application nodes.

other is to respond to queries with the sending of the requested data.

The monitoring app consists of an application that periodically sends messages with the node's relevant information. Examples of the content of these messages might be the node's routing table and the number and type of services running on the board. The data is communicated from a remote node in the LoRa mesh network to a gateway node and from there via an MQTT broker to an Internet service.

The query app serves a similar purpose as the monitoring app, to obtain information from a node. The main difference is that this app will only send messages to reply to queries received from the Internet-based service. Instead of sending monitoring messages periodically, it will only send data after receiving a query message asking for a specific piece of information.

The MQTT broker serves as the backbone of the communication between the two apps and the Internet service. The integration that the MQTT broker provides is crucial for the proposed solution as it will allow for the connection of the service in the Internet with the service in the LoRa mesh network, therefore enabling the bidirectional communication between the Internet service and the LoRa mesh nodes.

Regarding the Internet-based service, it is required to visualize the information sent by the apps. Functionalities include being able to graph out the information received, allowing the user to view the information in real-time, and query a node for specific pieces of information.

III. Implementation

A. LoRa mesh network solution

To implement LoRa mesh networks, only a few open-source libraries are available. For the proposed solution, we use

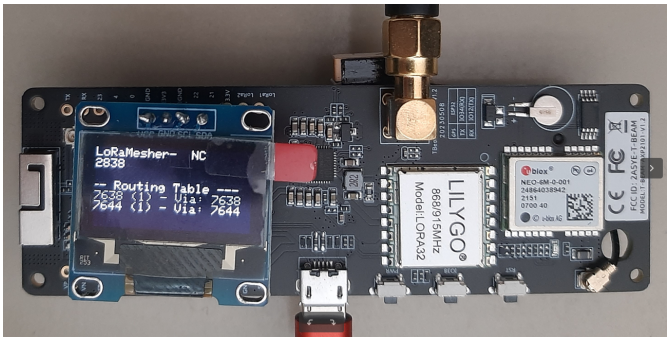

Figure 2. T-Beam boards with running LoRaMesher library.

LoRaMesher [5][1], which we have developed based on [6].

LoRaMesher is implemented with FreeRTOS and compiles for ESP32-based microcontroller boards. The board that we use for LoRaMesher is the T-Beam which has an ESP32 System on a Chip and features an SX1276 LoRa transceiver.

LoRaMesher enables routing in the LoRa mesh network by applying a proactive distance vector protocol. A node maintains a routing table that contains the nodes in the LoRa mesh network. Figure 2 illustrates a routing table shown in a node's OLED display.

The basic network service of LoRaMesher is the sending of LoRa packets to a destination over a routed network. For the development of the monitoring and query applications, we use LoRaMesher integrated into the LoRaChat implementation. We have developed LoRaChat to support the networking of multiple applications within a LoRa mesh network[2].

B. Monitoring and query application

LoRaChat apps are composed of three main files:

- `app.cpp`. Contains the main functions of the app, as well as the send and receive LoRaMesher messages functions.
- `app.h`. Header file to declare the functions and variables used in the app.cpp file.
- `appMessage.h`. Child of the LoRaChat DataMessage class. The contents of the app's LoRa message are defined.

For each app, an app port has to be specified. This is done in the DataMessage class (Figure 3). Defining the app port in the DataMessage class allows the apps to send and receive messages with an identifier.

Apps have to be enabled in the file `config.h`. In the case of a periodic task, its periodicity has to be indicated (Figure 4). file:

Finally, the apps have to be initialized in the `main.cpp` file. To do this the lines shown in Figure 5 are used.

Instances of the apps are created and initialized for their use. To add the apps to the LoRaMesher message service to be able to send and receive messages, the code shown in Figure 6 is added also to the `main.cpp` file:

Now that the apps are recognized by the library, the next step is to modify the specific `app.cpp` and `appMessage.h`

[1]https://github.com/LoRaMesher/LoRaMesher
[2]https://github.com/Jaimi5/LoRaChat

```
enum appPort: uint8_t {
    LoRaChat = 1,
    BluetoothApp = 2,
    WiFiApp = 3,
    GPSApp = 4,
    WalletApp = 5,
    CommandApp = 6,
    LoRaMesherApp = 7,
    MQTTApp = 8,
    SimApp = 12,
    LedApp = 13,
    ...
    QueryApp = 41,
    MonitoringApp = 42
};
```

Figure 3. Adding in `dataMessage.h` the appPorts for the new Query and Monitoring apps.

```
//Monitoring configuration
#define MONITORING_ENABLED
#define MONITORING_UPDATE_DELAY 90000 //ms
//QUERY
#define QUERY_ENABLED
```

Figure 4. Enabling the monitoring and query apps in `config.h`.

```
#ifdef QUERY_ENABLED
#pragma region Query
#include "query/query.h"

Query& query = Query::getInstance();

void initQuery() {
    query.init();
}
#endif

#ifdef MONITORING_ENABLED
#pragma region Monitoring
#include "monitoring/monitoring.h"

Monitoring& monitoring = Monitoring::getInstance();

void initMonitoring() {
    monitoring.init();
}
#endif
```

Figure 5. Initializing the monitoring and query apps in `main.h`.

```
#ifdef QUERY_ENABLED
    manager.addMessageService(&query);
    ESP_LOGV(TAG, "Query service added to manager");
#endif

#ifdef MONITORING_ENABLED
    manager.addMessageService(&monitoring);
    ESP_LOGV(TAG, "Monitoring service added to
        manager");
#endif
```

Figure 6. Adding the monitoring and query apps to the MessageManager, in `main.h`.

files with the necessary code to fulfill the functionalities of the apps.

Each app in LoRaMesher is defined as a class. The monitoring app, which has to send data periodically, is implemented as a FreeRTOS task. This means that at a pre-configured time period, the task aims to get into the running state. The task then reads all the monitoring data and sends it to the gateway node, where the data is forwarded to the queue of the node's MQTT publisher.

For the monitoring app, six different types of data have been chosen: The number of services on the node, the size of the routing table, the number of routes, the status of the LED on the node's board (ON/OFF), the number of outbound messages from the node, and the number of inbound messages to the node.

The query app on the node is only executed on request. This happens when the node receives a message that is addressed to the query application. The query app leverages the *User Receive Task* of the loramesh service of LoRaChat to implement the processing of the received query, read the requested data, and send back the answer to the gateway node, where, like in the monitoring app, the data gets forwarded to the MQTT publisher.

C. Monitoring service

There are many mature tools available to monitor and store IoT data. We have chosen InfluxDB and Telegraf, a widely used monitoring service combination, as it combines an open-source database with a powerful data collector plugin to be able to collect and process data in real-time. Using openly available technologies instead of customized solutions has the advantage for others to more easily reuse the proposed monitoring system.

Telegraf acts as a plugin for the information source, in this case, MQTT, and processes the information with the many processing plugins it has. Once the information is processed as one desires, Telegraf offers output plugins to send the information to InfluxDB.

Grafana is chosen as the visualization technology of the monitoring service. It is an open-source web application that allows for real-time data visualization. It can work with multiple data sources, one of them being InfluxDB. As a result, Grafana can graph and display all the information that it receives from the InfluxDB database, which receives the information from Telegraf.

D. Query application

The query application consists of the implementation of a web application using Reactjs. The main objective of this application is for the user to be able to send requests and receive answers from the nodes in the LoRaMesher network. For the user to send queries to the LoRa mesh network node, a web application with a graphical user interface was developed. With this web application, the user can connect to the desired MQTT broker and subscribe to the desired topic to receive messages from the nodes of the LoRa mesh network. In order

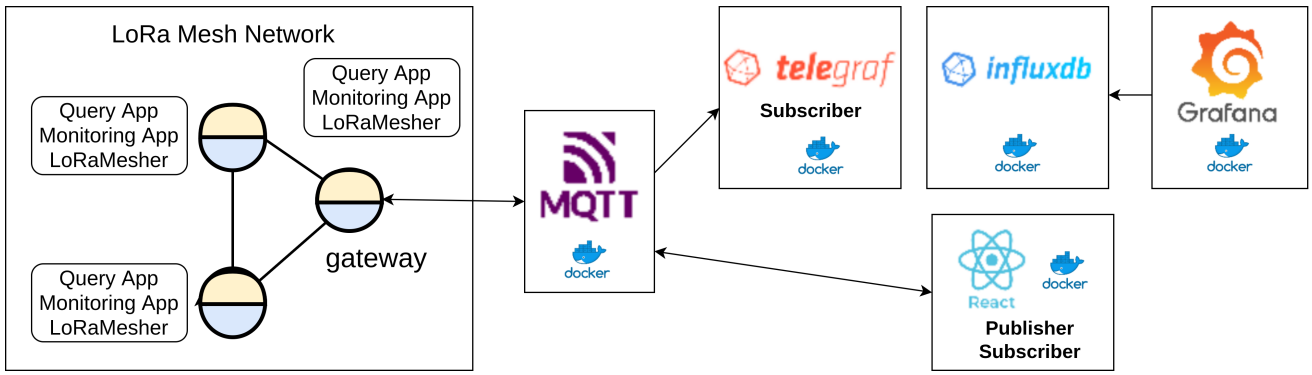

Figure 7. Technology stack.

to send a query to the node, the user publishes to the desired topic.

Figure 7 gives an overview of the technologies used to implement both the monitoring and query solution.

IV. Evaluation

For the proof of feasibility, we use three TTGO T-Beam ESP32 boards to form the LoRa mesh network. The boards are flashed with the two applications that integrate the LoRaMesher and LoRaChat code.

To conduct tests with the applications we create a controlled environment as shown in Figure 8. The three boards interconnect over LoRa with each other and form the LoRa mesh network. The nodes 7AFCh and 869Ch are connected only to the LoRa mesh network while Node DD36h is also connected to a Wifi access point and therefore acts as a gateway. In addition, a laptop is connected to the same Wifi network. It runs a MQTT broker and MQTT clients, and also the monitoring and query applications.

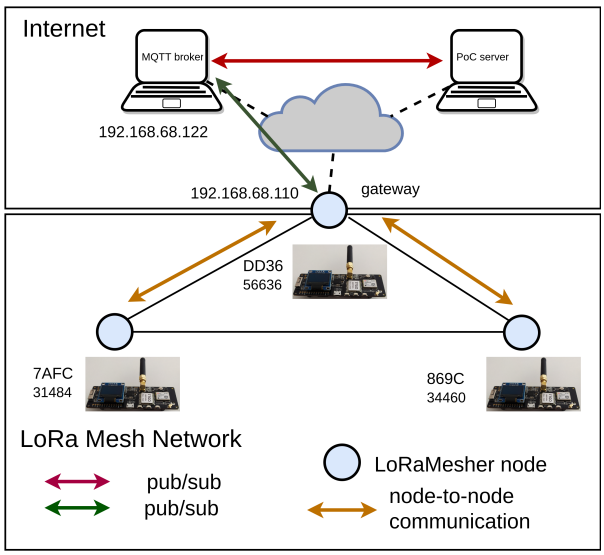

Figure 8. Setup of LoRa mesh network nodes and bidirectional communication from applications on LoRa nodes to Internet services over MQTT.

We deploy the monitoring and query application. Figure 9 shows the user interface of the Query application. It enables the user to choose the MQTT broker to connect to. Once connected, the application subscribes to the generic topic to which the LoRa mesh network publishes through the gateway node. The nodes from which data has been received are indicated. A proactive query can be done by indicating the gateway node, destination node, and the specific metric to be queried. On the right side, a set of answers from different deployed nodes are given.

Figure 10 shows the dashboard created by Grafana for the visualization of the data received from the monitoring app. It shows the data for the three nodes in the LoRa mesh network that sent the monitoring information.

To demonstrate the value of the monitoring system for network understanding, Figure 11 looks into details of the diagram created by Grafana for the *outbound messages* value received from the monitoring app. It can be observed that a third node with id 31384 joined at 12:25h. Furthermore, a loss of messages is shown since the message counters contained in the LoRa packet are received out of order. From this visualization, it can be concluded that the reliable messaging mode of LoRaMesher was activated to resent if the reception at the destination was not acknowledged.

V. Related Work

In this section, we review a few proposals and commercial systems that have demonstrated LoRa mesh networks in real environments with deployments of IoT boards.

Arratia et al. [7] presented a LoRa network-based solution to solve the data transmission of buoy sensor nodes in a lagoon. A protocol design called AlLoRa forwards messages in a multi-hop network. The system does not have a monitoring system for the application itself but data is collected for the evaluation of the communication that is centered on practical aspects of the IoT application case, such as the throughput with different SF and energy consumption.

In the work of Cotrim et al. [8], a LoRa multi-hop network is presented for monitoring soil moisture. Real nodes are deployed in a linear network topology. The ten nodes forward messages to a base station. Nodes wake up periodically to

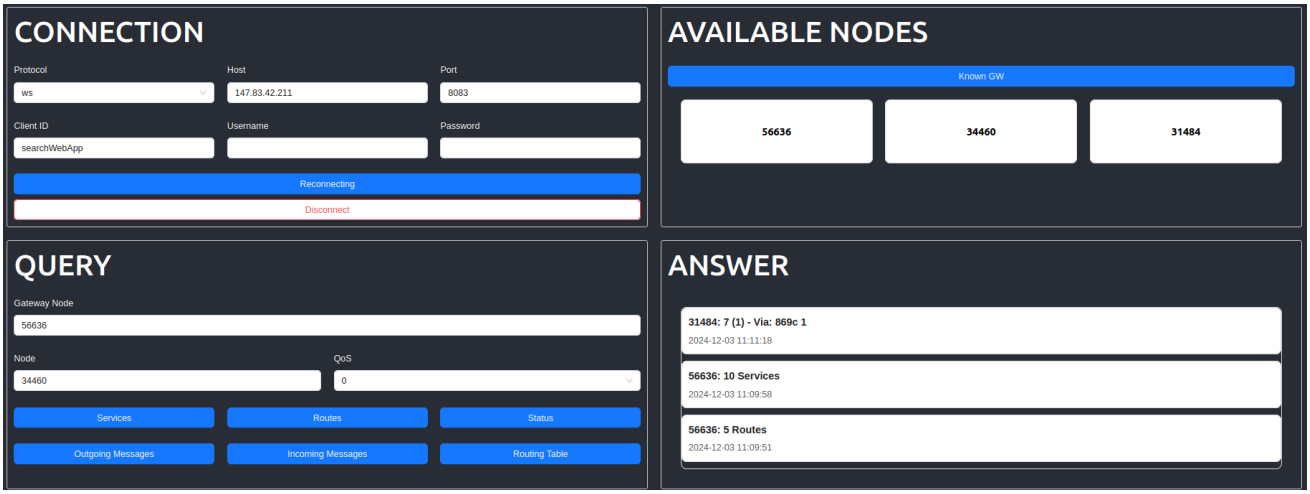

Figure 9. Application to send queries to specific nodes in the LoRa mesh network.

Figure 10. Grafana dashboard.

take sensor values and send the data. The communication of the data is unidirectional from the nodes to the base station as in the monitoring application we present. The presented application does not actively query the nodes.

Berto et al. [9] presented a study with a real LoRa-based mesh network deployment. The prototype of this network is based on the RadioHead library [10]. To validate their research, the authors presented a hardware/software prototype. The experiments used a setup of three nodes and focused on the research evaluation.

In [11], we introduced an initial exploration of monitoring a LoRa mesh network. In that work, an HTTP client was integrated into every LoRa mesh node that also had Wifi connectivity. The communication was unidirectional from the nodes to an ElasticSearch and Kibana-based cloud service. Differently, in our current solution, we use MQTT which allows performing a bidirectional communication that enables the query application.

From the above and the previously referenced works, we can conclude that LoRa mesh network monitoring solutions so far were mostly tailored to research evaluations, where LoRa mesh networks were not yet considered permanently to operated networks. Differently, with our monitoring system, we target a scenario where LoRa mesh networks have already moved from the early research stage into being operational.

VI. Conclusions and Outlook

This paper presented a monitoring system to collect data from a LoRa mesh network. A proof of feasibility was given by an operational implementation. Being able to monitor LoRa mesh networks is important when LoRa mesh networks are considered as a network substrate in professional IoT applications. For permanent operation, potential bottlenecks can be detected, helping to improve the configuration of the network.

In future work, we aim to study the case of hosting applications as services in the LoRa mesh network that can be queried from the Internet. Compared to LoRaWAN, having services on LoRa mesh network nodes opens new ways of how IoT nodes can be integrated into applications.

Acknowledgment

This work was funded by the Spanish State Research Agency grant PDC2023-145809-I00/AEI/10.13039/501100011033 and PID2023-146066OB-I00 (AEI 2023 Knowledge Generation Projects), the Recovery and Resilience Mechanism of the European Union, and by the European Union NextGenerationEU.

References

[1] C. Li and Z. Cao, "Lora networking techniques for large-scale and long-term iot: A down-to-top survey," *ACM Comput. Surv.*, vol. 55, no. 3, feb 2022. [Online]. Available: https://doi.org/10.1145/3494673

[2] A. W.-L. Wong, S. L. Goh, M. K. Hasan, and S. Fattah, "Multi-hop and mesh for lora networks: Recent advancements, issues, and recommended applications," *ACM Comput. Surv.*, vol. 56, no. 6, jan 2024. [Online]. Available: https://doi.org/10.1145/3638241

[3] R. P. Centelles, R. Meseguer, F. Freitag, R. B. Viñas, and L. Navarro, "A minimalistic distance-vector routing protocol for lora mesh networks," *IEEE Access*, pp. 1–1, 2024.

[4] A. Ghosh, S. Misra, V. Udutalapally, and D. Das, "Loraute: Routing messages in backhaul lora networks for underserved regions," *IEEE Internet of Things Journal*, vol. 10, no. 22, pp. 19 964–19 971, 2023.

[5] J. M. Solé, R. P. Centelles, F. Freitag, and R. Meseguer, "Implementation of a lora mesh library," *IEEE Access*, vol. 10, pp. 113 158–113 171, 2022.

[6] R. Pueyo Centelles, "Towards LoRa mesh networks for the IoT," Ph.D. dissertation, Universitat Politècnica de Catalunya, Nov 2021.

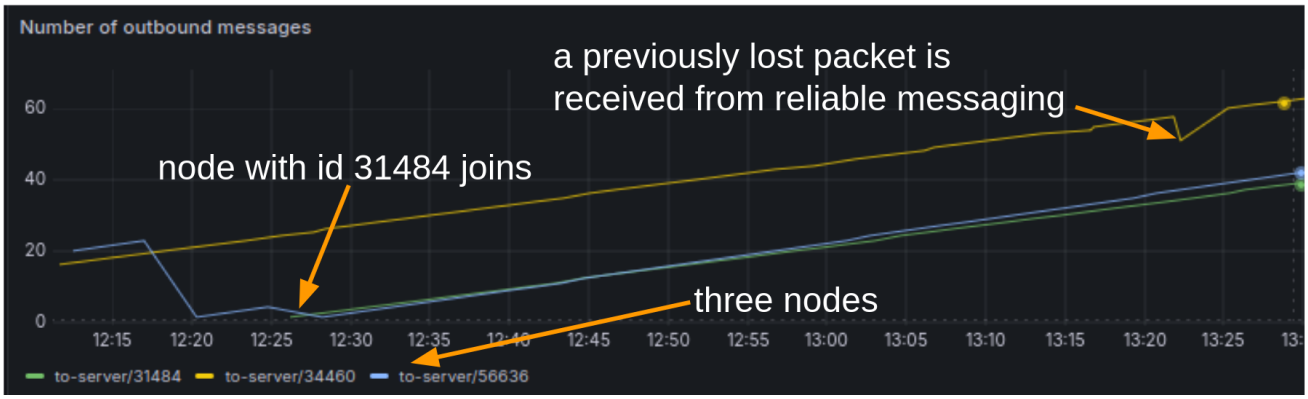

Figure 11. Example of monitoring data analysis.

[7] B. Arratia, E. Rosas, C. T. Calafate, J.-C. Cano, J. M. Cecilia, and P. Manzoni, "Allora: Empowering environmental intelligence through an advanced lora-based iot solution," *Computer Communications*, vol. 218, pp. 44–58, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366424000641

[8] R. Cotrim, F. Assis, A. dos Santos Brito, Y. S. Peixouto, and L. S. Peixouto, "Multi-hop lora-based underground network for monitoring soil moisture in agriculture," *Computers and Electronics in Agriculture*, vol. 227, p. 109592, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169924009839

[9] R. Berto, P. Napoletano, and M. Savi, "A lora-based mesh network for peer-to-peer long-range communication," *Sensors*, vol. 21, no. 13, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/13/4314

[10] M. McCauley, *RadioHead Packet Radio Library for embedded microprocessors*, 2014. [Online]. Available: https://www.airspayce.com/mikem/arduino/RadioHead/

[11] A. J. Capella Del Solar, J. Miquel Solé, and F. Freitag, "Towards a monitoring system for a lora mesh network," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022, pp. 1294–1295.