International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Stratos: A Resilient Lora Mesh Communication Network for Disaster Zones

Kadiyala Tejendra Sai [1], Ms. G. Naga Sujini [2]

[1] Student, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Gandipet, India.
[2]Assistant Professor, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Gandipet, India.

Abstract
In disaster scenarios, communication infrastructure such as cellular networks and internet connectivity often collapses, isolating affected populations and delaying rescue operations. Existing emergency communication systems rely heavily on centralized infrastructure, making them unreliable in such conditions. This paper presents *Stratos*, a resilient and decentralized communication system designed to operate independently of traditional networks. The proposed system utilizes LoRa-based mesh networking to enable long-range, low-power communication across multiple nodes using a multi-hop architecture. A Raspberry Pi-based gateway hosts a Wi-Fi captive portal, allowing civilians to send SOS messages directly from their smartphones without requiring internet access. The system integrates Bluetooth Low Energy (BLE) for device discovery and crowd estimation, along with TinyML-based prioritization to classify and transmit critical messages efficiently. Additionally, geotagged data is visualized on offline maps to assist rescue teams in real-time decision-making. The system demonstrates high reliability, fault tolerance, and scalability in constrained environments. Stratos provides a cost-effective and infrastructure-independent solution for disaster communication and emergency response.

Keywords: LoRa Mesh, Disaster Communication, Raspberry Pi, TinyML, BLE, Offline Mapping, Emergency Networks

1. Introduction

Natural disasters such as earthquakes, floods, and cyclones often result in the failure of communication infrastructure, including cellular towers and internet services. This disruption severely impacts coordination between rescue teams and affected individuals, leading to delays in emergency response.
Traditional communication systems depend on centralized infrastructure, which is vulnerable to failure during disasters. Even advanced systems such as satellite communication are expensive and limited in scalability. Therefore, there is a need for a decentralized, low-power, and resilient communication system that can operate independently.

The *Stratos* system addresses this challenge by leveraging LoRa mesh networking, edge computing, and local communication interfaces to establish a self-sustaining communication network. The system enables

civilians to send SOS messages without internet access while providing rescue teams with real-time situational awareness.

1.1 Problem Definition

Traditional communication systems become highly unreliable during disaster scenarios due to their dependence on centralized infrastructure such as cellular towers, internet connectivity, and power grids. When these systems fail, affected individuals are unable to communicate their location or request assistance, while rescue teams struggle to obtain real-time information from the ground. Existing solutions also suffer from limitations such as high power consumption, restricted communication range, and lack of adaptability to dynamic environments. Furthermore, most systems do not incorporate intelligent mechanisms to prioritize critical messages, resulting in delays in delivering life-threatening alerts. The absence of real-time local visualization and situational awareness further complicates rescue operations. These challenges emphasize the need for a resilient, low-cost, and infrastructure-independent communication system capable of operating efficiently in disaster conditions.

1.2 Existing Applications

Several communication technologies have been explored for use in disaster scenarios; however, each comes with inherent limitations. Satellite communication systems provide wide-area coverage but are expensive, bandwidth-constrained, and require specialized equipment, making them unsuitable for large-scale deployment. Wi-Fi mesh networks offer higher data rates but are limited by short communication range and high energy consumption, restricting their usability in wide disaster zones. Cellular networks, while widely used, depend on infrastructure that is often damaged during disasters, rendering them ineffective when most needed. LoRa-based point-to-point systems provide long-range communication with low power consumption, but they lack dynamic routing capabilities and fail to ensure reliable message delivery across complex terrains. As a result, existing systems do not offer a comprehensive, scalable, and resilient communication solution for disaster-affected environments.

1.3 Proposed Application

The proposed system, Stratos, is a decentralized and infrastructure-independent communication framework designed specifically for disaster environments. It leverages LoRa-based mesh networking to establish multi-hop communication between nodes, enabling reliable long-range data transmission even when parts of the network fail. A Raspberry Pi-based gateway acts as a central coordination unit, hosting a Wi-Fi captive portal that allows civilians to send SOS messages directly from their smartphones without requiring internet access or additional applications. The system incorporates Bluetooth Low Energy (BLE) for device discovery and crowd density estimation, enhancing situational awareness for rescue teams. Additionally, a TinyML-based prioritization mechanism is integrated to classify messages based on urgency, ensuring that critical alerts are transmitted first in bandwidth-constrained scenarios. All messages are geotagged and visualized on an offline map interface, enabling real-time monitoring and efficient decision-making. By combining low-power communication, edge intelligence, and user-friendly access, Stratos provides a scalable and effective solution for disaster communication.

1.4 Requirements Specification

Requirement Specifications describe the artifact of Software Requirements and Hardware Requirements used in this project.

Software Requirements

- **Operating System:** Raspberry Pi OS
- **Programming Language:** Python 3.x
- **Backend Framework:** Flask / FastAPI
- **Database:** SQLite
- **Libraries:** LoRa.h, TinyGPS++, NimBLE
- **Frontend:** HTML, CSS, JavaScript
- **Visualization:** Leaflet.js

Hardware Requirements

- Raspberry Pi (Gateway)
- ESP32 LoRa Nodes
- LoRa Module (SX1276)
- GPS Module (Neo-6M)

2. Literature Survey

"BPoL: A Disruption-Tolerant LoRa Network for Disaster Communication" by Daniel Schmidt, Franz Kuntke, Maximilian Bauer, and Lars Baumgärtner presents a disruption-tolerant networking approach using LoRa technology. The system employs store-carry-forward mechanisms to maintain communication even under network partitions. It demonstrates improved message delivery reliability in disrupted environments, although latency increases in highly fragmented networks. This work provides a strong foundation for implementing delay-tolerant communication in disaster scenarios.

"Humanity Lifeline: A Resilient Communication and Sensor Network Framework for Disaster Response" by Ozgun Pinarer and Omer Komili proposes a hybrid communication system integrating BLE and LoRa technologies. The framework introduces a multi-tier architecture combining sensing, communication, and data processing layers. Experimental results show high message delivery success rates and strong reliability, but the system relies partially on cloud connectivity. This study is highly relevant to the integration of BLE and LoRa in the Stratos system.

"LoRa-based Messaging Device for Remote Off-Grid Communication" by Naveenkumar R, M. Alex Benny, Ajay S, Jesswin Anto J, and Abhishekh R. V presents a low-cost communication device designed for off-grid environments. The system enables peer-to-peer messaging using LoRa technology with minimal power consumption. While effective for small-scale deployments, it lacks large-scale mesh networking capabilities. This paper supports the feasibility of LoRa for infrastructure-independent communication.

"The Impact of LoRa Parameters on UAV-to-X Communications in Emergency Response Scenarios" by Maria Karatzia, Nicolas Souli, Panayiotis Kolios, and Georgios Ellinas evaluates the

performance of LoRa communication in dynamic environments involving UAVs. The study highlights how parameter tuning affects range, latency, and reliability. Although focused on UAV applications, it provides valuable insights into optimizing LoRa communication for disaster scenarios.

"Cost-Effective Emergency Communication System for Remote and Disaster Areas" by P. Lingeswari, B. Mohan, V. Vijaya Bala, M. Vignesh, V. Vigneshkumar, and G. Vimal presents a LoRa-based system integrated with environmental sensors for early disaster detection. The system includes SOS alert mechanisms and RSSI-based location tracking. While it demonstrates practical implementation, it is limited by small-scale testing and lacks intelligent prioritization. This paper highlights the importance of combining sensing with communication in disaster systems.

"A LoRa Based Emergency Communication Device for Disaster Response" by Pradeep Kumar R, Diliban Selvarathinam, and C.N. Gnanaprakasam introduces a self-powered communication device using LoRa and Arduino. The system supports continuous operation through alternative power sources and provides pre-defined emergency messaging. Although reliable, it lacks advanced features such as dynamic routing and real-time prioritization. This study reinforces the need for energy-efficient communication systems.

"LoRa Communication Model Design for Environmental Data of Natural Calamities Area" by Anil Kumar Dubey, Sonam Sirohi, Qarib Anwer, Shazeb Alam, and Mohd Sahil proposes a LoRa-based data transmission model for disaster-prone areas. The system enables long-range communication without requiring network infrastructure and demonstrates high efficiency in environmental monitoring. However, it focuses primarily on data collection rather than communication between victims and rescuers.

"Optimizing Real-Time Communication using LoRa Protocols" by M. Jaiganesh, T.A. Hariprasad, M. Kabilesh, and R. Harish Raj explores techniques to improve LoRa communication performance through parameter optimization. The study demonstrates that proper tuning can significantly enhance latency and throughput. However, the inherent low bandwidth of LoRa remains a limitation. This paper is useful for understanding performance trade-offs in LoRa networks.

"Mobile Gateways for LoRa End-Devices: Support Connectivity on the Move" by Vasileios Moysiadis et al. proposes the use of mobile gateways to maintain connectivity with moving LoRa devices. The system uses RSSI-based estimation to dynamically adjust gateway positioning. While effective in maintaining communication, it introduces complexity in mobility management. This study is relevant for extending coverage in dynamic disaster environments.

3. Methodology

This section describes the approach used to design, develop, and implement the Stratos system, ensuring a reliable, scalable, and infrastructure-independent communication network for disaster environments. The methodology focuses on integrating low-power communication, edge computing, and decentralized networking to enable real-time emergency communication without relying on existing infrastructure.

3.1 Technologies Used

This section outlines the key technologies used in developing the Stratos system.

a) Programming Language: Python 3.x and Arduino C++, Python is used for backend development on the Raspberry Pi gateway due to its simplicity and strong library support, while Arduino C++ is used for programming ESP32-based LoRa nodes for efficient low-level hardware control and communication.

b) Hardware Platform: The system utilizes ESP32-based LoRa nodes for field communication and a Raspberry Pi as the central gateway. The ESP32 nodes handle message transmission and reception, while the Raspberry Pi processes incoming data, hosts the dashboard, and manages the overall network.

c) Backend Framework: Flask or FastAPI is used to build a lightweight backend server on the Raspberry Pi. It handles message processing, storage, API endpoints, and communication between the LoRa network and the user interface.

d) Database: SQLite is used as a lightweight local database to store SOS messages, node information, timestamps, geolocation data, and priority levels. It ensures fast and reliable data access in offline conditions.

e) Communication: The system uses LoRa for long-range, low-power communication between nodes and the gateway. Additionally, Wi-Fi is used to provide a captive portal for user interaction, and BLE is used for device discovery and proximity estimation.

f) Data Processing and Intelligence: TinyML models are integrated at the edge to classify and prioritize incoming messages based on urgency. This ensures that critical messages are transmitted first in bandwidth-constrained environments.

g) Visualization: Leaflet.js along with offline OpenStreetMap tiles is used to generate real-time map visualizations of SOS messages and node locations. This enables rescue teams to monitor affected areas without internet connectivity.

3.2 Development Process:

The development of the Stratos system follows an iterative and modular approach to ensure flexibility, reliability, and adaptability in disaster environments where conditions are unpredictable and dynamic.

a) Requirement Analysis: The system requirements were identified by analysing communication challenges in disaster scenarios, particularly the failure of cellular networks, internet infrastructure, and power systems. Key issues such as lack of connectivity, absence of real-time communication, and inability to prioritize emergency messages were identified, leading to the need for a decentralized, low-power, and infrastructure-independent communication solution.

b) System Design: The architecture was designed as a distributed mesh-based system consisting of LoRa-enabled field nodes and a Raspberry Pi gateway. The design ensures modularity, where each node operates

independently while contributing to the overall network. The gateway acts as a central processing unit, handling message aggregation, prioritization, and visualization.

c) Module Implementation: The system is structured into multiple functional modules to ensure modularity and efficient operation. The LoRa communication module enables long-range, multi-hop transmission of messages across distributed nodes, forming the backbone of the network. The Wi-Fi captive portal module provides a user-friendly interface that allows civilians to connect using their smartphones and send SOS messages without requiring internet access. Incoming data is handled by the message processing module, which validates, formats, and prepares messages for storage and further analysis. A TinyML-based prioritization module is integrated to classify messages based on urgency, ensuring that critical alerts are transmitted with higher priority. The BLE module facilitates device discovery and assists in estimating crowd density by detecting nearby Bluetooth signals. Finally, the visualization module presents all collected data on an offline map interface, enabling rescue teams to monitor locations, message status, and network activity in real time.

d) Backend Integration: The backend, implemented using Flask or FastAPI on the Raspberry Pi, processes incoming LoRa messages, validates data, assigns priority levels using TinyML, and stores the information in a local database. It also updates the rescue dashboard in real time, ensuring continuous monitoring without external connectivity.

e) Testing: Each module was tested individually to ensure correct functionality, followed by integration testing to validate communication between nodes, gateway, and user interfaces. The system was tested under simulated disaster conditions, including node failures and intermittent connectivity. Minor transmission delays and packet losses were observed and handled using retry mechanisms and store-and-forward techniques.

f) Deployment and Evaluation: The system was deployed in a controlled real-world environment to evaluate performance metrics such as message delivery success rate, network reliability, latency, and prioritization accuracy. The results demonstrated that the system operates effectively in low-connectivity conditions and maintains communication even in the presence of partial network failures.

4. Design of Stratos System

The Stratos system is designed as a modular, scalable, and infrastructure-independent communication platform for disaster environments. It follows a decentralized architecture where multiple LoRa-enabled field nodes interact with a central gateway to enable reliable message transmission and real-time situational awareness. The design focuses on resilience, fault tolerance, and accessibility, ensuring continuous operation even in the absence of conventional communication infrastructure.

4.1. System Architecture

The system consists of multiple components interacting through LoRa-based communication and local processing, as shown in Figure 4.1.

Key Components:

- **ESP32 LoRa Nodes (Field Nodes):** Capture user messages, perform local processing, and transmit data across the mesh network.
- **LoRa Communication Module:** Enables long-range, low-power, multi-hop communication between nodes and the gateway.
- **Raspberry Pi Gateway:** Acts as the central processing unit, aggregating messages, performing prioritization, and hosting local services.
- **Backend Server:** Processes incoming data, manages APIs, and updates the system dashboard.
- **Database (SQLite):** Stores SOS messages, node status, timestamps, and geolocation data for efficient retrieval and analysis.
- **Wi-Fi Captive Portal:** Provides a user interface for civilians to send SOS messages without internet access.
- **Offline Map Visualization:** Displays geotagged messages and node locations, enabling real-time monitoring for rescue teams.

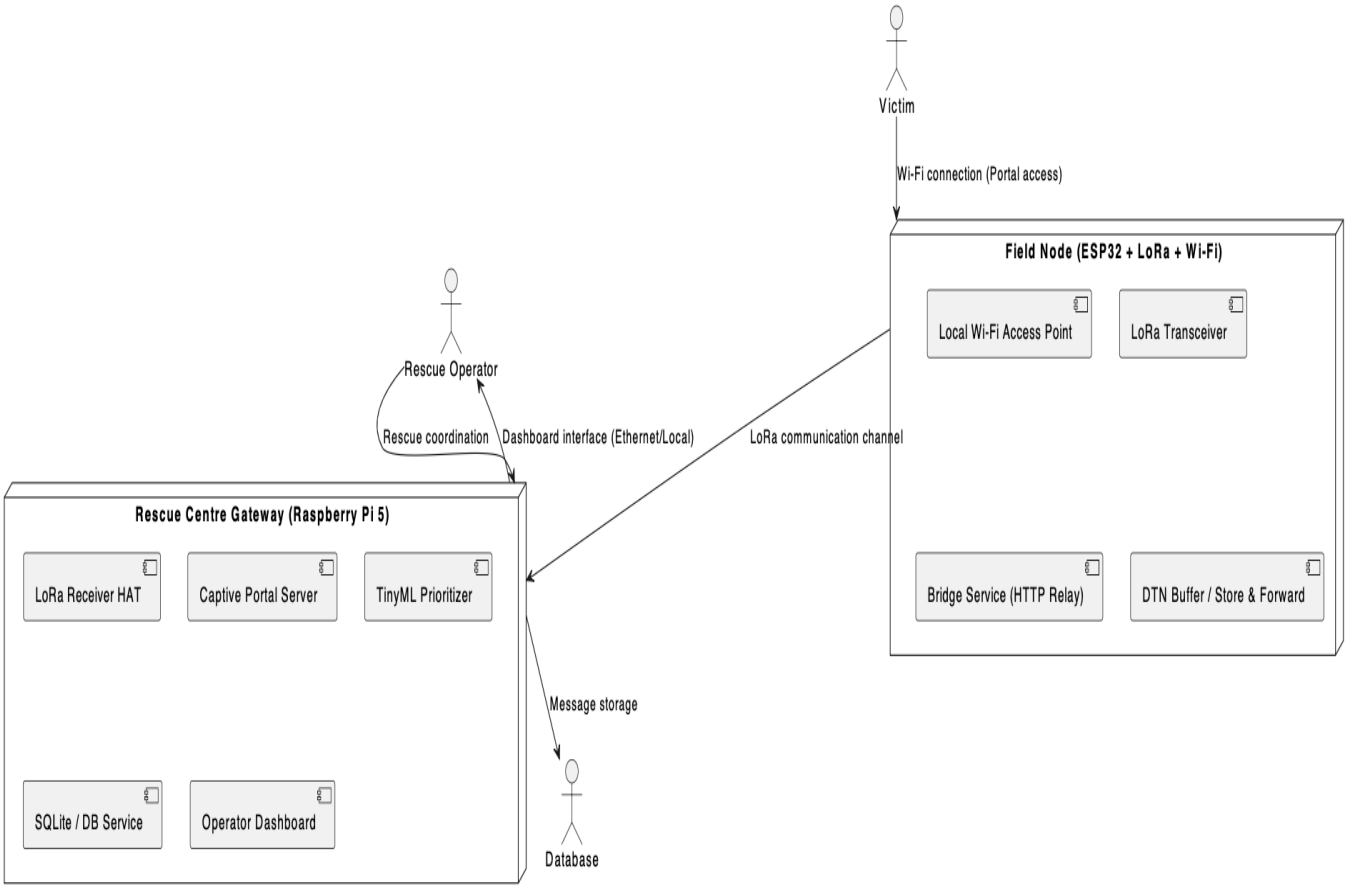

Figure 4.1: System architecture diagram

4.2. Database Design

The system uses a structured local database to store emergency messages, node information, and network-related data. The database is designed to operate efficiently in offline conditions, ensuring reliable storage and retrieval without dependence on external servers.

Main Data Elements:

SOS Message Data: Contains message content, user details, priority level, and status (pending, acknowledged, resolved)

Node Data: Includes node ID, connectivity status, signal strength (RSSI), and battery information

Geolocation Data: Stores GPS coordinates associated with each message for mapping and tracking

Timestamp Data: Records the time of message submission and updates for proper sequencing and analysis

Core Functions:

- **storeMessage()** → Stores incoming SOS messages along with associated metadata
- **classifyPriority()** → Assigns urgency levels to messages using TinyML-based classification
- **updateNodeStatus()** → Tracks and updates the status of each node in the network
- **fetchDashboardData()** → Retrieves processed data for real-time visualization on the rescue dashboard

This design ensures efficient storage, quick retrieval, and reliable processing of critical data, enabling real-time monitoring and decision-making in disaster scenarios.

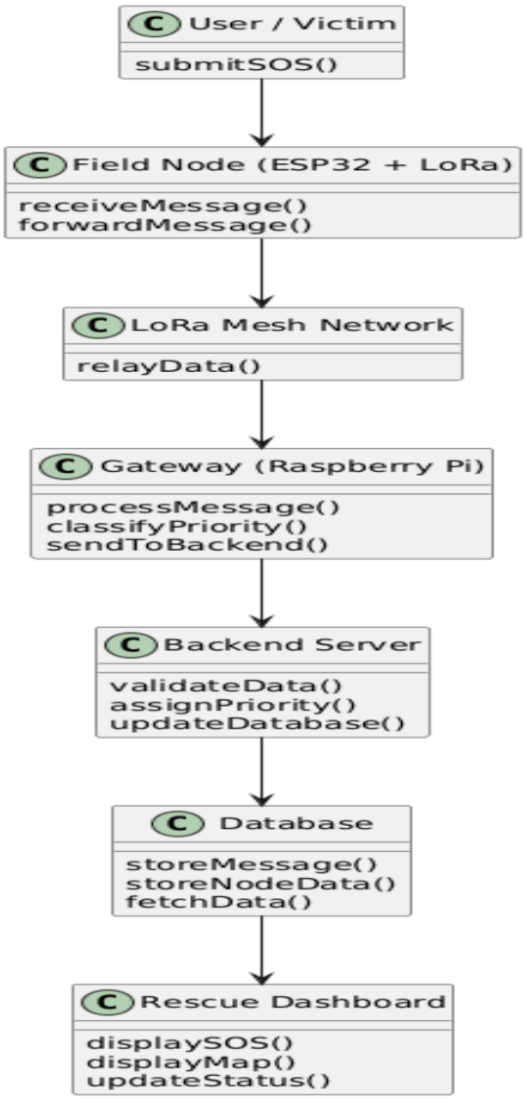

Figure 4.2: ER diagram

4.3. Control Flow showing System Activity

The control flow diagram illustrates how the Stratos system processes SOS messages and manages communication across the network in a step-by-step manner.

Workflow:

1. A user connects to a nearby node through the Wi-Fi captive portal
2. The user submits an SOS message along with required details
3. The field node receives and forwards the message via LoRa communication
4. The message is relayed across multiple nodes using mesh routing
5. The gateway receives the message and validates the incoming data
6. The TinyML module classifies the message based on urgency
7. The processed data is stored in the local database
8. The dashboard is updated with message details and geolocation

This flow ensures reliable message transmission, intelligent prioritization, and real-time visualization, enabling efficient communication and response during disaster situations.

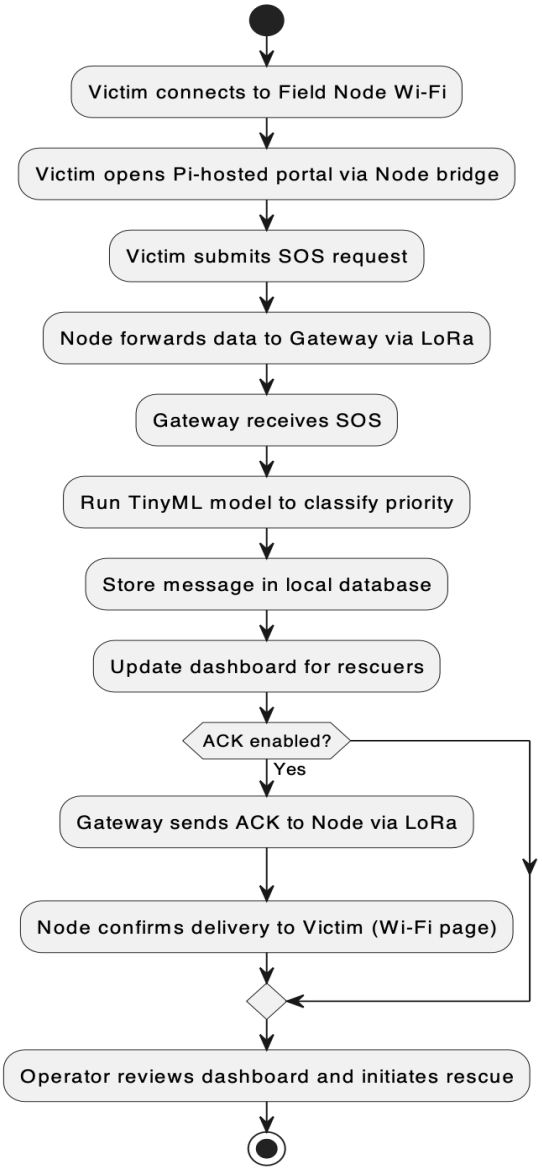

Figure 4.3: Control Flow Diagram

4.4. Modular Architecture & Object-Oriented Structure

The system is designed using modular components, where each module is responsible for a specific functionality. This modular architecture ensures flexibility, scalability, and ease of maintenance, allowing individual components to operate independently while contributing to the overall system.

Key Modules & Responsibilities:

- **LoRa Communication Module:**
  - o Handles long-range, multi-hop message transmission between nodes
  - o Ensures reliable data forwarding across the mesh network
- **ESP32 Node Module:**
  - o Receives user input through the captive portal
  - o Transmits and relays messages within the network
- **Wi-Fi Captive Portal Module:**
  - o Provides a user interface for submitting SOS messages
  - o Enables access without requiring internet connectivity
- **Message Processing Module:**
  - o Validates incoming messages and checks data integrity
  - o Prepares data for storage and further processing
- **TinyML Prioritization Module:**
  - o Classifies messages based on urgency and context
  - o Assigns priority levels to optimize communication
- **BLE Module:**
  - o Detects nearby Bluetooth devices
  - o Assists in estimating crowd density and proximity
- **Database Module:**
  - o Stores SOS messages, node data, and timestamps
  - o Retrieves information for dashboard visualization

• **Visualization Module:**

o Displays real-time SOS messages and node status on offline maps

o Provides situational awareness for rescue teams

This modular approach significantly enhances the overall effectiveness of the system by allowing each component to function independently while still contributing to the larger communication framework. In terms of scalability, new nodes, features, or functionalities can be added to the system without requiring major changes to the existing architecture. For example, additional LoRa nodes or sensing modules can be integrated seamlessly to expand network coverage in larger disaster zones.

From a maintenance perspective, modularity simplifies debugging and system updates. Since each module is isolated, faults can be identified and resolved within a specific component without affecting the entire system. This reduces downtime and makes the system easier to manage, especially in critical emergency scenarios where reliability is essential.

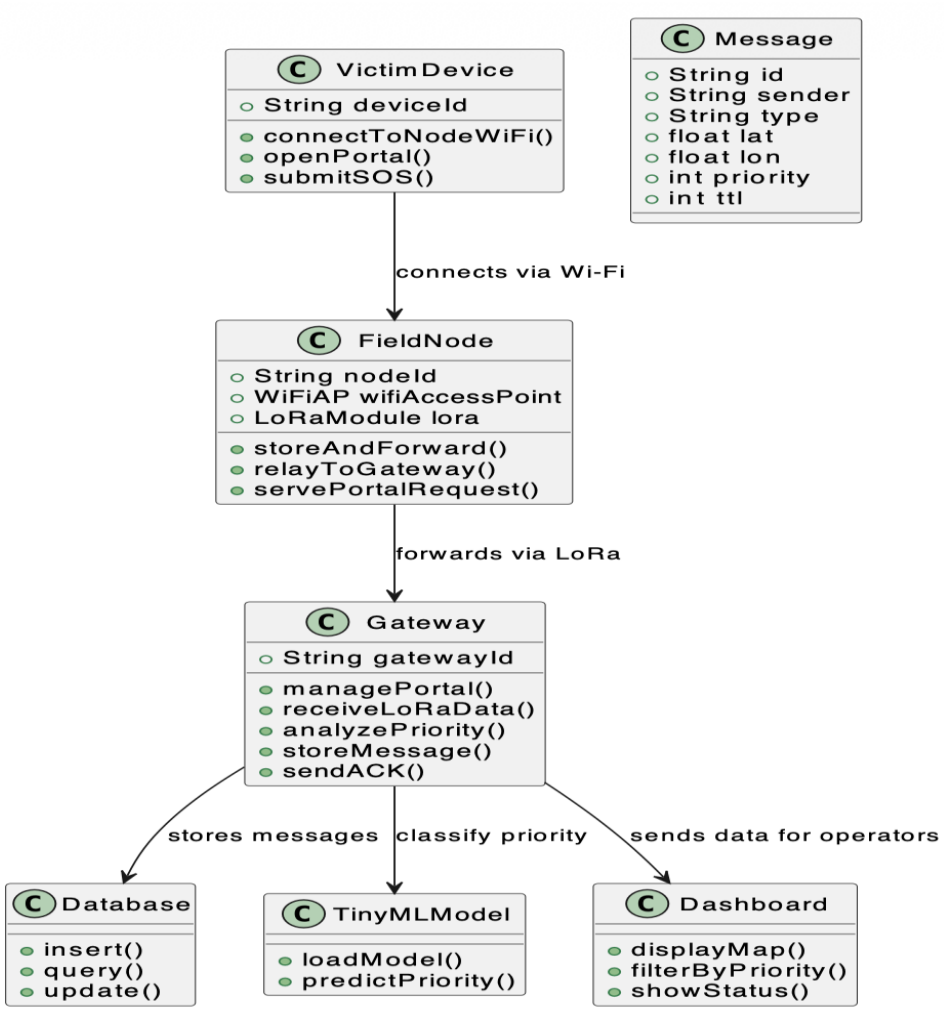

Figure 4.4: Class diagram

4.5 Role-Based System Interaction

The system supports interaction between different users and system components to enable efficient communication and coordination during disaster scenarios.

Actors & Interactions:

- **User (Victim):**
  - o Connects to the Wi-Fi captive portal through a nearby node
  - o Submits SOS messages along with necessary details
- **Rescue Operator:**
  - o Monitors incoming SOS messages through the dashboard
  - o Analyzes priority levels and coordinates rescue actions
- **System (Stratos Network):**
  - o Automatically processes incoming messages from nodes
  - o Classifies and prioritizes messages using TinyML
  - o Routes messages through the LoRa mesh network and updates the database

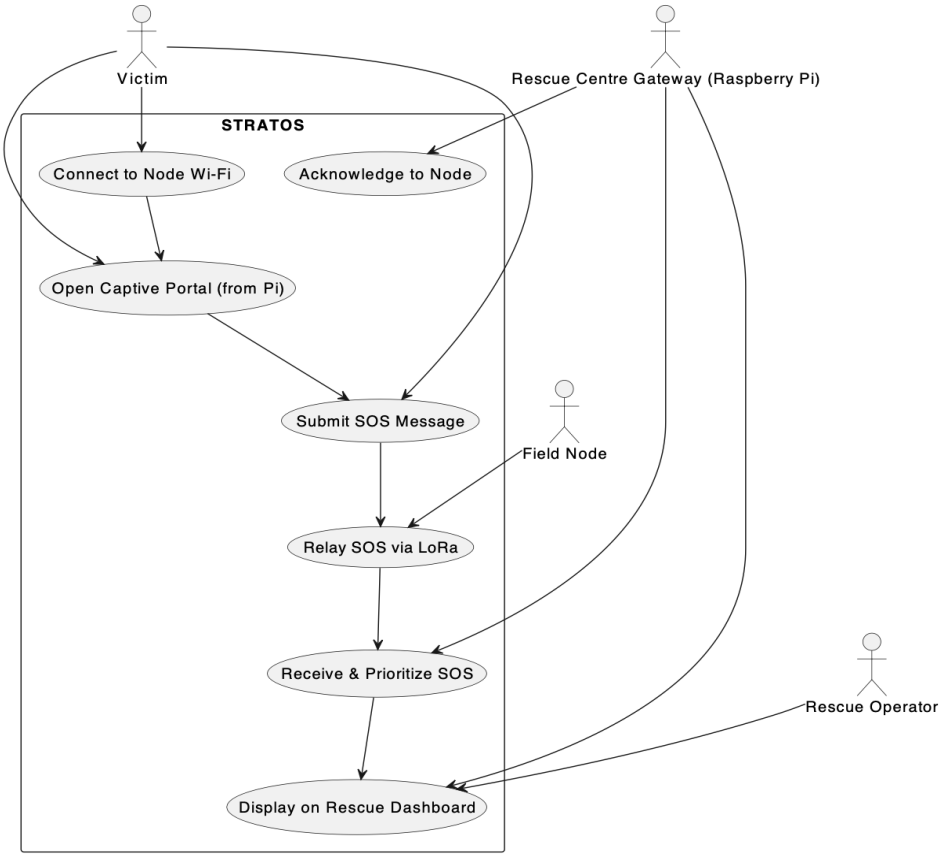

Figure 4.5: Use case diagram

4.6 User-Based Process Workflow

The sequence diagram represents the step-by-step interaction between system components during emergency message transmission and processing in the Stratos network.

Process Flow:

1. User connects to a nearby node via the Wi-Fi captive portal
2. User submits an SOS message with relevant details
3. The field node receives and prepares the message for transmission
4. The message is forwarded through the LoRa mesh network
5. The gateway receives the message and validates the data
6. The TinyML module classifies the message based on urgency
7. The message is stored in the database and processed further
8. The rescue dashboard displays the message along with location and priority

This workflow ensures reliable communication, intelligent prioritization, and real-time visibility of emergency situations without relying on external network infrastructure.

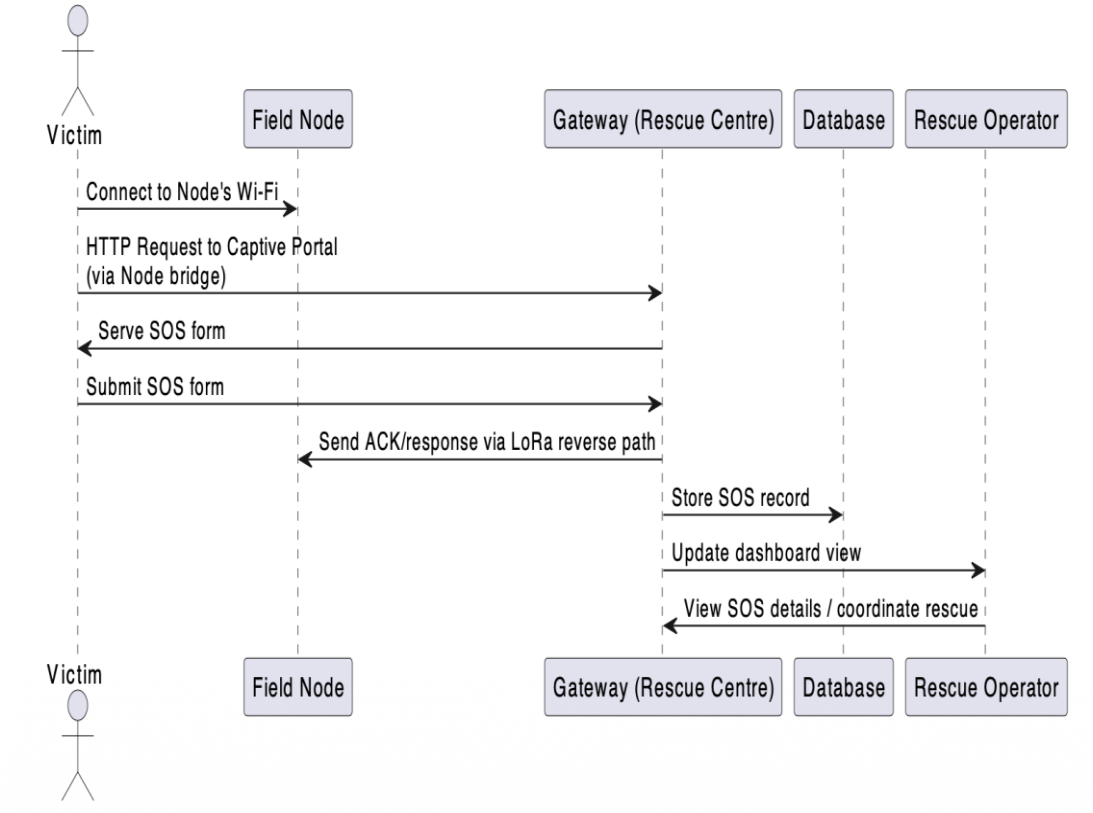

Figure 4.6: Sequence Diagram

5. Implementation

5.1 System Architecture

The Stratos system is implemented as a decentralized, real-time communication network consisting of LoRa-enabled field nodes and a Raspberry Pi-based gateway. The field nodes capture user-generated SOS messages through a Wi-Fi captive portal and transmit them across the network using LoRa communication. The Raspberry Pi gateway receives these messages and processes them locally using a lightweight backend server.

The system follows a modular architecture where each component operates independently while maintaining seamless communication. The primary modules include the LoRa communication module, message processing module, TinyML prioritization module, BLE module, database module, and visualization module. This design ensures scalability, flexibility, and efficient real-time handling of emergency data in infrastructure-less environments.

5.2 Database Integration

The system uses a structured local database to store SOS messages, node data, and network-related information. The database maintains records such as message ID, user details, timestamps, geolocation coordinates, and assigned priority levels.

Data is received from the gateway in structured format and stored sequentially in the database. Each incoming message is logged with its associated metadata, enabling continuous tracking and monitoring. The system updates message status and priority dynamically as new data is processed.

Database operations are managed through backend functions that ensure efficient storage, retrieval, and updating of information. This supports real-time monitoring, historical analysis, and improved situational awareness for rescue teams.

5.3 LoRa and Communication Integration

The LoRa communication module forms the backbone of the Stratos system, enabling long-range, low-power data transmission between nodes. Each ESP32 node communicates using LoRa transceivers and relays messages across the mesh network using multi-hop routing.

In addition to LoRa, Wi-Fi is used to host a captive portal for user interaction, allowing civilians to connect and send SOS messages without requiring internet access. BLE is also integrated to enable device discovery and assist in estimating nearby device density.

To ensure reliable communication, the system incorporates message validation, retransmission mechanisms, and store-and-forward techniques. These features help maintain consistent data flow even under unstable network conditions.

5.4 Message Processing and Prioritization

Incoming messages are processed at the gateway using a structured pipeline. The system validates message content, checks for completeness, and prepares the data for further analysis. A TinyML-based model is used to classify messages based on urgency by analyzing keywords and contextual information.

Messages are categorized into different priority levels, ensuring that critical alerts such as medical emergencies are handled first. This prioritization mechanism optimizes the use of limited bandwidth and improves response efficiency in high-load scenarios.

5.5 Data Visualization and Mapping

The system provides a real-time visualization interface through a locally hosted dashboard. Offline mapping is implemented using Leaflet.js along with preloaded OpenStreetMap tiles, allowing operation without internet connectivity.

Each SOS message is plotted on the map using its geolocation coordinates, with color-coded markers representing priority levels. The dashboard also displays node status, message details, and timestamps, enabling rescue teams to monitor the situation effectively.

5.6 User Interface

The user interface is implemented as a lightweight web application accessible through a Wi-Fi captive portal. It allows users to submit SOS messages with minimal input, ensuring ease of use even under stressful conditions.

The interface is designed using HTML, CSS, and JavaScript, focusing on simplicity, responsiveness, and quick interaction. Feedback mechanisms are included to confirm successful message transmission, improving user confidence and usability.

5.7 Error Handling and Reliability

The system incorporates several mechanisms to ensure reliability and robustness during operation.

- **Message Validation:** Incoming messages are verified to ensure data integrity before processing
- **Retry Mechanism:** Messages are retransmitted in case of temporary communication failures
- **Store-and-Forward:** Nodes buffer messages when connectivity is unavailable and forward them when the network is restored
- **Fault Tolerance:** The mesh network dynamically reroutes messages in case of node failure

The modular design allows each component to function independently, simplifying debugging and maintenance. These mechanisms ensure continuous and reliable operation in unpredictable disaster environments.

6. Testing and Results

The Stratos system was tested to evaluate its performance, reliability, and effectiveness under simulated and real-world disaster conditions. The testing phase focused on validating LoRa-based communication, message transmission reliability, TinyML-based prioritization, and offline data visualization.

Figure 6.1 shows the overall setup of the system, including ESP32 LoRa field nodes, the Raspberry Pi gateway, and the communication interfaces used for real-time message transmission across the mesh network.

Figure 6.1: Project Setup Overview

Figure 6.2 demonstrates the the Wi-Fi selection screen showing the availability of the “EMERGENCY_PORTAL” network for users in disaster zones. It enables quick access to the Stratos captive portal without requiring internet connectivity.

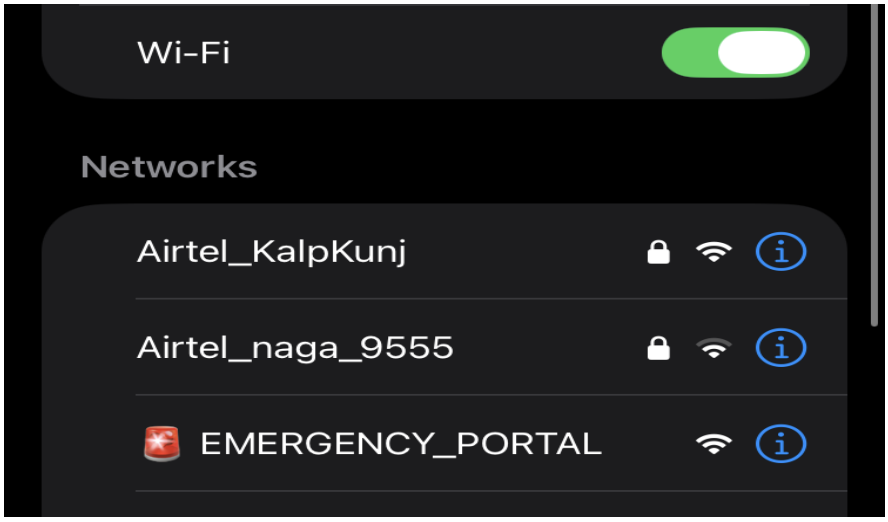

Figure 6.2: Wi-Fi selection screen showing EMERGENCY_PORTAL

Figure 6.3, 6.4, and 6.5 demonstrates the home page of the captive portal. The detailed submission interface allows users to provide additional information and request urgent medical assistance. The final submission button initiates the broadcasting of the SOS message across the LoRa mesh network.

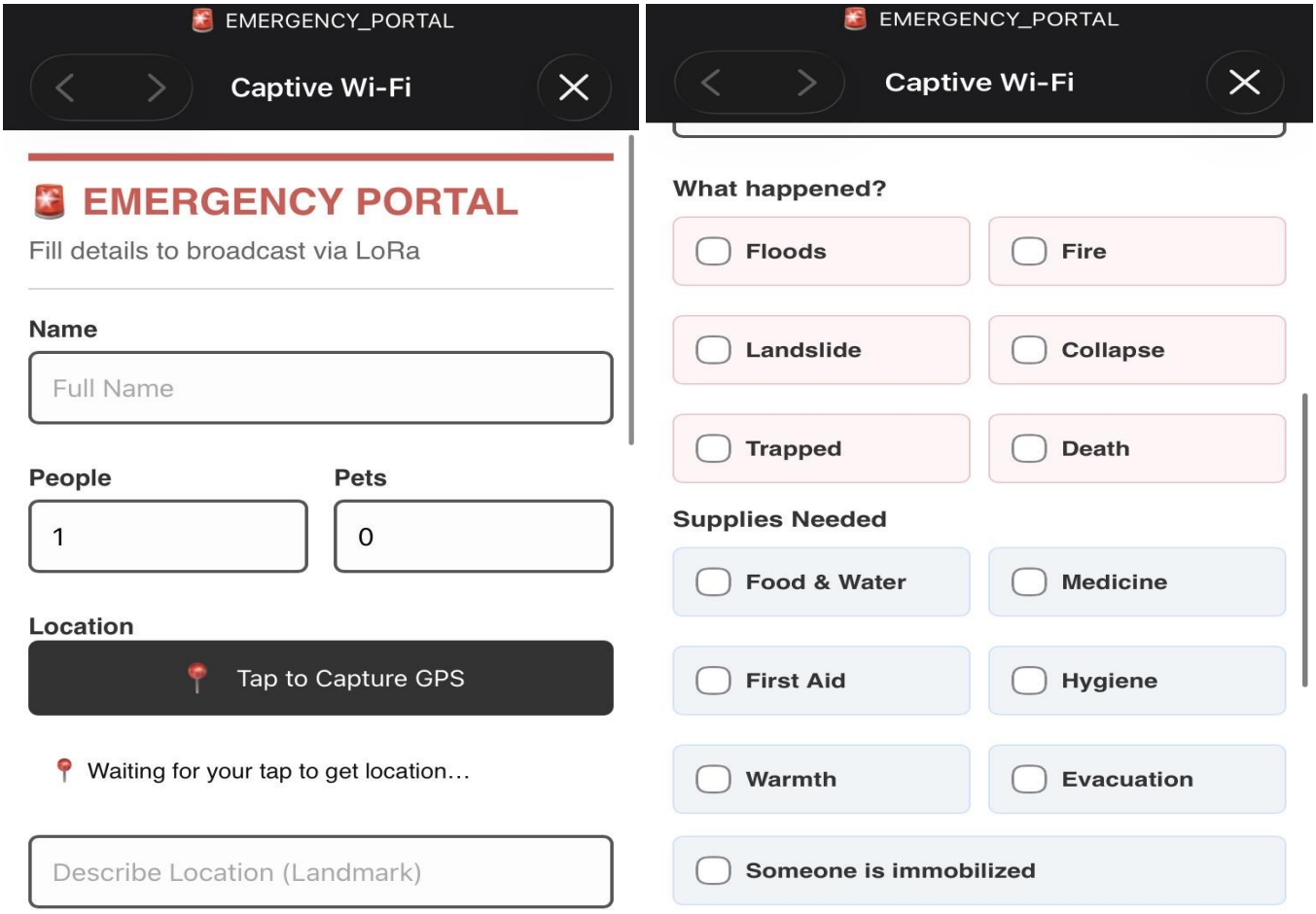

Figure 6.3, 6.4, and 6.5: Emergency portal home page

Figure 6.6 shows the the safety confirmation screen that allows users to mark themselves as safe after receiving assistance

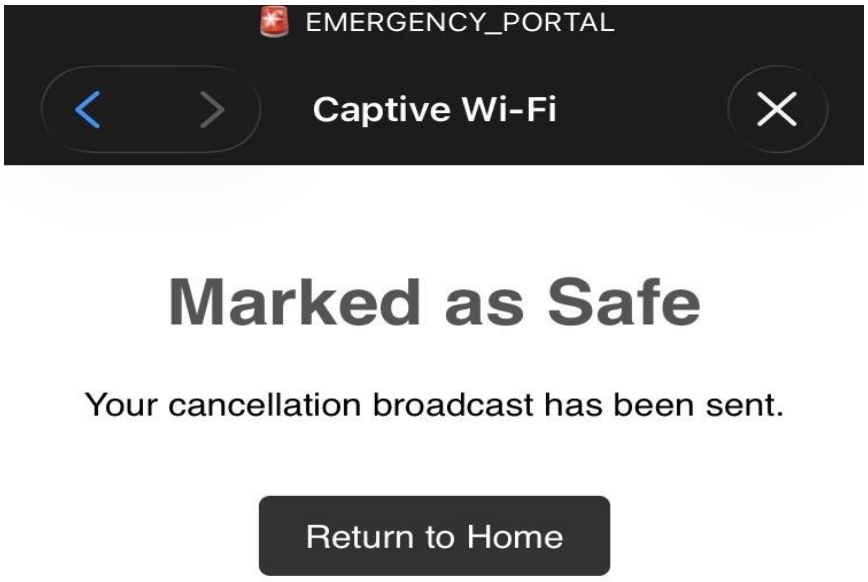

Figure 6.6: Safety Confirmation Screen

Figure 6.7 shows the Stratos Command dashboard displaying real-time incoming alerts along with priority levels and user details. It enables rescue teams to monitor and manage multiple emergency requests efficiently.

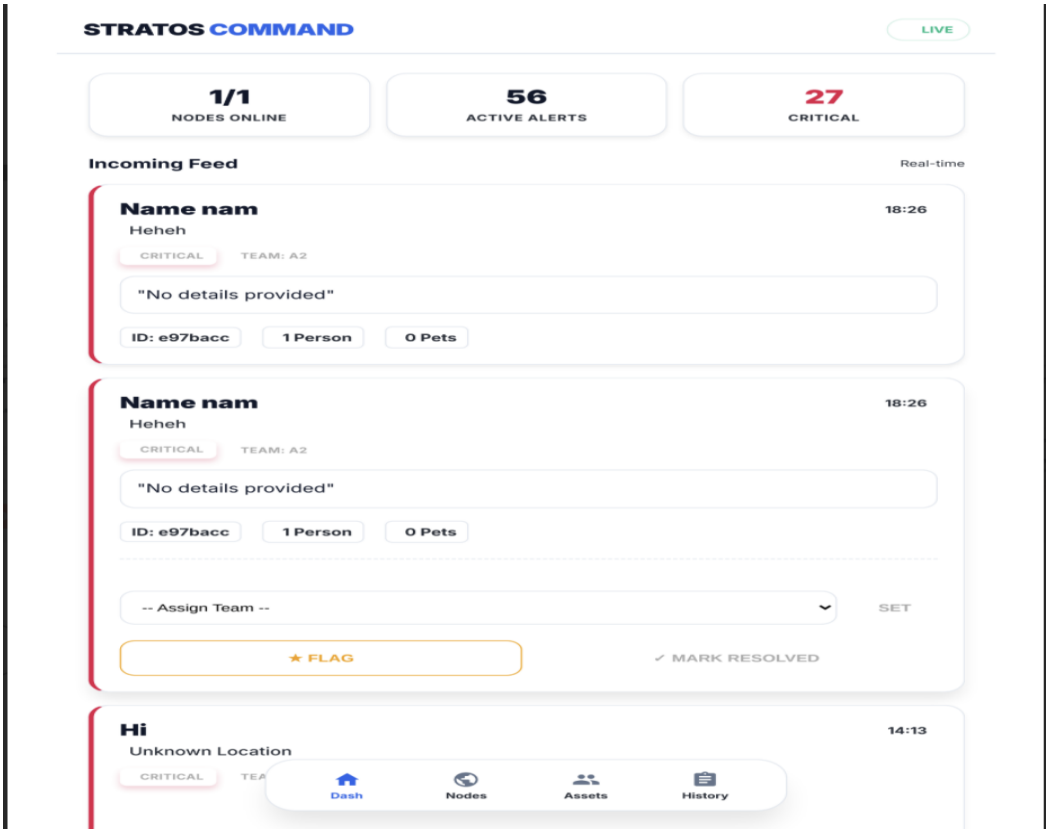

Figure 6.7: Stratos Command dashboard

Figure 6.8 illustrates the network mesh view providing an overview of active nodes and their signal strength. This helps in understanding network coverage and connectivity status in the affected area.

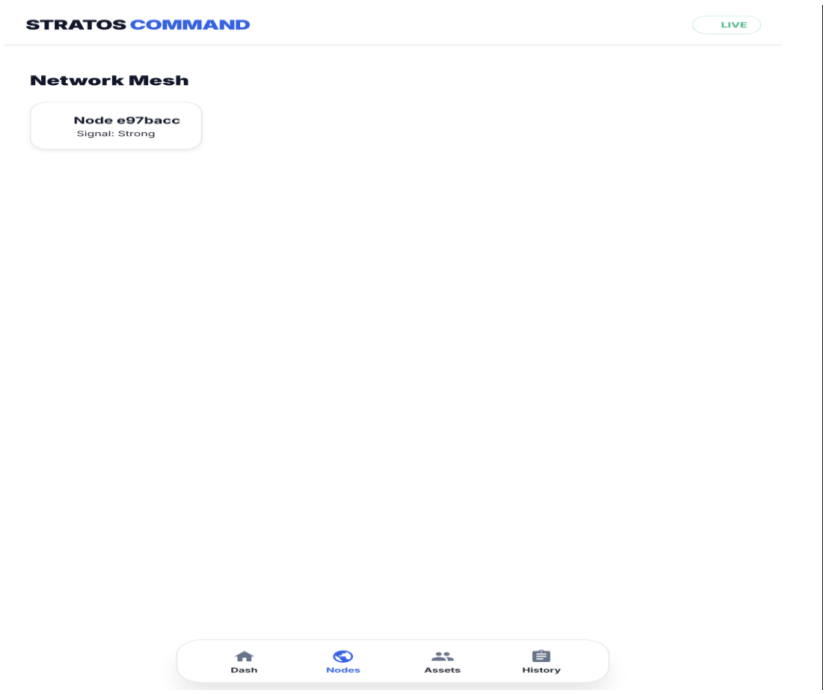

Figure 6.8: Network Mesh view dashboard

The system was tested under real-world and simulated disaster conditions by deploying multiple LoRa nodes and a Raspberry Pi gateway across a controlled environment. The results demonstrate that the system performs efficiently and reliably, ensuring successful message transmission, effective prioritization, and stable operation even under constrained network conditions.

7. Conclusion and Future Scope

7.1 Conclusion

The Stratos system presents an effective and resilient solution to the limitations of traditional communication systems in disaster scenarios. By leveraging LoRa mesh networking, Raspberry Pi-based edge processing, and infrastructure-independent design, the system successfully enables communication in environments where conventional networks are unavailable. The implementation demonstrates reliable multi-hop message transmission, efficient data handling, and real-time visualization of emergency information.

One of the key achievements of this system is the integration of decentralized communication with intelligent message prioritization. The use of TinyML ensures that critical SOS messages are identified and transmitted with higher priority, improving response efficiency during emergencies. The inclusion of a Wi-Fi captive portal allows civilians to communicate using standard smartphones without requiring internet access, significantly enhancing accessibility.

Overall, the Stratos system proves to be a scalable, cost-effective, and practical solution for disaster communication, offering a robust platform that can significantly improve emergency response and coordination.

7.2 Future Scope

Although the current system achieves its primary objectives, several enhancements can further improve its functionality and scalability.

- **Advanced Routing Algorithms:** Incorporating intelligent routing techniques can optimize message delivery paths and reduce latency in large-scale deployments.
- **Enhanced TinyML Models:** More advanced machine learning models can be integrated to improve message classification accuracy and support multilingual inputs.
- **Improved Localization Techniques:** Implementing GPS-independent methods such as RSSI-based or hybrid localization can enhance accuracy in challenging environments.
- **Integration with IoT Sensors:** Adding environmental sensors (temperature, gas, motion) can provide additional situational data for rescue operations.
- **Mobile and Web Applications:** Developing dedicated applications can improve user interaction and provide additional features such as notifications and tracking.
- **Cloud Integration:** Hybrid cloud-edge architecture can be introduced for large-scale deployments

requiring centralized monitoring and analytics.

- **Energy Optimization:** Incorporating solar power and advanced energy management techniques can extend node operational life in remote areas.
- **AI-Based Decision Support:** Future systems can integrate AI models to assist rescue teams in prioritizing actions and allocating resources effectively.

With these enhancements, the Stratos system can evolve into a comprehensive disaster response platform capable of providing intelligent communication, real-time insights, and scalable support for emergency management operations.

References

1. Daniel Schmidt, Franz Kuntke, Maximilian Bauer, and Lars Baumgärtner, "BPoL: A Disruption-Tolerant LoRa Network for Disaster Communication," 2023 IEEE Global Humanitarian Technology Conference (GHTC), Radnor, PA, USA, 2023, pp. 440–447. https://ieeexplore.ieee.org/document/10354717
2. Ozgun Pinarer and Omer Komili, "Humanity Lifeline: A Resilient Communication and Sensor Network Framework for Disaster Response," IEEE Access, vol. 13, pp. 95922–95933, 2025. https://ieeexplore.ieee.org/document/11020646
3. P. Lingeswari, B. Mohan, V. V. Bala, M. Vignesh, V. Vigneshkumar, and G. Vimal, "Cost-Effective Emergency Communication System for Remote and Disaster Areas," 2025 8th International Conference on Circuit, Power & Computing Technologies (ICCPCT), Kollam, India, 2025, pp. 1797–1802. https://ieeexplore.ieee.org/document/11176677
4. Maria Karatzia, Nicolas Souli, Panayiotis Kolios, and Georgios Ellinas, "The Impact of LoRa Parameters on UAV-to-X Communications in Emergency Response Scenarios," 2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring), Oslo, Norway, 2025, pp. 1–7. https://ieeexplore.ieee.org/document/11174599
5. Naveenkumar R., M. Alex Benny, Ajay S., Jesswin Anto J., and Abhishekh R. V., "LoRa-based Messaging Device for Remote Off-Grid Communication," 2025 7th International Conference on Intelligent Sustainable Systems (ICISS), India, 2025, pp. 343–349. https://ieeexplore.ieee.org/document/11076414
6. Pradeep Kumar R., Diliban Selvarathinam K. P., and C. N. Gnanaprakasam, "A LoRa Based Emergency Communication Device for Disaster Response," 2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM), Kanyakumari, India, 2025, pp. 1074–1081. https://ieeexplore.ieee.org/document/10988293
7. Anil Kumar Dubey, Sonam Sirohi, Qarib Anwer, Shazeb Alam, and Mohd Sahil, "LoRa Communication Model Design for Environmental Data of Natural Calamities Area," 2024 International Conference on Emerging Technologies and Innovation for Sustainability (EmergIN), Greater Noida, India, 2024, pp. 147–151. https://ieeexplore.ieee.org/document/10960925
8. Zafirah Faiza, Fatimah Zaharah Ali, Mohd Farid Omar, Mardeni Roslee, Muhammad Syazwan Johari, and Mohamad Huzaimy Jusoh, "Structured Packet Scheduling for Ground-Based LoRa IoT

Networks: Trends, Challenges and TDMA Solutions," 2025 IEEE 17th Malaysia International Conference on Communication (MICC), Malacca, Malaysia, 2025, pp. 1–6. https://ieeexplore.ieee.org/document/11210984