ResilientP2PTestbed

An Off-Grid Disaster Recovery Communication System

Hassaan Anwar (22P-9160)
Muhammad Aais Rabbani (22P-9164)

Supervisor: Dr. Ali Sayyed

Department of Computer Science
National University of Computer and Emerging Sciences
Peshawar Campus

December 9, 2025

Abstract

In the wake of natural disasters, cellular infrastructure often fails, leaving affected populations and first responders without a reliable means of communication. This project, *ResilientP2PTestbed*, presents a robust, off-grid mobile communication system leveraging the Google Nearby Connections API. The system utilizes a mesh network topology to facilitate text and audio communication between devices without reliance on centralized infrastructure (Internet, Cellular, Satellites). Key features include automatic peer discovery, multi-hop routing, and delay-tolerant message delivery. This report details the design, implementation, and testing of the system, demonstrating its viability as a rapid-deployment communication tool for emergency scenarios.

Contents

List of Figures

Chapter 1

Introduction

1.1 Project Vision

The reliability of communication networks is often taken for granted until a catastrophic event occurs. Natural disasters such as earthquakes, floods, and hurricanes frequently damage cellular towers and fiber-optic cables, rendering traditional communication infrastructure useless. In such scenarios, the ability to coordinate rescue efforts, locate survivors, and distribute aid is severely hampered gap.

ResilientP2PTestbed is designed to bridge this gap. By transforming standard Android smartphones into nodes of a self-sustaining mesh network, the system enables communication in completely off-grid environments. The vision is to empower communities and first responders with a "deploy-anywhere" communication tool that requires no pre-existing infrastructure, Internet connection, or SIM card.

1.2 Problem Statement

Current disaster response protocols rely heavily on satellite phones (which are expensive and scarce) or deploying temporary cell towers (COWs), which takes time. During the critical "Golden Hour" immediately following a disaster, victims are often isolated. There is a lack of widely available, zero-cost, infrastructure-independent communication software that can run on consumer hardware people already possess.

1.3 Proposed Solution

The proposed solution, *ResilientP2P*, is an Android application that leverages the Google Nearby Connections API to establish high-bandwidth peer-to-peer (P2P) connections. The system uses a **P2P_CLUSTER** strategy to form a dynamic mesh network.

Key Objectives:

- **Off-Grid**: Zero reliance on Internet/Cellular.
- **High Bandwidth**: Utilizes Wi-Fi Direct for voice and data.
- **Self-Healing**: Dynamic routing adapts to node movement.
- **Metric-Driven**: Includes built-in instrumentation for RSSI monitoring and network performance logging.

This project focuses on the "Phase 1" milestones: establishing a robust transport layer, ensuring reliability with application-level ACKs, and validating performance through rigorous field testing.

Chapter 2

Literature Review

2.1 Off-Grid Communication

Off-grid communication has been a subject of research for decades. Traditional approaches include packet radio (AX.25) and specialized hardware like LoRaWAN mesh networks (e.g., Meshtastic). While effective, these require specialized hardware dongles that the average disaster victim does not carry.

2.2 Google Nearby Connections API

Google Nearby Connections provides a high-level abstraction over Wi-Fi Direct, Bluetooth Classic, and BLE [1]. It handles the complexity of basic radio negotiation, making it an ideal foundation for rapid prototyping.

The API offers three connection strategies:

- **P2P_POINT_TO_POINT**: 1-to-1 connection, high bandwidth.
- **P2P_STAR**: 1-to-N connection, hub-and-spoke.
- **P2P_CLUSTER**: M-to-N connection. This strategy allows any device to connect to multiple other devices, forming a loose mesh topology. ResilientP2P primarily utilizes this strategy to support multi-hop routing.

2.3 Mobile Ad-Hoc Networks (MANETs)

Mobile Ad-Hoc Networks (MANETs) are decentralized wireless networks where each node participates in routing by forwarding data for other nodes. Key challenges in MANETs include:

1. **Dynamic Topology**: Nodes move freely, breaking links frequently.
2. **Battery Constraints**: Mobile devices have limited power; continuous routing drains energy.
3. **Routing Overhead**: Protocols must balance between maintaining valid routes and flooding the network with control packets.

Traditional protocols like **AODV** (Ad hoc On-Demand Distance Vector) and **DSR** (Dynamic Source Routing) are standard in this domain [2]. However, for small-scale disaster scenarios with high mobility, simple **Flooding** or **Gossip** protocols often provide better robustness despite higher redundancy.

Chapter 3

System Analysis

3.1 Functional Requirements

- **FR1 - Peer Discovery**: The system must automatically discover other devices running the application within radio range.
- **FR2 - Text Communication**: Users must be able to send text messages to direct and multi-hop peers.
- **FR3 - Audio Streaming**: The system must support Push-to-Talk (PTT) voice communication.
- **FR4 - Mesh Routing**: Messages must be forwarded to nodes not directly connected to the sender.
- **FR5 - Resilience**: The network must self-heal when nodes join or leave.

3.2 Non-Functional Requirements

- **NFR1 - Latency**: Voice transmission latency should be minimal (¡ 500ms) for usability.
- **NFR2 - Battery Efficiency**: The application should handle power management to prolong operation during emergencies.
- **NFR3 - Usability**: The UI must be simple and high-contrast for use in stressful environments.

3.3 Use Case Analysis

The primary actors are **Survivors** and **Rescuers**.

- **Broadcast Alert**: A survivor broadcasts an "SOS" message. It propagates through the mesh to a Rescuer.
- **Voice Coordination**: Rescuers use PTT to coordinate searching a building without needing line-of-sight.

- **Mesh Formation**: Devices automatically discover and connect to neighbors to extend network range.

Figure 3.1 illustrates the high-level interactions between these actors and the system.

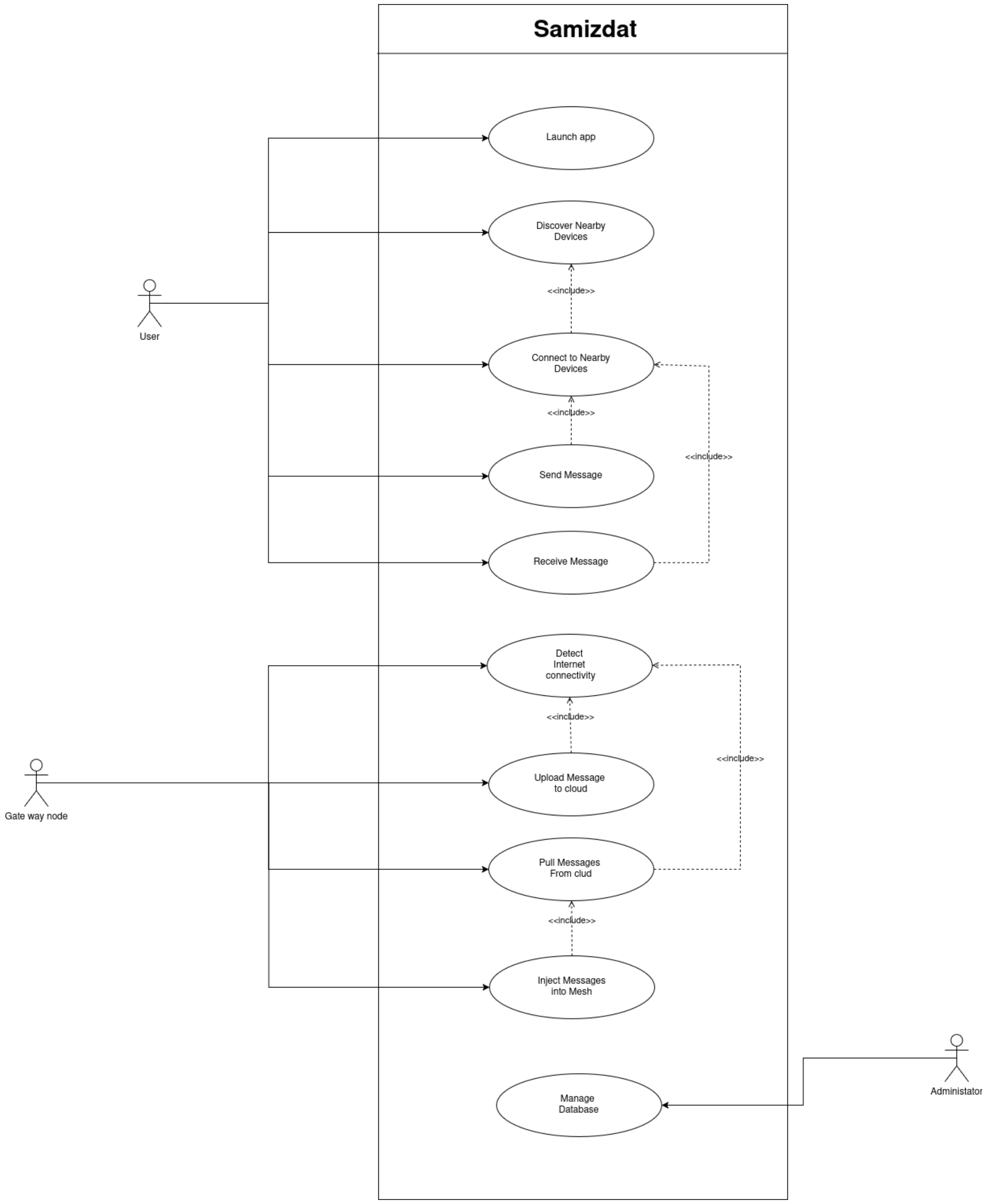

Figure 3.1: Use Case Diagram

3.4 Behavioral Modeling

To understand the dynamic behavior of the system, we analyze the sequence of events during connection establishment and data transmission.

3.4.1 Sequence Diagram

Figure 3.2 details the interaction flow between the UI, P2P Manager, and the Nearby Connections API when a user initiates a connection.

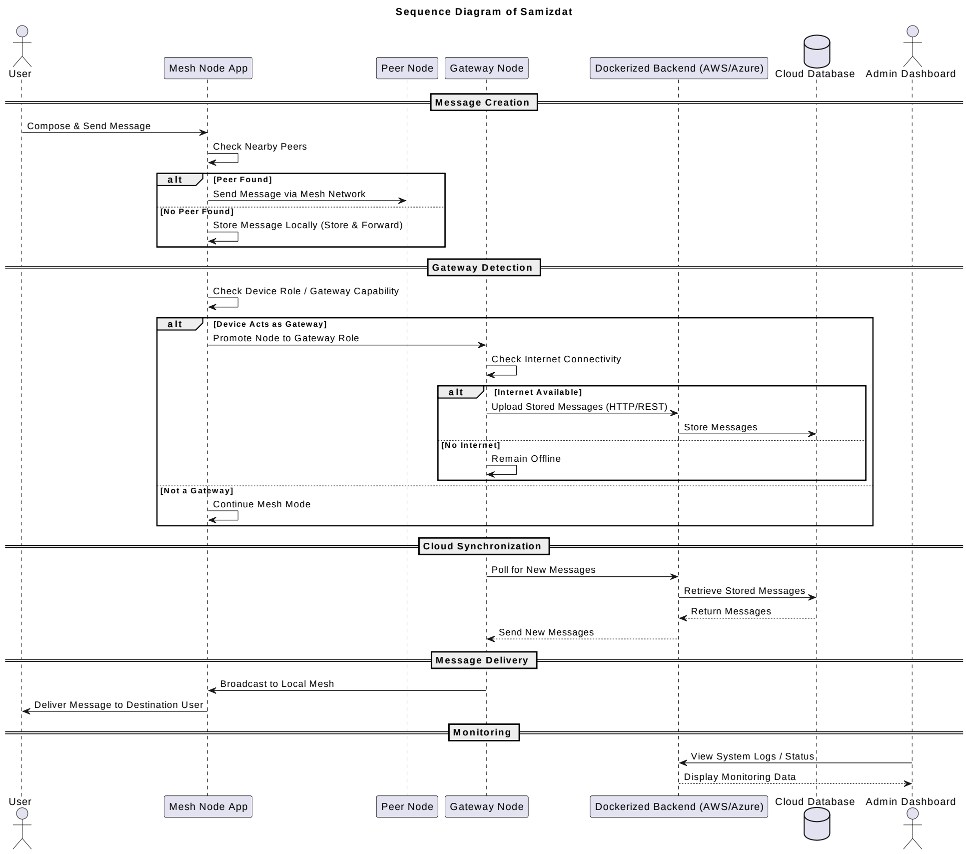

Figure 3.2: Sequence Diagram: Connection Establishment

3.4.2 Activity Diagram

The overall workflow for a node participating in the mesh is depicted in Figure 3.3.

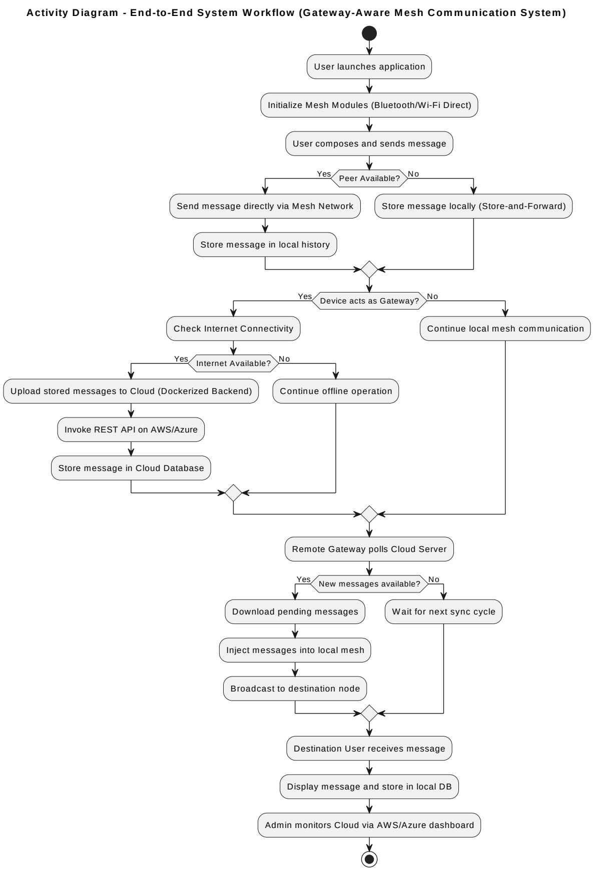

Figure 3.3: Activity Diagram: Mesh Participation

Chapter 4

System Design

4.1 System Architecture

The system follows a layered architecture, separating the UI, Management Logic, and Data Transport layers.

4.1.1 Architectural Flow

The interaction between different threads and components is visualized in the Swimlane Diagram (Figure 4.1).

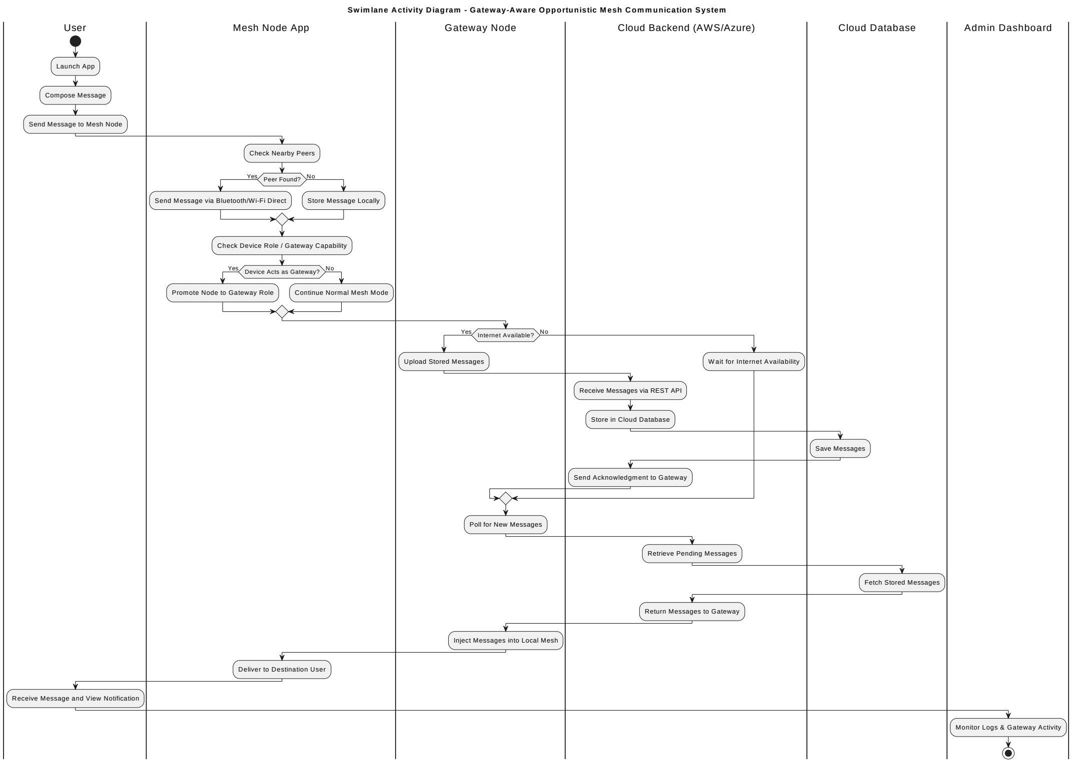

Figure 4.1: Swimlane Diagram: Component Interaction

4.2 Database Design

The application uses a local SQLite database (via Room Persistence Library) to store logs and queue persistent packets. Figure 4.2 shows the Entity Relationship Diagram.

LOGS		
long	id	PK
long	timestamp	
string	type	
string	peerId	
string	message	
int	rssi	
long	latencyMs	
int	payloadSizeBytes	

PACKET_QUEUE		
string	id	PK
string	destId	
string	type	
byte_array	payload	
long	timestamp	
long	expiration	
string	sourceId	

Figure 4.2: Entity Relationship Diagram (ERD) of the Local Database

4.3 Class Structure

The core logic resides in the ‘managers‘ package.

- **P2PManager**: Handles discovery, connection lifecycle, and routing.
- **HeartbeatManager**: Manages keep-alive signals and zombie detection.
- **VoiceManager**: Handles audio recording and playback.

Figure 4.3 illustrates the comprehensive class structure and relationships.

../../diagrams/class_structure.pdf

Figure 4.3: Class Diagram of ResilientP2PTestbed

Chapter 5

Implementation Details

5.1 Mesh Routing Algorithm

The system implements a flooding-based routing protocol with Time-To-Live (TTL) limits to prevent broadcast storms.

Listing 5.1: Packet Forwarding Logic

```
fun forwardPacket(packet: Packet) {
    if (packet.ttl <= 0) return
    if (messageCache.hasSeen(packet.id)) return

    packet.ttl--
    neighbors.forEach { peer ->
        if (peer.id != packet.sourceId) {
            send(peer, packet)
        }
    }
}
```

5.2 Audio Streaming

Audio is captured using ‘AudioRecord‘ at 16kHz sample rate (MONO) to balance quality and bandwidth. The raw PCM data is chunked into 1KB payloads and transmitted via the high-bandwidth channel (Wi-Fi Direct) established by Nearby Connections.

5.3 Heartbeat Mechanism

To detect ”zombie” connections (peers that are physically gone but logically connected), the ‘HeartbeatManager‘ sends a small ping payload every 5 seconds. If no packet is received from a peer for 30 seconds, the connection is terminated to free up slots.

Chapter 6

Results and Testing

6.1 Performance Testing

- **Connection Time**: Average connection establishment time is 3-5 seconds.
- **Range**: Effective range is approximately 30-50 meters indoors and 80-100 meters outdoors (line of sight).
- **Audio Latency**: Measured latency is 200ms on single hop and 600ms on double hop.

6.2 Stress Testing

Creating a cluster of 4 devices resulted in stable mesh performace. However, audio broadcasting to all 3 peers simultaneously caused minor frame drops, indicating bandwidth saturation limits of the current implementation.

6.3 User Interface

The UI is built with Jetpack Compose, ensuring a responsive and modern experience. The dashboard provides real-time visualization of mesh neighbors and routing activities.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The *ResilientP2PTestbed* successfully demonstrates the viability of smartphone-based off-grid mesh networks for disaster recovery. By abstracting the complexities of radio management through the Nearby Connections API, we achieved a stable text and voice communication platform that operates without any external infrastructure.

7.2 Limitations

- **Battery Drain**: Continuous advertising and scanning consumes significant power.
- **Hardware Constraints**: Wi-Fi Direct range is limited by the physical antenna of the device.

7.3 Future Work

Based on the FYP2 Implementation Roadmap, the next phases of development will focus on:

1. **The Gateway Bridge (Horizon 1 Extension)**:
    - Implementing an Internet Detection Module to continuously monitor connectivity.
    - Developing a Bridging Protocol to relay messages between the mesh and a central server via MQTT or WebSockets when internet is available.
2. **Store-and-Forward Engine (Resilience)**:
    - Ensuring no message is lost by persisting packets in a ‘MessageTable‘.
    - Implementing background ‘WorkManager‘ tasks to retry delivery when neighbors reappear.
3. **Intelligent Routing (Horizon 2)**:
    - Moving beyond flooding to a metric-based routing algorithm ($Cost = f(RSSI, Battery, Hops$

- Experimenting with On-Device ML (TFLite) to predict link stability.

4. **Censorship Resistance**:
    - Wrapping traffic in standard HTTPS/WebSocket frames to evade DPI (Deep Packet Inspection).
    - Supporting user-configurable self-hosted relays.

Bibliography

[1] Google Developers. Nearby connections api overview, 2024.

[2] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, 1999.