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 10, 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

1 Introduction **3**
1.1 Project Vision 3
1.2 Problem Statement 3
1.3 Proposed Solution 3

2 Literature Review **5**
2.1 Off-Grid Communication 5
2.2 Google Nearby Connections API 5
2.3 Mobile Ad-Hoc Networks (MANETs) 5

3 System Analysis **7**
3.1 Functional Requirements 7
3.2 Non-Functional Requirements 7
3.3 Use Case Analysis 7
3.4 Behavioral Modeling 8
3.4.1 Sequence Diagram 8
3.4.2 Activity Diagram 9

4 System Design **11**
4.1 System Architecture 11
4.1.1 Architectural Flow 11
4.2 Database Design 12
4.3 Class Structure 13

5 Implementation Details **15**
5.1 Mesh Routing Algorithm 15
5.1.1 Metric-Based Scoring 15
5.1.2 Self-Poisoning Prevention 15
5.2 State Management 15
5.3 Audio Streaming 16
5.4 Heartbeat Mechanism 16

6 Results and Testing **17**
6.1 Performance Testing 17
6.2 Stress Testing 17
6.3 User Interface 17

7 Conclusion and Future Work 18
7.1 Conclusion 18
7.2 Limitations 18
7.3 Future Work 18

List of Figures

3.1 Use Case Diagram 8
3.2 Sequence Diagram: Connection Establishment 9
3.3 Activity Diagram: Mesh Participation 10

4.1 Swimlane Diagram: Component Interaction 12
4.2 Entity Relationship Diagram (ERD) of the Local Database 13
4.3 Class Diagram of ResilientP2PTestbed 14

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.

```
../../../../diagrams/output/use_case_diagram.png
```

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.

`../../../../diagrams/output/sequence_diagram.pdf`

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.

../../../../diagrams/output/activity_diagram.pdf

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).

../../../../diagrams/output/swimlane_diagram.pdf

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.

`../../../../diagrams/output/database_schema.pdf`

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/output/class_structure.pdf

Figure 4.3: Class Diagram of ResilientP2PTestbed

Chapter 5

Implementation Details

5.1 Mesh Routing Algorithm

The system has evolved from simple flooding to a **Metric-Based Routing** protocol. While it still uses flooding for discovery, it maintains a routing table optimized for stability.

5.1.1 Metric-Based Scoring

Routes are evaluated based on a score derived from the Time-To-Live (TTL). A higher TTL indicates fewer hops, resulting in a higher score.

$$Score = TTL \tag{5.1}$$

When a packet arrives, the router compares the new score with the existing route's score. The routing table is updated only if the new route is strictly better ($NewScore > CurrentScore$), preventing route flapping.

5.1.2 Self-Poisoning Prevention

To prevent loops where a node learns a route to itself via a neighbor, the system inspects the 'sourceId' of incoming packets. If 'packet.sourceId == localId', the packet is dropped immediately, and the route is not updated.

Listing 5.1: Route Update Logic

```
if (packet.sourceId != localUsername) {
    val newScore = packet.ttl
    if (newScore > currentScore) {
        routingTable.put(packet.sourceId, sourceEndpointId)
        routingScores[packet.sourceId] = newScore
    }
}
```

5.2 State Management

The application state is managed by a reactive 'P2PState' data class, exposed via 'StateFlow'. This ensures UI consistency across the Composable hierarchy.

Key state fields include:

- **isHybridMode**: Toggles between pure P2P and Gateway-assisted modes.
- **isLowPower**: Reduces discovery frequency to save battery.
- **connectedEndpoints**: List of currently active physical connections.
- **logs**: A rolling buffer of the last 100 log entries for on-device debugging.

5.3 Audio Streaming

Audio is captured using ‘AudioRecord‘ at 16kHz sample rate (MONO). The raw PCM data is chunked into payloads and transmitted via the high-bandwidth Wi-Fi Direct channel. The system supports both Unicast (1-to-1) and Broadcast (1-to-All) streaming.

5.4 Heartbeat Mechanism

To detect "zombie" connections, the ‘HeartbeatManager‘ sends a ping every 5 seconds. If no packet is received for 15 seconds (reduced from 30s), the peer is marked as stale. Direct PINGs are used to verify physical link integrity independent of the mesh overlay.

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.