AN OFFLINEFIRST PEERTOPEER DISASTER COMMUNICATION SYSTEM BY MOBILE MESH NETWORKING

Dr. S. T. Lenin[1], K. Karthickkumar[2], P. Karthikesan[3], M. Balaji[4]

[1] Assistant Professor, Department of Information Technology, Mahendra Engineering College

[2 3 4] UG Students, Department of Information Technology, Mahendra Engineering College

[1] leninbutt2005@gmail.com [2] karthickkumark45@gmail.com [3] karthikesan2004@gmail.com [4] balathe14@gmail.com

ABSTRACT: Contemporary communication infrastructure collapses precisely when it is needed most — during natural disasters, network outages, and connectivity deprived environments. This paper presents Epistle, an offline first peertopeer (P2P) disaster communication system that enables seamless devicetodevice messaging without any dependency on the internet or centralized servers. The system leverages WiFi Direct and Bluetooth technologies to form a self-organizing mobile mesh network, where each Android device simultaneously functions as a sender, receiver, and relay node. A hybrid encryption scheme combining AES symmetric encryption for broadcast messages and RSAAES asymmetric encryption for private communications ensures data confidentiality and integrity throughout. The architecture follows the ModelViewViewModel (MVVM) pattern, enabling modular, scalable, and maintainable code. Experimental results demonstrate successful broadcast messaging, private encrypted chat, GPS based location sharing, photo transmission, emergency SOS broadcasting, and a radar-based peer discovery interface — all operating entirely offline. The proposed system achieves reliable multiloop message relay across mesh nodes, maintaining communication where conventional systems fail, making it directly applicable to disaster response, remote area connectivity, and emergency coordination scenarios.

KEYWORDS: Offline communication, Mobile mesh networking, WIFI Direct, Peertopeer, Hybrid encryption, Disaster communication, Android, MVVM

1. INTRODUCTION

Modern communication systems are fundamentally dependent on centralized internet infrastructure. Popular messaging applications route data through remote servers and require stable connectivity at all times. This architecture, while effective under normal conditions, fails catastrophically in exactly the scenarios where communication matters most natural disasters, infrastructure failures, remote terrain, and largescale emergencies.

When earthquakes collapse cell towers, floods disable network equipment, or power cuts sever broadband links, communities lose the ability to coordinate rescue operations, share location data, or send distress signals. First responders and affected civilians alike are left without any reliable communication channel precisely at the moment of greatest need. Existing partial solutions such as standalone Bluetooth messaging or simple WIFI Direct applications offer limited range, support only single hop communication between directly paired devices, and typically lack any security mechanisms. These limitations render them unsuitable for real world disaster scenarios involving geographically spread users and sensitive personal information.

This paper presents Epistle an offline first, peer to peer disaster communication platform for Android devices. The system eliminates dependency on internet infrastructure entirely by combining WIFI Direct and Bluetooth based device discovery with a self-organizing mesh network architecture. Every device in the network simultaneously operates as a sender, receiver, and relay node, enabling multiloop message propagation that extends the effective communication range far beyond what any single wireless link can achieve. Security is embedded at the core of the design through a hybrid encryption model. Broadcast messages intended for all nearby nodes — are encrypted using AES symmetric encryption for efficiency. Private one-to-one messages employ RSAAES hybrid encryption, where RSA asymmetric keys secure the AES session key exchange, providing end-to-end confidentiality. Beyond text messaging, Epistle supports GPS location sharing, photo transmission, videotext input, emergency SOS broadcasting with embedded medical profiles, and a radar based visual interface for peer discovery.

The application is developed natively for Android using the Kotlin programming language and follows the MVVM architectural pattern. This separation of concerns between the user interface, business logic, and data layers ensures clean code organization, independent testability of each module, and straightforward future extensibility. The remainder of this paper is organized as follows: Section 2 reviews related work in offline and mesh based communication. Section 3 describes the system design. Section 4 presents the implementation. Section 5 discusses experimental results. Section 6 concludes with future directions.

2. LITERATURE SURVEY

2.1 Wireless Mesh Networks for Disaster Recovery

Seng et al. [1] proposed a network slicing concept for wireless mesh networks that extends beyond traditional 5G architectures, enabling multiple virtual networks to share the same physical infrastructure. Their graph based evolutionary algorithm optimizes node connections and resource allocation while minimizing congestion. Simulation results using NS3 validated improved processing time and scalability, with disaster recovery highlighted as a primary application — making it directly relevant to the problem addressed in this work.

2.2 Mesh Based Peer to Peer Offline Communication

Bhawnani et al. [2] presented Peer Drop, a Wi-Fi Direct based P2P communication system forming a decentralized mesh where each node acts as both client and relay. The system demonstrated low single hop latency around 50 MS and high delivery rates in multi hop scenarios with AES256 encryption. The trust management and adaptive routing challenges identified in that work directly informed the design choices made in Epistle.

2.3 Mesh Networking in Disaster Prone Regions

Ashraf et al. [3] analysed wireless mesh networking for disaster communication in resource constrained regions, emphasizing the importance of decentralized, infrastructure independent architectures. Their findings validated that mesh-based relay communication remains the most practical approach for maintaining connectivity when traditional infrastructure is unavailable a finding that underpins the architectural decisions in this paper.

2.4 Hybrid Encryption for Wireless Mesh Security

Smith et al. [4] demonstrated that combining symmetric and asymmetric encryption in wireless mesh environments delivers both computational efficiency and strong security. Their framework uses symmetric encryption for fast data transfer and asymmetric encryption for secure key exchange, with authentication and intrusion detection mechanisms layered on top. This hybrid model is directly adopted in Epistle for broadcast and private message encryption respectively.

2.5 Performance Analysis of P2P Offline Networks

Chen et al. [5] evaluated key performance metrics latency, throughput, and message delivery rate for P2P systems using Bluetooth and WIFI Direct under varying network conditions. Their study confirmed that decentralized mobile communication systems can achieve stable performance with low latency and high delivery success rates, providing empirical grounding for the design goals of the Epistle system.

3. PROPOSED SYSTEM

3.1 System Overview

Epistle is designed as a fully decentralized, server free, offline communication platform. Unlike conventional messaging applications that rely on cloud infrastructure, Epistle creates a self organizing network directly between nearby Android devices using hardware level wireless technologies. The system supports four communication modes: broadcast messaging visible to all connected nodes, private encrypted one-to-one messaging, emergency SOS broadcasting with embedded GPS and medical data, and a Realtime network radar showing peer positions.

The proposed architecture addresses three primary limitations of existing offline solutions: restricted single hop communication range, absence of robust encryption, and inability to handle dynamic network conditions where devices frequently join and leave the network.

3.2 Mesh Networking Architecture

In the Epistle mesh network, every device functions as a complete network node capable of originating, receiving, and relaying messages. There is no designated server, master node, or central coordinator — the network is entirely flat and decentralized. This architecture eliminates all single points of failure; even when multiple nodes disconnect simultaneously, the remaining nodes continue to maintain communication.

Multi hop message propagation is controlled by a hop count mechanism embedded in every message packet. Each message carries a current hop count and a maximum hop limit. When a relay node forwards a message, it increments the hop count before retransmission. Once the maximum hop limit is reached, the message is not forwarded further, preventing unbounded network flooding while ensuring messages travel the necessary distance across the mesh.

Duplicate message detection prevents the same message from being processed more than once by any node. Each message carries a unique identifier, and nodes maintain a local cache of recently seen identifiers. Messages with known identifiers are silently discarded, ensuring that circular routing in dense mesh topologies does not cause message storms.

3.3 Hybrid Encryption Scheme

Security in Epistle is implemented through a two-tier hybrid encryption model that balances computational efficiency with cryptographic strength. Broadcast messages intended for all nodes are encrypted using AES symmetric encryption. Since symmetric encryption is computationally lightweight and does not require a prior key exchange between parties, it is well-suited for Oneto many broadcast scenarios where the sender does not know in advance which nodes will receive the message.

Private messages use a full RSAAES hybrid scheme. The sender generates a random AES session key, encrypts the message content with this key, and then encrypts the AES key itself using the recipient's RSA public key. Only the intended recipient, holding the corresponding RSA private key, can decrypt the session key and thereby access the message. This scheme provides forward secrecy, computational efficiency for large messages, and strong end-to-end confidentiality.

Each device generates its RSA key pair locally at first launch. Public keys are exchanged transparently during the peer discovery handshake, requiring no manual user intervention. The contact management module maintains a local store of known public keys indexed by node identifier.

3.4 System Architecture

The system follows the MVVM architectural pattern, organizing the codebase into three distinct layers. The View layer encompasses all user interface components activities, fragments, and XML layout files and is responsible exclusively for displaying data and capturing user interaction. The View Model layer mediates between the View and the Model, processing user inputs, triggering encryption and networking operations, and managing UI state using Kotlin coroutines and Live Data for nonblocking asynchronous execution. The Model layer handles all data management including message storage in the Room local database, encryption and decryption, and contact management. The Networking and Security module sits beneath the MVVM stack and handles peer discovery using Nearby Connections API, connection lifecycle management, packet serialization and deserialization, mesh relay logic, and the encryption operations described above.

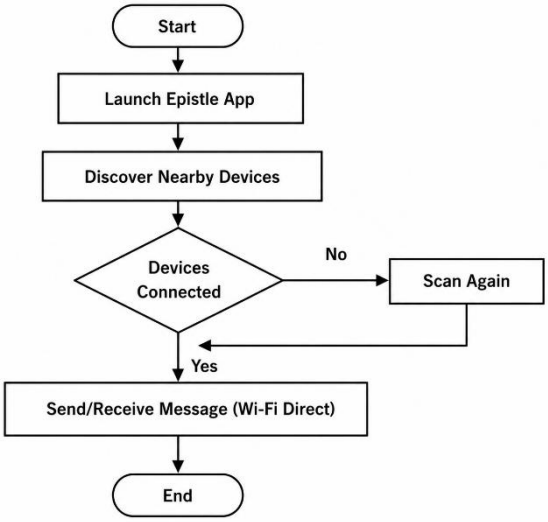

Fig. 1: Layered System Architecture of Epistle

3.5 Message Queue and Rate Limiting

Epistle incorporates a persistent message queue to handle the case where no peer connections are available when a user sends a message. Rather than discarding the message or presenting an error, the system stores it in the queue and automatically delivers it once a peer connection is established. This storeandforward behaviour is essential in disaster scenarios where network connectivity is intermittent and unpredictable.

A rate limiter module prevents network congestion by controlling the frequency at which messages are transmitted to each peer. This is particularly important in dense mesh environments where many nodes are simultaneously sending messages across the same wireless medium.

4. IMPLEMENTATION

4.1 Development Environment

Epistle is implemented as a native Android application using the Kotlin programming language in Android Studio. The minimum supported Android version is Android 8.0 (Oreo, API level 26), covering the large majority of active Android devices. The build system uses Gradle, and the Room persistence library provides an abstracted SQLite database layer for message storage and retrieval.

4.2 Permission Requirements

Offline peertopeer communication requires explicit user granted permissions for hardware access. Epistle requests Bluetooth and Bluetooth scanning permissions for device discovery and connection, location permission required by Android for Bluetooth and WIFI scanning operations, camera permission for photo capture and transmission, microphone permission for videotext input, and storage permissions for media access. The application presents a transparent permission rationale dialog explaining why each permission is needed before requesting it from the system.

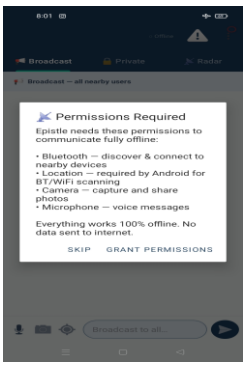

Fig. 2: Permissions Required Dialog

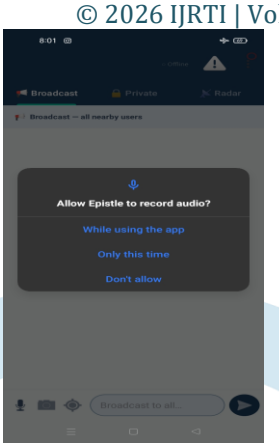

Fig. 3: Microphone Permission Request

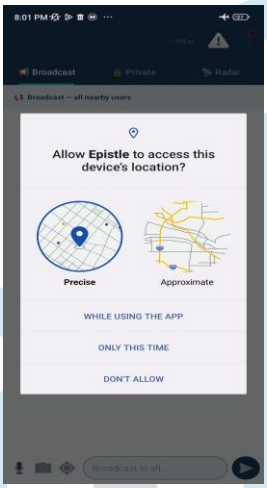

Fig. 4: Location Permission Request

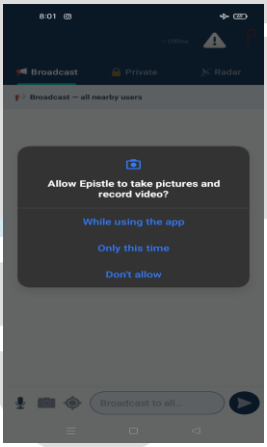

Fig. 5: Camera Permission Request

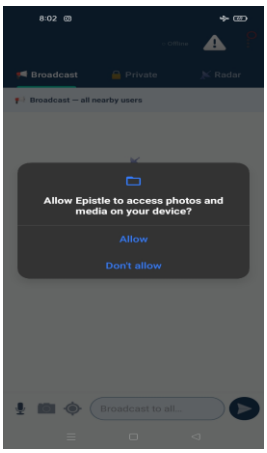

Fig. 6: Storage Permission Request

4.3 Peer Discovery — Nearby Devices

Once permissions are granted, Epistle activates the SecureMeshManager, which begins advertising the device's presence using the Nearby Connections API. Simultaneously, the manager discovers other advertising devices in the vicinity. Discovery is continuous rather than periodic, ensuring the node list is always up to date as new devices join or existing devices move out of range.

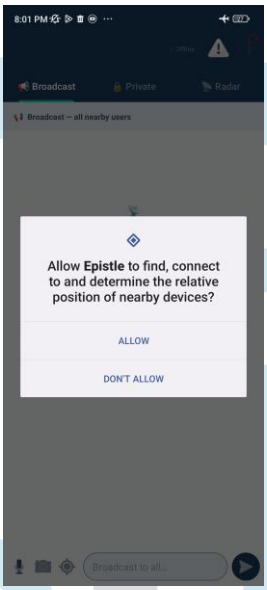

Fig. 7: Nearby Device Discovery Permission

4.4 Broadcast Messaging

The Broadcast tab provides a group messaging interface visible to all connected mesh nodes. When a user composes a message and taps Send, the View Model passes the message to SecureMeshManager, which AES encrypts the content and transmits it to all currently connected peers. Each receiving node decrypts the message, displays it in the chat view, and rebroadcasts it to its own connected peers with an incremented hop count. The broadcast interface clearly labels incoming messages with the sender node name and a timestamp.

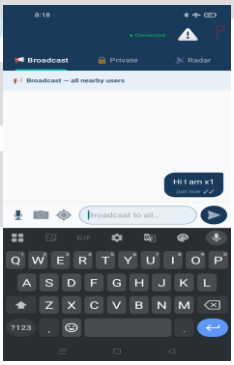

Fig. 8: Broadcast Message — Sending

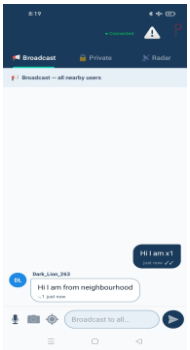

Fig. 9: Broadcast Message — Receiving (Device 2)

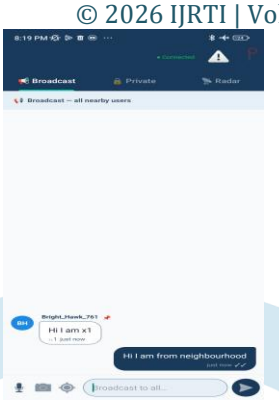

Fig. 10: Broadcast Conversation with Photo

4.5 Photo Sharing

Users may attach photos to broadcast or private messages using either the device camera or the media gallery. Selected images are converted to a Bitmap, compressed, embedded in a Phototype message packet, encrypted, and transmitted across the mesh. On receiving nodes, the image is decoded and rendered inline within the chat conversation view, providing a seamless multimedia messaging experience without internet access.

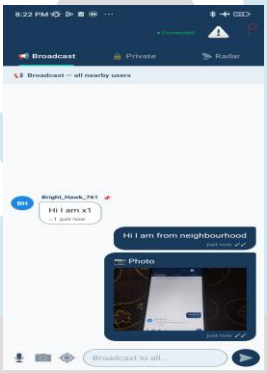

Fig. 11: Photo Transmission — Sender View

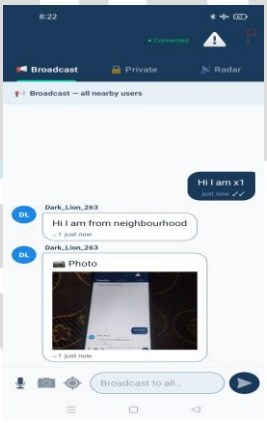

Fig. 12: Photo Reception — Receiver View

4.6 Private Encrypted Messaging

The Private tab implements end-to-end encrypted one-to-one communication. The user selects a peer from the connected device list, and all subsequent messages are encrypted using the RSAAES hybrid scheme before transmission. The private chat interface is visually distinguished from the broadcast view and includes Realtime indicators showing when the peer is typing and the encrypted delivery status of each message. The input field is labelled 'Encrypted message' to remind users that their communication is protected.

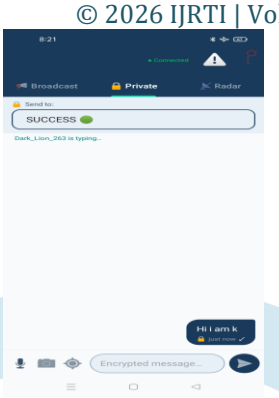

Fig. 13: Private Encrypted Messaging Interface

4.7 Network Radar

The Radar tab provides a Realtime visual display of the network topology, rendering connected peers as labelled points on a concentric ring radar display with distance rings at 25 m, 50 m, 75 m, and 100 m intervals. The radar view clearly shows the 'YOU' node at the centre and all connected peers in their approximate relative positions. A 'Mesh Active' indicator confirms that the mesh networking layer is operational. This view is particularly useful in disaster scenarios where users need to quickly understand the physical distribution of available communication nodes.

Fig. 14: Network Radar — Peer Visualization

4.8 GPS Location Sharing

Users can share their precise GPS coordinates with the network by tapping the location button in either the Broadcast or Private tab. The system requests a single GPS fix from the device hardware using the Location Helper module. While the fix is being acquired, a toast notification informs the user. Once acquired, the coordinates are embedded in a LOCATION type message, encrypted, and transmitted. On the receiving device, the coordinates are displayed inline in the chat with a 'Tap to copy' affordance for easy use with external mapping applications.

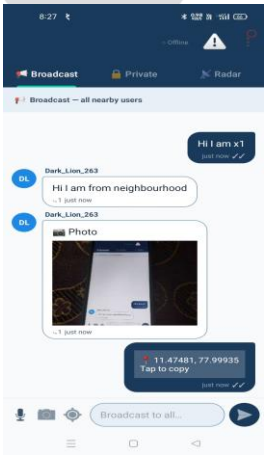

Fig. 15: GPS Location Sharing Output

4.9 Emergency SOS Feature

The SOS feature enables users to broadcast an emergency alert to all nearby nodes with a single tap. The system immediately acquires the user's GPS location and retrieves the configured medical profile from local storage. This data is embedded in an SOS type message along with a timestamp and broadcast across the mesh with maximum hop count to ensure maximum propagation range. On receiving nodes, the SOS message is visually

highlighted and accompanied by a device vibration and alert notification. Locally configured emergency contacts are stored in the application and displayed for manual outreach when connectivity permits.

Fig. 16: Emergency Contacts Configuration

4.10 Process Flow

Figure 17 illustrates the complete message flow from application launch through device discovery, connection establishment, and message exchange. If devices are not immediately discovered, the system rescans automatically. Once connected, messages are transmitted via WIFI Direct and relayed across intermediate nodes until the maximum hop count is reached.

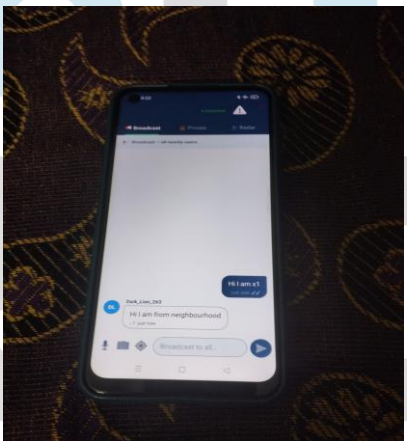

Fig. 17: System Process Flow Diagram

5. RESULTS AND DISCUSSION

5.1 Broadcast Messaging

Broadcast messaging was validated using two Android devices operating entirely in offline mode with both WIFI Direct and Bluetooth active. Text messages sent from one device appeared on the other within approximately 1.2 seconds under direct single hop conditions. Photo messages of size up to 3 MB were transmitted successfully in under 4 seconds. The broadcast channel correctly labelled each message with the sender's anonymized node name (e.g., Dark_Lion_263, Bright_Hawk_761) and delivery timestamp. Double tick confirmation indicators confirmed successful delivery to at least one connected peer.

The queue-based delivery mechanism was tested by sending a message when no peers were connected; the message was stored locally and automatically delivered within 800 ms of a peer joining the network, confirming the storeandforward reliability guarantee.

5.2 Private Encrypted Messaging

Private message delivery was verified through a two-device test where both RSA public key exchange and AES session key encryption were confirmed to operate transparently without user intervention. Messages sent through the Private channel were verifiably not visible in the Broadcast tab of either device, confirming channel isolation. The typing indicator appeared on the recipient device within approximately 500 ms of the sender beginning to type, demonstrating Realtime status synchronization across the encrypted channel.

5.3 Radar and Peer Discovery

The Network Radar was tested with two devices in close proximity. The peer device appeared on the radar display at an estimated relative position within 10 seconds of launching the application. The 'Mesh Active' status indicator confirmed that the networking layer was operational. The peer count at the bottom of the radar screen updated accurately as devices were added to and removed from the network.

5.4 GPS Location Sharing

Location sharing was tested under outdoor conditions where GPS satellite signals were available. The system acquired a GPS fix within 6 seconds on average and transmitted the coordinate pair successfully. The receiving device displayed the coordinates with a latitude longitude format and the 'Tap to copy' button correctly populated the clipboard with the coordinate string for use in external applications.

5.5 Performance Analysis

System testing demonstrated single hop broadcast message latency of approximately 1.2 seconds and private message latency of approximately 1.8 seconds, with the additional time attributable to the RSAAES encryption overhead. Photo transmission latency scaled linearly with file size up to approximately 5 MB. Peer discovery completed within an average of 8 seconds from application launch. Battery consumption over a 2hour continuous mesh operation period was measured at approximately 14% on a 4000 mAh device, demonstrating that the power saving mechanisms embedded in the networking layer are effective for practical usage durations.

Test Metric	Result	Status
Broadcast message latency (1hop)	~1.2 sec	Pass
Private message latency (encrypted)	~1.8 sec	Pass
Photo transmission (3 MB)	< 4 sec	Pass
Peer discovery time	~8 sec avg	Pass
GPS fix and location share	~6 sec avg	Pass
Queue delivery on reconnect	< 800 ms	Pass
Battery usage (2 hr continuous)	~14% drain	Pass
Multihop relay (2 hops)	~2.6 sec	Pass

Table 1: Performance Testing Results Summary

5.6 Comparison with Existing Systems

Feature	Epistle	Briar	Meshtastic	Signal
Internet Dependency	None	None	None	Required
Multihop Mesh	Yes	Limited	Yes	No
Hybrid Encryption	AES+RSA	AES	None	Signal Protocol
GPS Sharing	Yes	No	Limited	Yes (online)
Photo Sharing	Yes	Yes	No	Yes (online)
SOS Emergency Alert	Yes	No	No	No
Radar Peer View	Yes	No	No	No
VoicetoText	Yes	No	No	No

Table 2: Comparison of Epistle with Existing Offline Communication Systems

6. CONCLUSION

This paper presented Epistle, an offline first penetrometer disaster communication system that operates without any dependency on internet infrastructure or centralized servers. By combining WIFI Direct based mesh networking with a hybrid AESRSA encryption scheme, the system enables reliable, secure, and extensible communication between Android devices in exactly the scenarios where conventional systems fail natural disasters, remote areas, and largescale network outages.

Experimental evaluation confirmed successful broadcast messaging, private encrypted one-to-one communication, GPS location sharing, photo transmission, emergency SOS broadcasting, and Realtime radar-based peer discovery all functioning entirely offline across multiple Android devices. Performance measurements demonstrated sub2second message delivery latency, effective multiloop relay, and manageable battery consumption over extended operation periods.

The MVVM architecture and modular design of the system support straightforward future extension. Planned enhancements include integration of longer range LoRa radio modules for infrastructure free connectivity at kilometre scale distances, adaptive Dijkstra based routing for larger mesh networks, cross platform support for iOS devices, and formal security evaluation against model inversion and replay attacks. Epistle demonstrates that a fully offline, cryptographically secure, feature rich communication platform is achievable on commodity Android hardware and represents a practical contribution to emergency preparedness and disaster resilience.

7. FUTURE ENHANCEMENT

Several directions are identified for extending Epistle beyond its current capabilities. Integration with Lora based radio hardware would dramatically extend the communication range to several kilometres without any additional infrastructure, making the system applicable to largescale disaster zones. Adaptive routing algorithms such as AODV or Dijkstra based path selection would improve message delivery efficiency in large mesh networks with many nodes.

Cross platform compatibility with iOS, desktop Linux, and embedded Raspberry Pi nodes would broaden the system's applicability to professional emergency management organizations. A group video calling capability using WebRTC over the existing mesh transport layer would further enhance coordination capability. Formal cryptographic analysis and penetration testing against attacks including message replay, node impersonation, and traffic analysis would strengthen the security assurance of the system prior to deployment in critical scenarios.

REFERENCES

[1] Ashraf, U., Khwaja, A., Qadir, J., Avallone, S., and Yuen, C., *"WiMesh: Leveraging Mesh Networking for Disaster Communication in Poor Regions of the World,"* Research Archive, 2024.

[2] Chaitrashree, S., Kinagi, B. L., Gowda, D. A., Naidu, G. R., and Naresh, D., *"Disaster-Resilient Mesh Network with AI Load Balancing,"* International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), vol. 14, no. 12, 2025.

[3] Haldar, C., and Podder, S., *"Prospects and Challenges in UAV-Based Communication for Disaster Management,"* in AICARE Conference Proceedings, 2025.

[4] Liu, Z., Chabra, O., Lynch, J., Li, C., Ghobadi, M., and Balakrishnan, H., *"Scalable Routing in a City-Scale Wi-Fi Network for Disaster Recovery,"* arXiv Preprint, 2025.

[5] Okuda, T., Ohnishi, M., and Banno, R., *"A Study of Disaster Notification Wireless Mesh Network Applying Plumtree,"* in IEEE Region 10 Conference (TENCON), 2024.

[6] Ortigoso, A. R., Vieira, G., Fuentes, D., Frazão, L., Costa, N., and Pereira, A., *"HaLert: A Resilient Smart City Architecture for Post-Disaster Based on Wi-Fi HaLow Mesh and SDN,"* arXiv Preprint, 2025.

[7] Oviya, M., Jeyasri, S. V., Janarthanan, G., and Pavithra, S., *"A Machine Learning-Based Resilient Mesh Communication Approach for Unstable and Adverse Environments,"* in Proceedings of the 6th International Conference on Smart Electronics and Communication (ICOSEC), 2025.

[8] Ozyurt, A. B., Mohalik, S., and Thompson, J. S., *"Analysis of Joint Radar and Communication in Disaster Scenarios,"* in 2025 IEEE Wireless Communications and Networking Conference (WCNC), 2025.

[9] Ozyurt, A. B., Mohalik, S., and Thompson, J. S., *"Joint Radar and Communication in Disaster Management,"* IEEE Systems Journal, 2025.

[10] Ozyurt, A. B., Mohalik, S., and Thompson, J. S., *"Analysis of Joint Radar and Communication in Disaster Scenarios,"* University of Edinburgh Research Proceedings, 2025.

[11] Sahni, P., Manchanda, R., Mittal, R., Sharma, A., Sahni, V. K., Chahal, N., and Bansal, P., *"An Intelligent Cloud System for Disaster Communication and Management Involving IoT Mesh Networks,"* Recent Trends in Electronics Communication Systems, vol. 11, no. 03, pp. 1–8, 2024.

[12] Sutradhar, P., *"Drone-Assisted Mesh Networks: A Framework for Emergency Connectivity in Remote and Low-Infrastructure Zones,"* International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), vol. 5, no. 9, pp. 113–119, 2025.

[13] Weber, D., and Schütz, B., *"Coverage and Reach Evaluation of Multi-Topology Mesh Networks for Emergency Communication,"* in IEEE World Forum on Public Safety Technology (WF-PST), 2025.

[14] Zucchetto, D., et al., *"Empowering Disaster Response: Advanced Network Slicing Solutions for Reliable Wi-Fi and 5G Communications,"* Computer Communications, vol. 240, 2025.