ATAK Plugin for Decentralized Communication via FutureSDR

Junchi Weng
Technische Universität Darmstadt
junchi.weng@stud.tu-darmstadt.de

1 MOTIVATION

Currently, we live in an age defined by the interconnectedness of all things, where communication technologies evolve rapidly. Moreover, the advent of 5G technology signifies a pivotal breakthrough, providing faster data transmission, minimal delay, and extensive device connectivity. Such progress has greatly improved communication systems in multiple fields, particularly in areas like emergency rescue and disaster management. For rescue teams operating under time-sensitive and ever-changing conditions, their success largely relies on exchanging information promptly and ensuring the accuracy of data gathered at the scene.

Nevertheless, the settings where these teams work frequently pose significant challenges to standard communication infrastructures. Major catastrophes, including earthquakes, tsunamis, volcanic activity, or deliberate damage to infrastructure, often cause severe destruction or disablement of communication networks. Consequently, communication blackouts occur, where even satellite systems struggle to maintain stable connections due to physical obstructions or excessive network demand. An illustrative case is the 2022 eruption of Hunga Tonga-Hunga Ha'apai, which cut undersea fiber-optic cables and hindered satellite signals because of volcanic ash, isolating Tonga from global communications for multiple days.

Such events emphasize the critical requirement for communication solutions that do not rely on permanent infrastructure. Specifically, communication technologies that are compact, energy-efficient, and support direct device-to-device interaction provide essential robustness in these conditions. By facilitating direct messaging between devices and decentralized information sharing, such technologies can preserve crucial communication channels during network failures.

The primary objective of this paper is to address the aforementioned challenges by integrating and validating a prototype communication system. This system is designed to be infrastructure-independent and low-power, and its application within the Android Team Awareness Kit (ATAK), a mobile geospatial collaboration platform, will be thoroughly examined.

Specifically, the implementation leverages the open-source nature and plugin-based architecture of the ATAK platform to develop a dedicated communication plugin. This plugin features a loosely coupled design, enabling the seamless integration of existing physical-layer communication solutions, such as acoustic or LoRa technologies. Critically, the design allows for the flexible replacement of the underlying communication method based on operational requirements, all without necessitating modifications to the application-layer logic. This integration strategy ensures the system's capacity for reliable device-to-device communication even in dynamic and network-deprived environments.

The plugin-based integration scheme offers significant advantages, making the system particularly well-suited for disaster relief operations where communication infrastructure has suffered severe damage. By utilizing the powerful map visualization capabilities of the ATAK platform, all communication information and device locations can be presented in a unified and intuitive manner. This capability substantially enhances the situational awareness and collaborative efficiency of rescue teams. Furthermore, the proposed solution demonstrates broad versatility, with its applicability extending to specialized missions where conventional communication is often limited. Examples of such scenarios include firefighting, scientific expeditions, and remote medical support.

2 RELATED WORK AND BACKGROUND

In the specific context of disaster communication, Deepak G. and his collaborators, through their investigative research, introduced a classification methodology for scenarios based on the degree of damage sustained by communication networks. This classification encompasses congested networks, partially functional networks, and completely isolated networks.

A congested network describes a situation where the communication infrastructure remains operational; however, its performance significantly degrades due to a surge in communication demands. In such instances, mechanisms like enhanced Mobile Priority Services (eMPS) or network slicing can be implemented to ensure the prioritization of emergency communications. A partially functional network signifies that certain nodes within the communication infrastructure have failed, resulting in connectivity being maintained only in localized areas or for specific services. Conversely, a completely isolated network represents the most extreme scenario, characterized by the failure of all base stations and a complete loss of communication connectivity within the affected area. Under the conditions of partially functional or completely isolated networks, traditional centralized communication systems often prove ineffective, thereby necessitating autonomous and decentralized communication solutions.

In recent years, the field of emergency communication has witnessed the emergence of various technologies specifically suited for scenarios involving infrastructure failure. These include:

- **Device-to-Device communication**. This enables direct communication between end devices without requiring support from base stations.
- **Unmanned Aerial Vehicle relays**. These systems utilize UAVs equipped with communication devices to temporarily restore communication coverage in disaster-affected regions.
- **Mobile Ad-hoc Networks (MANETs)**. These networks are formed by mobile nodes that dynamically self-organize into network topologies, providing rapid and flexible wireless communication.

Additionally, LoRa technology has progressively become a prevalent choice for infrastructure-independent communication solutions. This is attributable to its inherent characteristics of long-distance communication, low power consumption, and robust signal penetration capabilities.

2.1

LoRa technology offers significant advantages over traditional communication methods like Device-to-Device (D2D), Unmanned Aerial Vehicles (UAVs), and Mobile Ad-hoc Networks (MANETs) in disaster communication scenarios. Its notable benefits include long-range communication, low power consumption, and strong interference immunity. Numerous LoRa-based communication solutions have been proposed, with LoRAgent standing out as a system architecture that integrates geographical awareness with delay-tolerant mechanisms.

The LoRAgent system comprises two types of nodes. One type consists of fixedly deployed relay nodes responsible for caching and forwarding messages. The other type involves mobile nodes that "carry" information in areas without network coverage until they encounter a suitable forwarding target, enabling non-real-time message delivery. To enhance forwarding efficiency, LoRAgent incorporates a quadrant-based directional forwarding strategy and a historical record filtering mechanism. This reduces redundant relays and improves transmission efficacy.

In contrast to LoRAgent's delay-tolerant architecture, Meshtastic offers a LoRa-based solution focused on real-time communication. It is built upon a self-organizing mesh network architecture, achieving multi-hop message forwarding through continuous broadcasting and relaying. By employing an event-driven transmission mode and a lightweight message structure, Meshtastic maintains both low power consumption and minimal latency. This makes it particularly well-suited for deployment on handheld or small-scale devices. In practical applications, developers have successfully integrated Meshtastic into ATAK through a plugin.

2.2 ATAK

The integration path of Meshtastic as a plugin within the Android Team Awareness Kit (ATAK) demonstrates the feasibility of embedding infrastructure-independent communication modules into a robust situational awareness platform. Among various platforms, ATAK stands out as an ideal integration target due to its highly modular architecture, real-world validated stability, and extensive support for plugins. Originally developed by the United States Department of Defense to fulfill military tactical requirements, the platform has subsequently garnered increasing popularity within civilian sectors. Its widespread deployment across military units, special police forces, search and rescue teams, and emergency management agencies in the United States exemplifies its utility. Similarly, in Germany, users within police emergency response and disaster control domains have highly commended the application's comprehensive functionalities.

To accommodate the complex and dynamic demands of tactical communication, ATAK's architectural design incorporates a highly modular approach, supporting the rapid integration of multi-source information and communication interfaces through its plugin mechanism. The inherent adaptability of the platform's communication layer allows it to flexibly accommodate heterogeneous physical networks, including LTE, Wi-Fi, MANETs, and satellite links. The establishment of communication paths can dynamically switch among UDP broadcast, TCP unicast, and XMPP-based centralized message distribution. Furthermore, the stability and latency characteristics of network transmission are determined by the plugin at runtime, based on the current link status. In extreme conditions lacking a central router or trusted node, ATAK's built-in GeoChat client module utilizes a UDP multicast mechanism to establish a logical broadcast domain, facilitating local information synchronization through peer-to-peer methods.

All data is ultimately encapsulated within the Cursor-on-Target (CoT) protocol, a lightweight XML-based structure developed by MITRE. This protocol features a clear event model and semantic hierarchy. Each CoT message is composed of an `<event>` element, which includes fields such as a unique identifier (`uid`), event type (`type`), time information (`start`, `stale`, `time`), generation method (`how`), geographical coordinates (`<point>`), and extended details (`<detail>`). The system utilizes the `<detail>` node to flexibly carry extended data, such as `__chat` for message bodies, `chatgrp` for group identifiers, `link` for associating entity relationships, and `__serverdestination` for explicitly specifying the message's target address and transmission protocol (e.g., `IP:PORT:PROTOCOL`). This protocol format is lightweight and semantically clear, supporting various information types including event types, geographical locations, and message content, thereby forming the foundation for inter-plugin collaboration and system compatibility.

2.2.1 *ATAK's Plugin Architecture for Customization.*

The plugin mechanism of ATAK is one of its most distinctive architectural features. Plugins exist as independent APK (Android Package Kit) files and do not rely on a whitelist from the main application. Instead, the main application automatically identifies, loads, and registers them at runtime. Plugins extend modular functionality based on predefined TAK APIs, gaining direct access to core functional modules such as map rendering, message distribution, databases, location information, and contact lists.

Typical class structures within this architecture include:

- **PluginLifecycle**: This component functions as the plugin's life cycle manager, analogous to Android's `Application` class, and is responsible for initialization and configuration hooks.
- **DropDownMapComponent**: This acts as the main controller, registering user interface logic, event listeners, and task scheduling components. It is functionally similar to an Android `Activity`.
- **DropDownReceiver**: This is primarily responsible for UI display, typically bound to a slide-out panel, and supports loading custom XML layouts.

User interface interactions are rendered via `DropDownReceiver.showDropDown()`, supporting responses to button clicks, map interactions, and background listening tasks.

Developing ATAK plugins requires Android Studio, with a minimum supported Android version of 5.0 (`API 21`) or higher. It is recommended to use Java 11, Gradle Plugin version `4.2.2`, and main project Gradle version `6.9.1` to ensure compatibility with the existing `ATAK-CIV` SDK. Plugins must possess a signing key for system validation, and their deployment path should be located within the `atak-civ/plugins` directory, maintaining consistent directory hierarchy with components such as `main.jar` and `atak.apk` within the SDK to ensure automatic loading.

3 METHODS

To achieve the goal of enabling infrastructure-less communication within ATAK in post-disaster scenarios, we propose a modular implementation and evaluation strategy, combining formal software integration, signal processing design, and hardware testing. The overall workflow consists of four main phases:

Phase 0: ATAK Chat Interface Integration

As a preliminary step, the project will integrate the ATAK GeoChat interface with a custom plugin. The plugin will observe, capture, and display incoming and outgoing chat messages. This is essential to verify that all required information (e.g., message text, sender, receiver ,timestamp) can be correctly extracted from CoT events. This validation step ensures that the plugin can operate independently of UI behavior and forms a solid foundation for later integration with a physical-layer communication backend.

Phase 1: WASM Runtime Integration

To facilitate the internal execution of physical-layer logic within ATAK, a static FutureSDR flowgraph will be compiled into WebAssembly (WASM) and subsequently embedded directly into the plugin. The interaction between the plugin and this flowgraph will be managed through memory-sharing mechanisms and exported function calls. This particular design allows for the complete encoding and decoding of Cursor-on-Target (CoT) messages into baseband In-phase/Quadrature (IQ) data entirely within the Android runtime environment. This initial phase primarily focuses on enabling signal processing capabilities without external dependencies, serving as a robust foundation for loopback testing, debugging, and fallback operational modes.

Phase 2: RF + CoT Integration

In scenarios requiring real-world over-the-air communication, the system will interface with external software-defined radio (SDR) hardware. Since WASM cannot directly access low-level device drivers or I/O APIs, the flowgraph will be executed as a native binary outside the plugin environment. ATAK will communicate with this flowgraph over UDP, TCP, or the local filesystem. Outgoing CoT messages will be serialized, forwarded to the external runtime, and transmitted via LoRa. Conversely, decoded messages received over the air will be injected back into ATAK as valid CoT events. This phase bridges ATAK with the physical RF layer in a modular and platform-agnostic way.

Phase 3: Dummy MAC (Multi-device)

To support multiple ATAK devices sharing the same LoRa channel, a simplified static MAC layer will be implemented. This protocol will embed sender and receiver UIDs into either the remarks field or a custom extension within the CoT detail structure. Basic filtering logic on the receiving side will drop irrelevant traffic. Although this MAC scheme is non-adaptive and stateless, it provides lightweight addressability and serves as a baseline for future protocol upgrades.

Phase 4: Evaluation + Fallback Testing

The final system will be evaluated under infrastructure-denied conditions. Testing will include:

- End-to-End Message Exchange Between ATA on Different Devices
- Compliance of injected CoT events with native ATAK GeoChat behavior
- Plugin behavior in fallback mode (using WASM flowgraph in loopback)

Fallback testing will simulate the absence of SDR hardware by replacing the RF path with in-app WASM flowgraphs, ensuring the plugin remains functional under limited-resource conditions.

4 LIMITATIONS, AND RISK MANAGEMENT

This project explicitly does not address issues related to message encryption and authentication, secure communication protocols, long-range transmission optimization, or power consumption control. The primary research focus is the development of a minimal working prototype that demonstrates end-to-end LoRa communication through the integration of FutureSDR and ATAK.

While the physical-layer signal processing flowgraph is provided as a static component at the project's outset, several technical risks related to integration persist.

One primary concern involves the compatibility of the WebAssembly (WASM) runtime within the Android environment. This ecosystem is still maturing, and potential issues could lead to abnormal communication or degraded performance between the plugin and the embedded flowgraph. Another significant risk arises during the process of reconstructing Cursor-on-Target (CoT) messages after the plugin receives demodulated data. This reconstruction relies heavily on XML parsing and structural mapping. Any inconsistencies in format or failures during parsing could result in message loss or rejection by the GeoChat module, hindering effective communication.

To address such extreme contingencies, the project defines two fallback solutions:

WASM Module Failure Fallback

Should WASM runtime compatibility issues arise within the Android environment during Phase 1manifesting as invocation failures or memory access conflicts, for instancethe system will pivot to an alternative approach. It will then interact with a locally running non-WASM software flowgraph via inter-process communication interfaces, such as TCP or UDP. This setup will continue to simulate the modulation and demodulation processes of CoT messages. It's important to note this alternative won't activate an actual radio frequency link; its sole purpose is to validate the correct integration of the plugin with the underlying physical layer interface logic.

RF Module Failure Fallback

Should SDR hardware connectivity fail during Phase 2, or if the RF link proves unstable under specific test conditions, the system will revert to the loopback simulation path established in Phase 1. This fallback involves the plugin utilizing the embedded WASM flowgraph to simulate physical channel behavior. This ensures continued validation of core functionalities, including message format conversion, Cursor-on-Target (CoT) structure generation, and GeoChat display, even without a live RF connection.