Open-Source Governance Architecture of Full-Stack OSI Layers and Tactical Communication Protocol Link32 with Skynet Reference Implementation for Drone Swarm Coordination in Cluster Hybrid Topologies over Local LTE 4G/5G Networks

Namdak Tonpa

June 11, 2025

Abstract

Coordinating large-scale drone swarms in battlefield or demonstration scenarios is a complex challenge, particularly under constraints of cost, scalability, and communication reliability. This article addresses the problem of designing an affordable, LTE 4G-based swarm coordination framework for unmanned aerial vehicle (UAV) or micro air vehicles (MAV) using cluster hybrid topologies, with low-resolution video streams from cluster lead drones for command-and-control (C2C) monitoring. We propose a hybrid architecture combining terrestrial and UAV-based base stations, leveraging low-bandwidth payloads (100 bytes at 10 Hz) and minimal video streaming (200 kbps per cluster lead). Key algorithms for path planning, collision avoidance, task allocation, formation control, and communication optimization are reviewed, drawing from bio-inspired and AI-driven methods. The solution supports approximately 5,000 drones over a 30 km radius, balancing cost, scalability, and performance.

Link32 is a tactical communication protocol designed for low-latency, secure, and scalable data exchange in contested environments, enabling swarm drone coordination, real-time position location information (PLI), command and control (C2), and tactical chat over UDP-based multicast networks. Skynet, its reference implementation, is a lightweight C99 framework inspired by LTE's QoS Class Identifier (QCI) framework, tailored for military applications and resource-constrained devices such as drones and embedded systems. With minimal dependencies and a focus on portability, Skynet supports dynamic TDMA slot management, AES-256-GCM encryption, and lock-free concurrency, ensuring robust performance in mission-critical scenarios.

Contents

1 Open Source Governance

The Skynet/Link32 framework adopts an open-source governance model to balance the dual objectives of fostering collaborative development and ensuring military-grade security. This section outlines the governance principles, delineates the separation of public and private protocol components, and addresses the motivations and benefits of open-sourcing parts of the system, particularly

for simulation, research, and interoperability with commercial systems. By adhering to a structured governance model, Skynet mitigates risks associated with open-source development while maximizing its utility for academic, industrial, and military stakeholders.

1.1 Governance Principles

The open-source governance of Skynet/Link32 is guided by the following principles, designed to withstand scrutiny from security, scalability, and interoperability perspectives:

- **Modular Separation**: The protocol is divided into public and private components. Public fields, such as common message headers and payload structures (e.g., `SkyNetMessage` version, type, and QoS fields), are openly documented and compatible with commercial-off-the-shelf (COTS) appliances, ensuring interoperability. Private fields, containing sensitive military data (e.g., tactical C2 payloads, encrypted sensor data), are abstracted and inaccessible to non-authorized entities, safeguarding mission-critical information.
- **Transparency and Collaboration**: Public components, including the core Link32 protocol specification, Skynet server implementation, and simulation tools, are released under an open-source license (e.g., MIT or Apache 2.0) to encourage community contributions, academic research, and third-party validation. This transparency fosters trust and accelerates innovation in UAV swarm coordination algorithms.
- **Security by Design**: Private components, such as cryptographic key management, proprietary algorithms, and military-specific payloads, are maintained in closed-source modules or referenced via private tables. These are protected by AES-256-GCM encryption and ECDH key exchange (SECP384R1), ensuring compliance with military security standards (e.g., MIL-STD-6016).
- **Community Oversight**: A governance board, comprising xAI representatives, academic partners, and military stakeholders, oversees contributions, reviews code for security vulnerabilities, and ensures alignment with tactical requirements. Contributions to public components undergo rigorous peer review to prevent backdoors or exploits.
- **Version Control and Auditability**: All open-source components are versioned and hosted on a public repository (e.g., GitHub: `https://github.com/BitEdits/skynet`. Comprehensive audit logs track changes, ensuring traceability and accountability.

1.2 Public and Private Payload Separation

The Link32 protocol employs a clear delineation between public and private payloads within the `SkyNetMessage` structure to address security concerns while

enabling interoperability. Public fields, are standardized and documented in the open-source specification, allowing commercial appliances (e.g., srsRAN-based LTE systems) to process message headers without accessing sensitive data. The `payload` field, which may contain military-specific data (e.g., waypoints, sensor readings), is encrypted using AES-256-GCM with a 16-byte initialization vector (`iv`) and a 16-byte authentication tag, rendering it opaque to unauthorized systems.

Private payloads are further abstracted through reference tables, which define proprietary data structures (e.g., tactical C2 formats) accessible only to authorized nodes with pre-provisioned keys. For example message for a Bomber UAV includes a public header (NPG 100, QoS 2) and a private payload (encrypted coordinates), ensuring that commercial LTE base stations can route the message without decrypting its contents. This separation mitigates risks of reverse-engineering while enabling integration with COTS hardware.

1.3 Motivations for Open-Source Approach

The decision to open-source select components of Skynet/Link32 is driven by strategic objectives that address potential criticisms regarding security, scalability, and community engagement:

- **Interoperability with Commercial Systems**: By exposing public fields and protocol specifications, Skynet ensures compatibility with commercial LTE 4G infrastructure, reducing deployment costs. This enables seamless integration with open-source tools like srsRAN and Open5GS, broadening adoption in both military and civilian contexts.

- **Research and Simulation**: Open-sourcing the core protocol and simulation tools (e.g., Skynet server, client utilities) provides researchers with a platform to develop and test UAV swarm algorithms, such as those for path planning (A*, PSO) and formation control (consensus-based). This fosters academic collaboration and accelerates innovation without compromising military-specific components.

- **Community Validation**: Public scrutiny of open-source code enhances security by allowing independent researchers to identify and fix vulnerabilities, reducing the risk of zero-day exploits. For example, the QoS handling is openly auditable, ensuring robust performance under diverse network conditions.

- **Cost Efficiency**: Leveraging open-source software and COTS hardware minimizes development and maintenance costs, addressing criticisms about affordability in large-scale deployments (e.g., 5,000 UAVs over 30 km).

1.4 Addressing Criticisms

Critics may argue that open-sourcing any part of a military communication protocol risks security breaches or reduces control over proprietary technology. These concerns are addressed as follows:

- **Security Risks**: The modular separation of public and private components ensures that sensitive data remains encrypted and inaccessible. Only authorized nodes with ECDH-derived AES keys can decrypt private payloads, and manual key provisioning is being enhanced with automated distribution to further secure access. The open-source community's ability to audit public code strengthens, rather than weakens, security by identifying vulnerabilities early.
- **Loss of Proprietary Control**: Private payloads and algorithms are maintained in closed-source modules or referenced via proprietary tables, ensuring that military-specific intellectual property remains protected. The governance board retains authority over which components are open-sourced, preventing unauthorized disclosures.
- **Interoperability Overhead**: Public fields are designed to align with standards like MQTT (ISO/IEC 20922:2016) and LTE QCI (3GPP TS 23.203), ensuring compatibility without compromising performance. The NPG-to-MQTT mapping demonstrates this alignment, with QoS levels optimized for tactical needs.

2 Introduction to UAV/MAV classes

Each UAV class has distinct roles, payloads, and operational requirements, analogous to military aircraft. These roles drive their communication needs in terms of data type, frequency, priority, and reliability.

2.1 Scout (Reconnaissance UAV)

Comparable to reconnaissance aircraft like the RQ-4 Global Hawk or U-2 Dragon Lady. Primary function is surveillance, intelligence gathering, and real-time situational awareness. Scouts collect data (e.g., imagery, sensor readings) and transmit Position Location Information (PLI), sensor feeds, and alerts.

Communication Needs. Data Types: PLI (position, velocity), high-frequency sensor data (e.g., EO/IR imagery), and network alerts. Frequency: High-frequency updates (10 Hz for PLI, 1 Hz for sensor data). Priority: Medium to high (timely intelligence is critical but not as urgent as C2). Reliability: Reliable delivery for PLI and alerts, best-effort for non-critical sensor data. Bandwidth: Moderate (100-byte PLI at 10 Hz 9.6 kbps; 200 kbps for low-resolution video).

Requirements. High data volume from sensors, need for robust encryption, and operation in contested environments with potential jamming.

2.2 Bomber (Heavy Payload UAV)

Similar to strategic bombers like the B-2 Spirit or B-1 Lancer. Delivers heavy payloads (e.g., precision-guided munitions) to strategic targets, requiring precise navigation and command-and-control (C2) coordination.

Communication Needs. Data Types: C2 (waypoints, target assignments), PLI, and status updates (e.g., payload status). Frequency: Moderate (C2 updates at 1 Hz, PLI at 5–10 Hz). Priority: Very high (mission-critical C2 commands require low latency). Reliability: Exactly-once delivery for C2, reliable for PLI. Bandwidth: Low to moderate (100-byte C2/PLI at 5–10 Hz 4.8–9.6 kbps).

Requirements. Low-latency C2 for real-time targeting, secure communication to avoid interception, and coordination with other UAVs for swarm operations.

2.3 Interceptor (Anti-UAV/Countermeasure UAV)

Comparable to air superiority fighters like the F-22 Raptor. Engages and neutralizes enemy UAVs or missiles, requiring rapid response and dynamic coordination.

Communication Needs. Data Types: C2 (target tracking, engagement orders), PLI, and alerts (e.g., threat detection). Frequency: High (PLI and alerts at 10 Hz, C2 at 2 Hz). Priority: Very high (real-time engagement demands sub-50ms latency). Reliability: Exactly-once for C2 and alerts, reliable for PLI. Bandwidth: Moderate (100-byte messages at 10 Hz 9.6 kbps).

Requirements. Ultra-low latency for real-time threat response, robust anti-jamming, and coordination with other interceptors for swarm-based defense.

2.4 Fighter (Multi-Role UAV)

Similar to multi-role fighters like the F-16 Falcon or F-35 Lightning II. Performs a mix of reconnaissance, strike, and air superiority tasks, requiring flexibility in communication for diverse missions.

Communication Needs. Data Types: C2, PLI, sensor data, tactical chat, and formation control. Frequency: Variable (PLI at 10 Hz, C2 at 1–2 Hz, sensor data at 1 Hz). Priority: High for C2 and formation, medium for sensor data and chat. Reliability: Exactly-once for C2, reliable for PLI and formation, best-effort for chat. Bandwidth: Moderate to high (9.6 kbps for PLI, 200 kbps for video, 1 kbps for chat).

Requirements. Balancing diverse traffic types, maintaining formation in dynamic environments, and ensuring scalability in mixed swarms.

2.5 Attack (Close Air Support UAV)

Comparable to attack aircraft like the A-10 Thunderbolt II or AH-64 Apache. Provides close air support, engaging ground targets with precision munitions, often in coordination with ground forces.

Communication Needs. Data Types: C2 (target coordinates, engagement orders), PLI, tactical chat, and status updates. Frequency: Moderate to high (C2 at 1–2 Hz, PLI at 10 Hz, chat at 0.1 Hz). Priority: High for C2, medium for PLI and chat. Reliability: Exactly-once for C2, reliable for PLI, best-effort for chat. Bandwidth: Moderate (9.6 kbps for PLI, 1 kbps for chat, 5 kbps for C2).

Requirements. Close coordination with ground units, low-latency C2 for real-time targeting, and resilience to battlefield interference.

3 Protocol Requirements

Link32 is a tactical real-time communication protocol inspired by standards such as VMF, TSM, SRW, TDL [11] (message formats) along with MQTT [14], LTE (HSS, PDCP, MME, QCI), Link16 [12] (telecommunication protocols), tailored for military applications requiring robust, low-latency, and secure data exchange in contested environments. It facilitates swarm drone coordination, real-time position location information (PLI), command and control (C2), and tactical chat using UDP-based multicast networks.

Skynet is the reference server implementation of Link32, developed in C99 with minimal dependencies to ensure portability and performance on resource-constrained devices. Its convergence layer draws inspiration from LTE's QoS Class Identifier QCI framework [13], enabling granular QoS control and dynamic resource allocation for mission-critical communications.

The proliferation of micro air vehicles (MAVs) has enabled applications such as battlefield surveillance, search and rescue, and large-scale demonstrations. Coordinating thousands of drones requires robust architectures and algorithms to manage communication, navigation, and task allocation in dynamic environments. Traditional centralized systems face scalability issues, while high-cost 5G solutions are impractical for affordable deployments. This article proposes an affordable, LTE 4G-based swarm coordination framework using cluster hybrid topologies, where each drone transmits a 100-byte payload (position, velocity, tactical data) at 10 Hz, and 5–10 cluster lead drones stream low-resolution video (200 kbps) for C2C monitoring. We focus on a 30 km radius operational area, addressing the challenges of bandwidth, latency, and cost through hybrid architectures and optimized algorithms.

The Link32 protocol and Skynet implementation must address the following requirements to support these UAV classes:

- Low Latency: Critical for C2 and alerts (sub-50ms for Interceptor, Bomber, Fighter, Attack).
- Reliability: Exactly-once delivery for C2 (QoS 2), reliable delivery for PLI (QoS 1), and best-effort for chat and non-critical sensor data (QoS 0).
- Scalability: Support up to 5,000 UAVs, with cluster-based organization (50–100 drones per cluster) to manage bandwidth (50 Mbps uplink).
- Security: AES-256-GCM, ECDH SECP384R1 in contested environments.
- Bandwidth Efficiency: Optimize multicast topics to minimize overhead, using compact 100-byte payloads at 10 Hz and 200 kbps video streams for cluster leads.
- Dynamic Adaptation: Support dynamic TDMA slot allocation and topic subscriptions based on UAV roles and mission requirements.
- Interoperability: Map to MQTT QoS levels for integration with existing tactical communication systems.

3.1 Principles

Link32 adheres to the following design principles:

- **Threat Model**: Confidentiality and integrity, no non-repudiation.
- **Latency**: Microsecond-precision timings .
- **Implementation**: Written in C99 for portability and performance.
- **Key Provisioning**: Key distribution for controlled setup.
- **Mandatory Encryption**: All messages encrypted with AES-256-GCM.
- **Node Identification**: Node names hashed to 32-bit using FNV-1a.
- **Lock-Free Design**: Uses atomic compare-and-swap for concurrency.
- **Topic Architecture**: Topics map to IP multicast groups.
- **Queue Management**: Global network queue, per-topic subscribes.
- **Key Storage**: Separate key stores per executable.

3.2 Properties

Link32 and Skynet are designed with the following properties:

- **Message Size**: 32-byte header (48 bytes payload with GCM tag).
- **Security**: ECDH key exchange SECP384R1, Mandatory AES-256-GCM.
- **Non-blocking**: Using monotonic clocks and non-blocking I/O.
- **Footprint**: ∼64KB L1 cache usage, ∼2000 lines of code (LOC).
- **Swarm Scalability**: Supports large-scale drone swarms.
- **Dependencies**: Single dependency on OpenSSL for cryptography.

3.3 Problem Statement

The problem is to design a cost-effective LTE 4G-based communication and coordination system for a swarm of approximately 5,000 MAVs operating over a 30 km radius in a battlefield or demonstration setting. Each drone sends a 100-byte payload (position, velocity, tactical data) at 10 Hz, requiring approximately 9.6 kbps per drone, while 5–10 cluster lead drones stream low-resolution video at 200 kbps each for C2C monitoring. The total uplink bandwidth is approximately 50 Mbps. The system must:

- Ensure reliable communication within LTE 4G constraints (50 Mbps uplink, 20–100 ms latency).

- Achieve 30 km coverage using minimal, low-cost equipment.
- Support scalable coordination for path planning, collision avoidance, task allocation, and formation control.
- Maintain affordability through open-source or commercial off-the-shelf (COTS) components.

Challenges include limited bandwidth, latency constraints, signal interference in battlefield environments, and the need for decentralized control to scale to thousands of drones.

3.4 Cluster Hybrid Topology

To address the problem, we propose a cluster hybrid topology combining a terrestrial LTE 4G base station with one or more UAV-based aerial base stations (ABSs). The architecture is designed for affordability and scalability, leveraging open-source solutions and COTS hardware.

3.4.1 Terrestrial Base Station

A single terrestrial base station, implemented using srsRAN with a LimeSDR Mini (cost: ∼$1,500), provides core coverage of 10–15 km. A high-gain omnidirectional antenna (e.g., Laird FG24008, ∼$150) and a 20 W RF amplifier (∼$500) extend the range toward 20 km. The base station connects to an open-source Evolved Packet Core (EPC) like Open5GS on a low-cost server (∼$500), handling up to 2,000 simultaneous connections.

3.4.2 UAV-Based Aerial Base Station

One custom-built UAV (cost: ∼$2,000) equipped with a LimeSDR Mini (∼$300) and a lightweight antenna (e.g., Taoglas GW.26, ∼$50) serves as an aerial base station at 100–120 m altitude, extending coverage to 20–30 km via line-of-sight (LoS) propagation. The UAV rotates with spares to ensure continuous operation, supported by a portable generator (∼$1,100) and battery swap system (∼$300).

3.4.3 Cluster-Based Swarm Organization

The swarm is organized into clusters of 50–100 drones, each led by a cluster head responsible for intra-cluster coordination and LTE communication with the base station. Cluster heads transmit 100-byte payloads (9.6 kbps) and, for 5–10 designated leaders, low-resolution video streams (200 kbps). Drone-to-drone (D2D) communication via LTE sidelink reduces network load, enabling scalability to 5,000 drones within the 50 Mbps uplink capacity.

3.4.4 Backhaul and Ground Control

A point-to-point microwave link (e.g., Ubiquiti airFiber 24, ∼$1,500) or satellite terminal (e.g., Starlink, ∼$600) provides backhaul to a command center. A rugged laptop with open-source software like QGroundControl (∼$500) manages UAV flight and network configuration. Total cost is approximately $8,400, making the system affordable for battlefield or demonstration use.

4 Swarm Coordination

The coordination of 5,000 MAVs requires algorithms for path planning, collision avoidance, task allocation, formation control, and communication optimization. These algorithms are optimized for low computational overhead and LTE 4G constraints, drawing from bio-inspired and AI-driven approaches. Below, we describe each algorithm, highlighting its essence and relevance to the proposed architecture.

4.1 Path Planning and Navigation

Effective path planning ensures drones navigate efficiently in complex environments, balancing computational simplicity with adaptability to dynamic conditions.

4.1.1 A* Algorithm

The A* algorithm is a cornerstone of path planning, leveraging a graph-based approach to find the shortest path from a drone's current position to its target. By combining heuristic estimates with actual costs, A* ensures optimal trajectories while integrating with genetic algorithms for real-time optimization in cluttered environments, such as battlefields with obstacles. Its computational efficiency suits micro drones with limited processing power [19, 20].

4.1.2 Particle Swarm Optimization (PSO)

Inspired by the flocking behavior of birds, PSO optimizes drone trajectories by treating each drone as a particle exploring a solution space. Particles adjust their paths based on local (individual best) and global (swarm best) solutions, making PSO ideal for target tracking and search missions in dynamic settings. Its distributed nature aligns with cluster-based topologies [21, 22].

4.1.3 Ant Colony Optimization (ACO)

ACO mimics the pheromone trails of ants to discover optimal paths in Flying Ad Hoc Networks (FANETs). Drones share virtual pheromone data to guide routing and task allocation, enabling efficient navigation in large-scale swarms. Its robustness to dynamic changes suits battlefield scenarios with intermittent connectivity [23, 24].

4.1.4 Differential Evolution

Differential Evolution dynamically adjusts swarm coordination parameters, such as trajectory weights, by evolving a population of candidate solutions. This adaptability ensures drones respond to environmental changes, optimizing paths in real-time for tasks like formation control. Its lightweight computation is suitable for micro drones [25, 26].

4.1.5 Kalman Filter

The Kalman Filter estimates drone positions and velocities by fusing noisy sensor data, critical for navigation in GPS-denied environments like urban canyons or forests. Its recursive nature minimizes computational load, enabling robust localization in large swarms [27, 28].

4.1.6 Spatial-Temporal Joint Optimization

This approach synchronizes trajectory shapes and timing to optimize navigation in cluttered environments. By jointly considering spatial paths and temporal constraints, it ensures efficient coordination for tasks like surveillance, minimizing energy use and collisions [29, 30].

4.2 Collision Avoidance

Collision avoidance is critical for dense swarms, ensuring safe navigation without excessive computational or communication demands.

4.2.1 Artificial Potential Field (APF)

APF treats drones as particles in a potential field, repelled by obstacles and neighbors while attracted to targets. This intuitive method generates smooth trajectories with low computational cost, though it risks local minima where drones become static. It suits resource-constrained MAVs [31, 63].

4.2.2 Reactive Collision Avoidance

Using onboard sensors like time-of-flight, reactive collision avoidance enables drones to detect and avoid obstacles in real-time. Its simplicity and low power requirements make it ideal for micro drones in dynamic battlefield environments [33, 34].

4.2.3 Flocking Algorithms

Based on Reynolds' rules (cohesion, separation, alignment), flocking algorithms ensure drones maintain safe distances while moving cohesively. This bio-inspired approach scales well for large swarms, supporting cluster-based coordination [35, 36].

4.2.4 Vision-Based Avoidance

Vision-based avoidance uses cameras to detect neighbors and obstacles, offering precise spatial awareness. Despite limitations from field-of-view and lighting, it enhances safety in dense swarms when paired with lightweight sensors [37, 38].

4.3 Task Allocation and Coordination

Task allocation ensures drones efficiently distribute mission responsibilities, adapting to dynamic conditions with minimal communication.

4.3.1 Deep Reinforcement Learning (DRL)

DRL, using algorithms like Proximal Policy Optimization (PPO) and Deep Q-Network (DQN), enables drones to learn optimal task allocation and path planning through environmental interaction. Its adaptability excels in dynamic settings like battlefield patrolling [39, 40].

4.3.2 Stigmergy

Inspired by social insects, stigmergy allows drones to leave virtual "pheromones" (data markers) to guide others toward tasks, reducing communication overhead. This decentralized approach suits LTE-constrained swarms [41, 42].

4.3.3 Consensus Algorithms

Consensus algorithms ensure drones agree on shared goals or states, even under poor LTE connectivity. They are critical for merging swarms or maintaining coordination in disrupted environments [43, 44].

4.3.4 Virtual Navigator Model

This model dynamically adjusts patrol paths based on environmental changes, enhancing flexibility for tasks like surveillance. It integrates with cluster-based topologies for scalable coordination [45, 46].

4.4 Formation Control

Formation control maintains swarm geometry, balancing precision with robustness to disruptions.

4.4.1 Leader-Follower

In leader-follower formations, a cluster head guides followers, maintaining relative positions. Its simplicity suits small clusters but is vulnerable to leader failure [47, 48].

4.4.2 Virtual Structure

The virtual structure approach treats the swarm as a rigid geometric shape, assigning each drone a fixed position. It ensures precise formations but lacks flexibility in dynamic environments [49, 50].

4.4.3 Behavior-Based

Behavior-based methods use simple rules (e.g., avoid collisions, stay near neighbors) to achieve emergent formations. Their scalability makes them ideal for large swarms [51, 52].

4.4.4 Consensus-Based

Consensus-based methods share state information to maintain formations, robust to communication disruptions. They suit LTE 4G environments with variable connectivity [53, 54].

4.4.5 Graph-Based Models

Graph-based models represent drones as vertices and communication links as edges, enabling flexible formation adjustments. They support dynamic reconfiguration in cluster-based swarms [55, 56].

4.5 Communication Optimization

Communication optimization ensures efficient data exchange within LTE 4G constraints, minimizing latency and energy use.

4.5.1 Reactive-Greedy-Reactive (RGR) Protocol

RGR combines Ad Hoc On-Demand Distance Vector (AODV) with Greedy Geographic Forwarding (GGF) to reduce latency in FANETs. It adapts to dynamic topologies, ensuring reliable packet delivery [57, 58].

4.5.2 Bee Colony Optimization (BCO)

BCO mimics bee foraging to optimize routing in FANETs, balancing exploration and exploitation for efficient communication in large swarms [59, 60].

4.5.3 EPOS

EPOS is a decentralized multi-agent learning algorithm that optimizes energy consumption and task allocation for spatio-temporal sensing, ideal for LTE-constrained swarms [61, 62].

4.5.4 Graph Attention-Based Decentralized Actor-Critic

This method uses graph neural networks to enable dual-objective control (task completion, energy efficiency), enhancing communication efficiency in cluster-based topologies [63, 64].

5 Link32 Protocol

5.1 S-Message Format

The Link32 message structure, `SkyNetMessage`, is compact for large-scale swarm communication:

```
typedef struct {
  uint8_t version : 4;    // Protocol version (current: 1)
  uint8_t type : 4;       // Message type (0-6)
  uint8_t qos : 4;        // Quality of Service (0-3)
  uint8_t hop_count : 4;  // Hop count for routing (0-15)
  uint32_t npg_id;        // Topic identifier (1-103)
  uint32_t node_id;       // Sender node ID (FNV-1a hash)
  uint32_t seq_no;        // Sequence number for deduplication
  uint8_t iv[16];         // AES-256-GCM initialization vector
  uint16_t payload_len;   // Payload length (0-32767)
  uint8_t payload[MAX_BUFFER];     // GCM-TAG
} SkyNetMessage;
```

- **Header Size**: 32 bytes.
- **Minimum Payload Size**: 48 bytes minimum (32-byte + GCM tag).
- **Maximum Payload Size**: Up to 32720 bytes.

5.2 Message Types

The protocol defines seven message types, as shown in Table 1.

Table 1: Link32 Message Types

Type	Name	Description
0	Key Exchange	Exchanges ECC public keys for ECDH session setup.
1	Slot Request	Requests a TDMA slot from the server.
2	Chat	Sends tactical chat messages.
3	Ack	Acknowledges slot assignments or control messages.
4	Waypoint	Specifies navigation waypoints for C2.
5	Status	Reports position, velocity, or sensor data (e.g., PLI).
6	Formation	Coordinates swarm formations.

5.3 Multicast Topics

Link32 uses multicast topics mapped to IP multicast groups 239.255.0.*, where * is NPG identifier as in Table 2.

Table 2: Multicast Topics

NPG	Name	Purpose
1	npg_control	Key exchange, Slot requests, Ack.
6	npg_pli	All Position/Velocity updates.
7	npg_surveillance	Scout, Fighter.
29	npg_chat	Tactical chat and acks.
100	npg_c2	C2, Bomber, Interceptor, Fighter, Attack.
101	npg_alerts	Scout, Interceptor, Fighter.
102	npg_logistics	Bomber, Attack.
103	npg_coord	Fighter, Attack.

5.4 Slot Management

Link32 employs a Time Division Multiple Access (TDMA)-like slot manager to minimize collisions:

- **Slot Array**: Fixed-size (256) array `slots`.
- **Dynamic Topics**: Each slot creates a temporary multicast group.
- **Allocation**: First-come, first-serve with no timeouts.
- **Timing**: Slots cycle every `TIME_SLOT_INTERVAL_US=1000`μ`s`.

Clients send `SKYNET_MSG_SLOT_REQUEST` to NPG 1, and the server assigns slots via `SKYNET_MSG_ACK`.

5.5 Deduplication

A fixed-size circular buffer (`seq_cache`) prevents message loops:

- **Structure**: Stores `node_id`, `seq_no`.
- **Memory**: $\sim$16KB (1024 $\times$ 16 bytes).
- **Complexity**: O(1) lookup using FNV-1a hashing.
- **Threshold**: Discards duplicates within 1 second.

5.6 Security

Security mechanisms include:

- **Key Exchange**: ECDH over secp384r1 for 256-bit AES keys.
- **Encryption**: AES-256-GCM with 16-byte IV and 16-byte tag.
- **Key Storage**: Per Process in `ec_priv` and `ec_pub`.
- **Key Derivation**: HKDF-SHA256 for AES keys.
- **Self-messages**: Skips (drops) messages.

5.7 Subscriptions

Nodes subscribe to topics based on roles, as shown in Table 3:

Table 3: Role-Based Subscriptions

Role	NPGs	Purpose
Infantry	1, 29	Network control and tactical chat.
Drone	1, 6, 7, 100, 101	C2, PLI, surveillance, alerts.
Air	1, 6, 7, 100, 101, 103	C2, PLI, surveillance, alerts, coord.
Sea	1, 7, 29, 102, 103	C2, surveillance, chat, logistics, coord.
Ground	1, 7, 29, 102	C2, surveillance, chat, logistics.
Relay	1, 6, 101	C2, PLI, alerts for relaying.
Controller	1, 6, 100, 101	C2, PLI, alerts for command posts.

6 Skynet Implementation

6.1 Dependencies

- **OpenSSL**: For ECC, ECDH, and AES-256-GCM.
- **C99 Compiler**: GCC or equivalent.
- **POSIX Environment**: For threading, epoll, and timerfd.

6.2 Build

To build Skynet:

```
# git clone git@github.com:BitEdits/skynet
# cd skynet
# cc -o skynet_client skynet_client.c skynet_proto.c -lcrypto
# cc -o skynet skynet.c skynet_proto.c skynet_conv.c -lcrypto
```

6.3 Installation

The provisioning script `skynet.sh` generates ECC key pairs:

```
# ./skynet.sh
Generated keys for node npg_control (hash: 06c5bc52) in /secp/
Generated keys for node npg_pli (hash: c9aef284) in /secp/
Generated keys for node npg_surveillance (hash: 4d128cdc) in /secp/
Generated keys for node npg_chat (hash: 9c69a767) in /secp/
Generated keys for node npg_c2 (hash: 89f28794) in /secp/
Generated keys for node npg_alerts (hash: 9f456bca) in /secp/
Generated keys for node npg_logistics (hash: 542105cc) in /secp/
Generated keys for node npg_coord (hash: e46c0c22) in /secp/
Generated keys for node server (hash: 40ac3dd2) in /secp/
Generated keys for node client (hash: 8f929c1e) in /client/secp/
# cp /secp/*.ec_pub /client/secp/
```

6.4 Server Operation

The server binds to UDP port 6566, joins multicast groups, and processes messages using a global queue. Example output:

```
# skynet server
Node 40ac3dd2 bound to 0.0.0.0:6566.
Joined multicast group 239.255.0.1 (NPG 1: control).
Joined multicast group 239.255.0.6 (NPG 6: PLI).
Message received, from=8f929c1e, to=1, size=231.
Decryption successful, from=8f929c1e, to=1, size=215.
Saved public key for client 8f929c1e.
Assigned slot 0 to node 8f929c1e.
Message received, from=8f929c1e, to=6, size=40.
Decryption successful, from=8f929c1e, to=6, size=24.
Message sent from=8f929c1e, to=6, seq=3, multicast=239.255.1.0.
```

6.5 Client Operation

The client joins topic-specific multicast groups and sends key exchange, slot requests, and status messages. Example output:

```
# skynet_client client
Node 8f929c1e connecting to port 6566.
Joined multicast group 239.255.0.1 (NPG 1).
Joined multicast group 239.255.0.6 (NPG 6).
Sent key exchange message to server.
Sent slot request message to server.
Received slot assignment: slot=0.
Joined slot group 2399540.1.
Sent status message: multicast=239.255.1.0,
                     pos=[0.1, 0.1, 0.1],
                     vel=[0.0, 0.0, 0.0],
                     seq=2.
```

6.6 Usage and Limitations

Skynet includes five utilities:

Keys Provisioning

Generates ECC key pairs for Skynet Client telecommunication module on User Equipment (UE).

```
skynet_keygen <node> [--server|--client]
```

Message Encryption

Encrypts a test message to `file.sky`. For manual message encryption before manual publishing on topic channel.

```
skynet_encrypt <sender> <recipient> <file>
```

Message Decryption

Decrypts `<file.sky>`. For manual message decryption after recieving on subscribed topic channel.

```
skynet_decoder <sender> <recipient> <file.sky>
```

Skynet Server

Runs the server with FNV-1a hashed `<node>`. Skynet Server implements Link32 Broker along with Channel Protocol Handlers which coordinate drone swarms using Link32 telecommunication protocol.

```
skynet <node>
```

Skynet Client User Equipment

Runs the client with FNV-1a hashed `<node>`. Skynet Client implements FSM loop of drone telecommunication module which is using Link32 protocol (IP/UDP) while User Equipment implements FSM loop of drone telecommunication module which is using LTE 4G protocol (MAC/RRC/NR/LTE/CELL).

```
skynet_client <node>
```

COTS User Equipment SIM-card Programming

Flash SIM cards for srsRAN setups with `PCSC` and `CCID` supported valued in `<device>` placeholder.

```
skynet_sim <device> <sim.imsi>
```

- **Slot Scalability**: Fixed `SLOT_COUNT=256` limits nodes to 256.

- **No Retransmission**: Dropped messages are not retransmitted.
- **Key Management**: Manual public key copying required.
- **Deduplication**: `SEQ_CACHE_SIZE=1024` may cause collisions.

7 Convergence Architecture

The Skynet system's convergence layer employs a 3-level structural hierarchy comprising Quality of Service (QoS), Bearer, and Entity components. This design is driven by the need for granular QoS control, per-node resource management, reliable packet delivery, dynamic slot allocation, and scalability in a Time Division Multiple Access (TDMA)-based tactical network. Inspired by the Long-Term Evolution (LTE) QoS Class Identifier (QCI) framework, the hierarchy ensures military-grade performance, including latency below 50ms for command-and-control (C2) traffic and reliable delivery for critical communications, such as swarm drone coordination. Compared to alternative designs, such as flat or centralized structures, this approach excels in dynamic, resilient scenarios, providing robust and scalable QoS management.

7.1 Flat QoS Structure

A flat QoS structure assigns slots directly to Network Protocol Groups (NPGs) based on their QoS profiles, without intermediate bearer or entity layers. For example, an NPG like `SKYNET_NPG_C2` might be statically allocated three slots with QoS level 3. While simpler and requiring less memory, this approach lacks per-node isolation, making it unsuitable for dynamic networks with multiple nodes sharing the same NPG. It also complicates reliable delivery, as there is no mechanism for per-flow reordering or sequence tracking. Additionally, static slot assignments cannot adapt to node arrivals or departures, leading to inefficient resource utilization. The 3-level hierarchy overcomes these limitations by introducing bearers for flow isolation and entities for node-level coordination, enabling dynamic and scalable QoS management.

```
typedef struct {
    uint32_t npg_id;
    uint8_t qos;
    uint32_t slot_count;
    uint32_t slot_ids[MAX_QOS_SLOTS];
    uint8_t priority;
} QoSSlotAssignment;
```

7.2 Hierarchical QoS Structure

The 3-level hierarchy separates concerns into QoS, Bearer, and Entity layers, each addressing specific aspects of network management. The `SkyNetBearerQoS` structure defines QoS parameters such as priority (1–15, where 1 is highest),

delay budget (in milliseconds), reliability (best-effort or reliable), and minimum TDMA slots, allowing precise service differentiation. The `SkyNetBearer` represents a logical communication channel for a node-NPG pair, encapsulating QoS parameters, assigned slots, and state for reordering and reliability. The `SkyNetConvergenceEntity` aggregates bearers for a single node, managing slot requests and coordinating resource allocation. This modular design ensures scalability, flexibility, and robustness, outperforming flat structures by providing flow isolation and dynamic adaptation.

```
typedef struct {
    uint8_t priority;        // 1-15 (1 = highest)
    uint32_t delay_budget_ms;// Delay tolerance (ms)
    uint8_t reliability;     // 0 (best-effort), 1 (reliable)
    uint32_t min_slots;      // Minimum TDMA slots
} SkyNetBearerQoS;

typedef struct {
    uint32_t bearer_id;      // Unique bearer ID
    SkyNetBearerQoS qos;     // QoS parameters
    uint32_t node_id;        // Owning node
    uint32_t npg_id;         // Associated NPG
    uint32_t assigned_slots[SKYNET_MAX_SLOTS]; // Assigned slot IDs
    uint32_t slot_count;     // Number of assigned slots
    SkyNetMessage reorder_queue[SKYNET_REORDER_SIZE];
    uint32_t expected_seq_no;// Next expected sequence number
    uint32_t last_delivered; // Last delivered sequence number
    uint64_t last_reorder_time_us; // Last reorder check
} SkyNetBearer;

typedef struct {
    SkyNetBearer bearers[SKYNET_MAX_BEARERS]; // Active bearers
    uint32_t bearer_count;   // Number of active bearers
    atomic_uint slot_requests_pending; // Pending slot requests
} SkyNetConvergenceEntity;
```

7.3 Granular QoS Control

Tactical networks handle diverse traffic types, such as C2, position location information (PLI), and chat, each with distinct latency, reliability, and bandwidth requirements. A uniform QoS approach fails to meet these needs. The `SkyNetBearerQoS` structure enables granular control by defining specific parameters for each bearer. For instance, `SKYNET_NPG_CONTROL` (NPG 1, QoS 3) is assigned three slots with a low delay budget to ensure timely C2 delivery, while `SKYNET_NPG_CHAT` (NPG 103, QoS 0) receives one slot for best-effort traffic. This differentiation, inspired by LTE QCI, guarantees that high-priority traffic meets stringent military requirements, such as sub-50ms latency for C2, while optimizing resource allocation for lower-priority flows.

7.4 Per-Node and Per-Flow Resource Management

Each node in the network may support multiple concurrent flows (e.g., C2, PLI, control) with varying QoS needs. Without isolation, these flows compete for

resources, risking contention and degraded performance. The `SkyNetBearer` structure provides flow-level isolation by associating each bearer with a specific node-NPG pair, tracking its assigned slots and QoS parameters. The `SkyNetConvergenceEntity` groups all bearers for a node, enabling centralized resource management and preventing one flow from starving others. This per-node and per-flow approach ensures efficient slot allocation, supports multiple simultaneous communications, and scales to accommodate dynamic network topologies.

7.5 Reliability and Reordering

TDMA networks may deliver packets out of order due to slot scheduling or retransmissions, particularly for reliable traffic (e.g., `reliability=1`). The bearer includes a `reorder_queue` and tracks `expected_seq_no` and `last_delivered` to reorder packets and ensure reliable delivery. The entity coordinates reorder checks across bearers using `last_reorder_time_us`, minimizing overhead. This mechanism is critical for applications requiring guaranteed delivery, such as C2 or control messages, and enhances robustness compared to flat structures, which lack per-flow reordering capabilities.

7.6 Dynamic Slot Allocation

Tactical networks operate in dynamic environments where nodes join or leave, and traffic patterns shift. Static slot assignments are inefficient and inflexible. The entity tracks `slot_requests_pending` and manages bearer slot assignments via `assigned_slots` and `slot_count`. The bearer parameter `min_slots` ensures minimum resource guarantees, while the entity facilitates dynamic re-allocation through the `skynet_convergence_schedule_slots` function. Weighted Fair Queuing, driven by bearer priorities, optimizes slot distribution, ensuring high-QoS traffic receives preferential treatment. This adaptability is a key advantage over static or centralized designs.

7.7 Scalability and Modularity

A flat QoS structure becomes unwieldy as the number of nodes and NPGs increases, complicating scheduling and state management. The 3-level hierarchy addresses this by separating concerns: QoS defines service requirements, bearers manage flow-specific state, and entities coordinate node-level convergence. This modular design scales to support large networks, simplifies debugging, and facilitates maintenance. The entity layer aggregates bearer state, reducing scheduling complexity from $O(n)$ for n bearers to $O(m)$ for m nodes. The hierarchy also supports future enhancements, such as preemption or adaptive QoS, without requiring a system overhaul.

7.8 LTE QCI Compatibility

The 3-level hierarchy draws inspiration from LTE's QCI framework, which uses bearers with QoS profiles to manage diverse traffic types (e.g., VoIP, video, best-effort) per User Equipment (UE). By adapting this model to TDMA, Skynet replaces LTE's EPS bearers with TDMA slot-based bearers, leveraging telecom best practices. The bearer parameters mirror QCI attributes, such as priority and delay budget, ensuring compatibility with established standards. This alignment reduces design risk, enhances interoperability with telecom systems, and provides a familiar framework for engineers, making it easier to develop and maintain the system.

7.9 MQTT Compatibility

Skynet's convergence layer maps its QoS levels to MQTT's QoS semantics (ISO/IEC 20922:2016) and LTE's QCI framework (3GPP TS 23.203, Release 15), enabling compatibility with both standards for tactical communications. MQTT defines three QoS levels:

- **QoS 0 (At Most Once)**: Best-effort delivery with no acknowledgment or retransmission. Suitable for non-critical data like tactical chat where packet loss is tolerable.
- **QoS 1 (At Least Once)**: Guarantees at least one delivery with acknowledgment (ACK), allowing duplicates. Used for data requiring delivery, such as position location information (PLI), where duplicates are acceptable.
- **QoS 2 (Exactly Once)**: Ensures exactly one delivery via a four-way handshake (`PUBLISH`, `PUBREC`, `PUBREL`, `PUBCOMP`). Critical for command and control (C2) messages requiring no loss or duplication.

Skynet implements these using its `SkyNetBearerQoS` and `SkyNetBearer` structures, configured via `skynet_convergence_init`. QoS 0 skips `reorder_queue` and ACKs, QoS 1 uses ACKs (`SKYNET_MSG_ACK`) with retransmission on timeout (100ms), and QoS 2 employs a handshake with strict ordering and deduplication via `seq_no`. All Network Protocol Groups (NPGs) and message types are mapped to QoS levels, as shown in Table 4.

This mapping ensures military-grade performance: QoS 2 meets sub-50ms latency for C2 (`npg_c2`, NPG 100) and control (`npg_control`, NPG 1), QoS 1 supports reliable PLI (`npg_pli`, NPG 6) with 100ms latency, and QoS 0 optimizes bandwidth for chat (`npg_chat`, NPG 29). The `SkyNetConvergenceEntity` dynamically allocates slots via Weighted Fair Queuing, prioritizing high-QoS bearers, while `reorder_queue` ensures reliability for QoS 1 and 2, aligning with LTE QCI's packet error loss rates (e.g., 10^{-6} for QCI 5).

In conclusion, the 3-level hierarchy, inspired by LTE QCI, enables Skynet to replicate MQTT's QoS semantics, ensuring scalable, low-latency, and reliable communication for swarm drone coordination and other tactical applications.

Table 4: Skynet QoS Mapping to MQTT and LTE QCI

QoS	Priority	NPG	QCI	Budget	Slots	Type
0	15	29	QCI 9	300ms	1	2
0	14	102	QCI 9	300ms	1	2, 5
0	13	103	QCI 9	300ms	1	2, 4
1	7	6	QCI 7	100ms	2	5
1	6	7	QCI 7	100ms	2	5
2	3	1	QCI 5	50ms	3	0, 1, 3
2	2	100	QCI 3	50ms	3	3, 4, 6
2	1	101	QCI 5	50ms	3	3, 6

8 Deploying Private LTE 4G Network

This section provides a concise guide for administrators to deploy a private LTE 4G network using srsRAN with a USRP B200mini or USRP X310 FPGA-based Software Defined Radios (SRD) and iPhone User Equipment as UE LTE terminals in battlefield EDGE scenarios. Private LTE 4G Network acts as a Data Link Layer (Media Access Control) for Skynet, which operates as Network and Transport Layer framework for drone swarm coordination. It outlines the main functionality of srsRAN, key configuration files, and a technical overview of LTE 4G components relevant to the deployment.

8.1 LTE 4G Overview and srsRAN Components

LTE 4G is a cellular standard providing high-speed, low-latency communication through a layered architecture. Key components include:

- **Evolved NodeB (eNB)**: Manages radio resources, handles UE connections, and interfaces with the core network. In srsRAN, the eNB is implemented using a USRP X310 for radio transmission/reception.
- **Evolved Packet Core (EPC)**: Handles network control, authentication, and IP connectivity. srsRAN uses Open5GS or similar for EPC functions (MME, HSS, SP-GW).
- **UE**: Devices (e.g., iPhones) connecting to the eNB. Requires programmable SIMs for private network authentication.
- **Protocol Stack**: Includes PHY (physical layer), MAC (medium access control), RLC (radio link control), PDCP (packet data convergence protocol), and RRC (radio resource control) for reliable data transfer and QoS management.

srsRAN leverages open-source software and COTS hardware (e.g., USRP X310) to create a private LTE network, supporting secure, low-latency communication for tactical applications like drone coordination and C2C.

8.2 Deployment Steps

To deploy srsRAN at the battlefield EDGE:

1. **Hardware Setup**:
   - Use a USRP X310 with a GPSDO (e.g., Leo Bodnar) for clock synchronization.
   - Attach a directional antenna (e.g., VERT2450) to minimize RF footprint.
   - Connect to a rugged server (e.g., Intel NUC, ∼$500) running Ubuntu 22.04+.
2. **Software Installation**:
   - Install srsRAN: `https://github.com/srsran/srsRAN_4G`, build using instructions.
   - Install dependencies: UHD drivers (`libuhd-dev`), Open5GS, and OpenSSL.
   - Configure kernel for GTP-U: `modprobe gtp`.
3. **Network Configuration**:
   - Set up a private IP range (e.g., 172.16.0.0/16) for EPC and eNB.
   - Ensure firewall allows UDP ports (e.g., 2152 for GTP-U, 36412 for S1AP).
4. **SIM Provisioning**:
   - Use programmable USIMs (e.g., sysmoUSIM) with custom IMSI and keys.
   - Update `user_db.csv` in `[hss]` with UE credentials.
5. **Configuration Files**: Customize the following srsRAN configuration files (located in `/root/.config/srsran/`).

8.3 Configuration Files

Below are key configurations for srsRAN, tailored for battlefield EDGE with Skynet integration. Only essential parameters are shown; defaults suffice for others.

8.3.1 enb.conf

Configures the eNB for radio and network settings.

```
[enb]
enb_id = 0x19B
mcc = 001
mnc = 01
mme_addr = 127.0.1.100
gtp_bind_addr = 127.0.1.1
s1c_bind_addr = 127.0.1.1
n_prb = 50

[rf]
dl_earfcn = 3350
tx_gain = 15  % Low power to reduce detectability
rx_gain = 40
device_name = uhd
device_args = type=x300,master_clock_rate=184.32e6

[enb_files]
sib_config = sib.conf
rr_config = rr.conf
rb_config = rb.conf
```

8.3.2 epc.conf

Configures the EPC for authentication and IP connectivity.

```
[mme]
mme_code = 0x1a
mme_group = 0x0001
tac = 0x0007
mcc = 001
mnc = 01
mme_bind_addr = 127.0.1.100
apn = srsapn
dns_addr = 8.8.8.8
encryption_algo = EEA2
integrity_algo = EIA2

[hss]
db_file = user_db.csv

[spgw]
gtpu_bind_addr = 127.0.1.100
sgi_if_addr = 172.16.0.1
sgi_if_name = srs_spgw_sgi
```

8.3.3 rb.conf

Defines QoS for radio bearers, aligned with Skynet's QoS requirements.

```
qci_config = ({
  qci = 7;
  pdcp_config
    = { discard_timer = -1;
        pdcp_sn_size = 12; }
  rlc_config
    = { ul_um = { sn_field_length = 10; };
        dl_um = { sn_field_length = 10;
                  t_reordering = 45; }; }
  logical_channel_config
    = { priority = 13;
        prioritized_bit_rate = -1;
        bucket_size_duration = 100;
        log_chan_group = 2; }
},{
  qci = 9;
  pdcp_config
    = { discard_timer = 150;
        status_report_required = true; }
  rlc_config
    = { ul_am = { t_poll_retx = 120;
                  poll_pdu = 64;
                  poll_byte = 750;
                  max_retx_thresh = 16; };
        dl_am = { t_reordering = 50;
                  t_status_prohibit = 50; }; }
  logical_channel_config
    = { priority = 11;
        prioritized_bit_rate = -1;
        bucket_size_duration = 100;
        log_chan_group = 3; }
});
```

8.3.4 rr.conf

Configures radio resources and cell parameters.

```
cell_list = ({
  cell_id = 0x01;
  tac = 0x0007;
  pci = 1;
  dl_earfcn = 3350;
  ho_active = false;
});
```

8.3.5 sib.conf

Defines System Information Blocks for network discovery.

```
sib1 = { intra_freq_reselection = "Allowed";
         q_rx_lev_min = -65;
         cell_barred = "NotBarred";
         si_window_length = 20; }
sib2 = { rr_config_common_sib
           = { rach_cnfg = { num_ra_preambles = 52;
                             preamble_init_rx_target_pwr = -104;
                             pwr_ramping_step = 6;
                             ra_resp_win_size = 10;
                             mac_con_res_timer = 64;
                             max_harq_msg3_tx = 4; }; }; }
sib3 = { cell_reselection_common
           = { q_hyst = 2; };
         intra_freq_reselection
           = { q_rx_lev_min = -61;
               p_max = 23;
               s_intra_search = 5; }; }
```

8.3.6 ue.conf

Configures UE (Skynet Client) settings for network attachment.

```
[rf]
dl_earfcn = 3350
tx_gain = 80
device_name = uhd
device_args = type=x300,master_clock_rate=184.32e6

[usim]
mode = soft
algo = milenage
opc = 63BFA50EE6523365FF14C1F45F88737D
k = 00112233445566778899aabbccddeeff
imsi = 001010123456780
imei = 353490069873319

[nas]
apn = srsapn
apn_protocol = ipv4
```

8.4 Operation

1. Start EPC: `srsmbd -c epc.conf`.
2. Start eNB: `srsenb -c enb.conf`.
3. Configure iPhones with programmable USIMs and attach to the network (MCC=001, MNC=01).
4. Integrate with Skynet: Configure `npg_c2` (NPG 100) and `npg_pli` (NPG 6) for drone C2 and PLI, using QoS 2 for low-latency C2 (50 ms) and QoS 1 for PLI (100 ms).

8.5 Security and Visibility

- **Encryption**: Use AES-256 (EEA2) and integrity protection (EIA2) in `epc.conf` to secure communications.
- **Low Visibility**: Set `tx_gain = 15` and use directional antennas to minimize RF footprint.
- **Interference Mitigation**: Select Band 3 (EARFCN 3350) to avoid commercial spectrum.

8.6 Troubleshooting

- **UE Connection Issues**: Verify IMSI in `user_db.csv`, ensure `dl_earfcn` matches, and check GPSDO synchronization.
- **GTP-U Errors**: Ensure `modprobe gtp` and correct `gtpu_bind_addr`.
- **Logs**: Set `all_level = debug` in `enb.conf` and `epc.conf` for detailed diagnostics.

9 Swarm Modeling and Simulation Architecture

The simulation of large-scale drone swarms, as implemented in the Link32 protocol, requires efficient and scalable numerical models to capture the dynamics of 256 to 50,000 drones operating in a shared environment. Two complementary approaches are employed: a discrete ordinary differential equation (ODE) model for individual drone tracking and a continuum partial differential equation (PDE) model for approximating swarm behavior as a density field. These models are implemented in the `skynet_ode.c` and `skynet_pde.c` codes, respectively, using the PETSc library for parallel numerical computations.

9.1 Swarm Modeling Approaches

9.1.1 Discrete ODE Model

The ODE model represents each drone as an individual agent with position $x_i(t)$ and velocity $v_i(t)$ in a 1D or 2D domain, governed by a system of ordinary differential equations:

$$\frac{dx_i}{dt} = v_i, \quad \frac{dv_i}{dt} = u_i + f_{env,i} + f_{swarm,i},$$

where u_i is the control input from the Skynet/Link32 protocol (e.g., velocity adjustments via UDP packets), $f_{env,i}$ is an environmental force (e.g., wind), and $f_{swarm,i}$ accounts for inter-drone interactions, such as repulsion to avoid collisions. The repulsion term $f_{swarm,i}$ is defined as a sum over all other drones

$j \neq i$. If the distance between drones i and j, denoted $\|x_i - x_j\|$, is less than a minimum threshold r_{min}, the repulsion force is:

$$f_{swarm,i,j} = \frac{\kappa(r_{min} - \|x_i - x_j\|)}{\|x_i - x_j\| + \epsilon},$$

where $\kappa = 0.05$, $r_{min} = 1.0$, and $\epsilon = 10^{-6}$ ensures numerical stability. Otherwise, if $\|x_i - x_j\| \geq r_{min}$, the repulsion force is zero. The total repulsion force is:

$$f_{swarm,i} = \sum_{j \neq i} f_{swarm,i,j}.$$

This model is suitable for smaller swarms (e.g., 256 drones) where individual tracking is feasible and integrates directly with Skynet/Link32 control signals for precise drone coordination.

9.1.2 Continuum PDE Model

For large swarms (up to 50,000 drones), a continuum approach is adopted, treating the swarm as a density field $\rho(x, y, t)$ with an associated velocity field $u_x(x, y, t)$. The dynamics are governed by a system of partial differential equations, comprising a continuity equation for density conservation and a momentum equation for velocity evolution:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0, \quad \frac{\partial u_x}{\partial t} = f_{control} + f_{env},$$

where $u = (u_x, 0)$ is the velocity field (simplified to 1D flow for initial testing), $f_{control}$ represents Skynet/Link32 control inputs (e.g., density-based velocity adjustments), and f_{env} is an environmental force field (e.g., wind, set to a constant 0.1 in initial simulations). The PDE model is discretized on a 2D grid (e.g., 100×100) using finite differences, with the divergence term approximated as:

$$\nabla \cdot (\rho u_x) \approx \frac{(\rho u_x)_{i,j} - (\rho u_x)_{i-1,j}}{\Delta x}.$$

This approach is computationally efficient for large swarms, capturing collective behavior while integrating Skynet/Link32 control signals to guide the swarm density and velocity fields.

9.2 Architecture of Simulation Codes

Both `skynet_ode.c` and `skynet_pde.c` are implemented using the PETSc library, leveraging its parallel data structures and time-stepping solvers to ensure scalability across multiple MPI processes. The codes are designed to interface with the Skynet/Link32 protocol, which provides control inputs via UDP packets for real-time swarm coordination.

9.2.1 `skynet_ode.c` Architecture

The `skynet_ode.c` code simulates the discrete ODE model for individual drone dynamics. Its key components are:

- **Data Structure**: A global vector X of size $2N$ (where N is the number of drones, e.g., 17 in the test case) stores positions x_i and velocities v_i. An additional vector env_data holds environmental forces (e.g., wind, set to 0.1 per drone).
- **RHS Function**: The `RHSFunction` computes the time derivatives $\frac{dx_i}{dt} = v_i$ and $\frac{dv_i}{dt} = u_i + f_{env,i} + f_{swarm,i}$, incorporating repulsion terms to prevent collisions and Skynet/Link32 control inputs (currently a placeholder, to be updated with UDP-parsed signals).
- **Solver**: The PETSc `TS` solver with the explicit Runge-Kutta method (`TSRK`) advances the solution with a time step $\Delta t = 0.1$. The simulation runs for a short duration (e.g., 1 second) to ensure stability during testing.
- **Initialization**: Drones are initialized with random positions in a domain (e.g., [0, 10]) using `PetscRandom`, with zero initial velocities.
- **Output**: The final state is written to `skynet_ode.txt`, containing position and velocity pairs for each drone (e.g., 34 values for 17 drones).
- **Parallelism**: The code supports MPI parallelism, distributing drones across processes, though the test output indicates a single process, suggesting a configuration to be optimized for larger swarms.

The code is designed for scalability, with plans to increase N to 256 or 50,000 drones by optimizing MPI distribution and integrating Skynet/Link32 control logic.

9.2.2 `skynet_pde.c` Architecture

The `skynet_pde.c` code implements the continuum PDE model on a 2D grid. Its architecture includes:

- **Data Structure**: A distributed array (`DMDA`) manages a 2D grid (e.g., 10×10 in testing, scalable to 100×100) with 2 degrees of freedom per grid point (ρ, u_x). A global vector U stores the state, and a separate vector env_data holds environmental forces (e.g., wind, set to 0.1 per grid point).
- **RHS Function**: The `RHSFunction` computes the time derivatives: $\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho u_x)$ using finite differences, and $\frac{\partial u_x}{\partial t} = f_{control} + f_{env}$, with $f_{control}$ as a placeholder for Skynet/Link32 inputs.
- **Solver**: The PETSc `TS` solver with the explicit Runge-Kutta method (`TSRK`) advances the solution with $\Delta t = 0.01$. The simulation runs for a short time (e.g., 0.1 seconds) to verify stability.

- **Initialization**: The density ρ is initialized as a Gaussian blob centered at (50, 50) in a 100×100 domain, $\rho(x, y) = e^{-((x-50)^2+(y-50)^2)/10}$, with zero initial velocity.
- **Output**: The final state is written to `skynet_pde.txt`, containing density and velocity values (e.g., 200 values for a $10 \times 10 \times 2$ grid).
- **Parallelism**: The `DMDA` structure supports MPI parallelism, distributing the grid across processes, though the test output indicates a single process, requiring optimization for larger grids.

The PDE code is designed to scale to larger grids (e.g., 100×100) and integrate Skynet/Link32 control signals to adjust the velocity field based on swarm density or external inputs.

9.3 Future Enhancements

Both codes are currently stable for small-scale tests (17 drones for ODE, 10×10 grid for PDE). Future work includes:

- Scaling the ODE model to 256–50,000 drones with optimized MPI parallelism.
- Increasing the PDE grid to 100×100 or larger for large swarms.
- Integrating Skynet/Link32 control logic, parsing UDP packets for real-time control inputs (e.g., target positions or velocity fields).
- Developing a hybrid ODE-PDE model using PETSc's `DMComposite` to combine individual drone tracking with continuum density modeling.

These enhancements will ensure the simulation framework supports the full range of swarm sizes and integrates seamlessly with the Skynet/Link32 protocol for real-time drone swarm coordination.

10 Dedication

The Skynet is dedicated to Vitalii Karvatskyi.

11 Conclusion

Link32 and Skynet provide a robust framework for tactical communication, combining low-latency, security, and scalability for military applications, including swarm drone coordination. Future improvements could address slot scalability beyond 256 nodes, automated key distribution to replace manual provisioning, retransmission mechanisms for dropped messages, and enhanced deduplication to support larger networks.

The proposed LTE 4G-based cluster hybrid topology provides an affordable, scalable solution for coordinating 5,000 MAVs over a 30 km radius, with each drone transmitting a 100-byte payload at 10 Hz and 5–10 cluster leads streaming 200 kbps video for C2C monitoring. The hybrid architecture, combining a terrestrial base station and UAV relay, achieves full coverage at a cost of approximately $\sim$ \$10,000, leveraging open-source srsRAN and COTS hardware. Key algorithms, including PSO, DRL, and consensus-based methods, enable efficient path planning, collision avoidance, task allocation, formation control, and communication optimization. Future work should explore 5G integration for higher capacity and anti-jamming techniques for battlefield reliability.

References

[1] Fotouhi, A., et al., "Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges," *arXiv preprint arXiv:1809.01752*, 2019.

[2] Muruganathan, S. D., et al., "An Overview of 3GPP Release-15 Study on Enhanced LTE Support for Connected Drones," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 140–146, 2021.

[3] Awasthi, S., et al., "Artificial Intelligence Supervised Swarm UAVs for Reconnaissance," *Data Science and Analytics: 5th International Conference on Recent Developments in Science, Engineering and Technology, REDSET 2019, Gurugram, India, November 15–16, 2019, Revised Selected Papers, Part I 5*, Springer, pp. 375–388, 2020.

[4] Nguyen, H. C., et al., "How to Ensure Reliable Connectivity for Aerial Vehicles Over Cellular Networks," *IEEE Access*, vol. 6, pp. 12304–12317, 2018.

[5] Yue, X., et al., "Software Defined Radio and Wireless Acoustic Networking for Amateur Drone Surveillance," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 90–97, 2018.

[6] Shakhatreh, H., et al., "A Survey on UAV-Assisted Wireless Communications: Recent Advances and Future Trends," *ScienceDirect*, 2023.

[7] Wang, J., and Jiang, C., *Flying Ad Hoc Networks*, Springer, 2022.

[8] Sai, S., et al., "A Comprehensive Survey on Artificial Intelligence for Unmanned Aerial Vehicles," *IEEE Open Journal of Vehicular Technology*, vol. 4, pp. 713–738, 2023.

[9] Alqudsi, Y., "Coordinated Formation Control for Swarm Flying Robots," *2024 1st International Conference on Emerging Technologies for Dependable Internet of Things (ICETI)*, IEEE, Sana'a, Yemen, pp. 1–8, 2024.

[10] Colomina, I., and Molina, P., "Unmanned Aerial Vehicles and Their Applications in Remote Sensing," *Survey Review*, vol. 46, no. 336, pp. 159–174, 2014.

[11] U.S. Department of Defense, "MIL-STD-6016: Tactical Data Link (TDL) J-Message Standard," 2008.

[12] U.S. Department of Defense, "Link 16 Network Management and Operations," ADA404334, 2003.

[13] 3GPP, "TS 23.203: Policy and Charging Control Architecture (Release 15)," 2018.

[14] ISO/IETF standard "MQTT Version 5.0,", 2019.

[15] Maksym Sokhatsky. A Non-Blocking C Implementation of a Link16 Service-Oriented Architecture: Design and Verification. 2025.

[16] Maksym Sokhatsky. A Non-Blocking C Implementation of a GSM HSS and PDCP Services. 2025.

[17] J. Li and Y. Zhang, "TDMA-Based Scheduling for Tactical Wireless Networks," IEEE Transactions on Vehicular Technology, vol. 68, no. 5, pp. 4987–4999, 2019.

[18] A. Sharma and P. Kumar, "Communication Protocols for UAV Swarm Coordination: A Survey," Journal of Network and Computer Applications, vol. 172, 2020.

[19] Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[20] Liu, J., and Yang, J., "Path Planning for UAVs Using A* and Genetic Algorithms," *Journal of Unmanned Aerial Systems*, vol. 5, no. 3, pp. 45–52, 2019.

[21] Kennedy, J., and Eberhart, R., "Particle Swarm Optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[22] Zhang, Y., and Li, S., "UAV Path Planning Using PSO in Dynamic Environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 4, pp. 1654–1663, 2017.

[23] Dorigo, M., and Gambardella, L. M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[24] Yang, Q., and Yoo, S. J., "Optimal UAV Path Planning Using Ant Colony Optimization in FANETs," *Ad Hoc Networks*, vol. 78, pp. 54–63, 2018.

[25] Storn, R., and Price, K., "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[26] Das, S., and Suganthan, P. N., "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2016.

[27] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[28] Welch, G., and Bishop, G., "An Introduction to the Kalman Filter," *University of North Carolina Technical Report*, TR 95-041, 2006.

[29] Lin, Y., and Saripalli, S., "Sampling-Based Path Planning for UAVs in 3D Environments," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 656–670, 2018.

[30] Chen, J., and Zhang, W., "Spatial-Temporal Path Planning for UAV Swarms," *Journal of Intelligent and Robotic Systems*, vol. 99, no. 2, pp. 233–245, 2020.

[31] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[32] Zhang, Z., and Zhao, S., "UAV Collision Avoidance Using Artificial Potential Fields," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1745–1756, 2021.

[33] Gageik, N., Benz, P., and Montenegro, S., "Obstacle Detection and Collision Avoidance for a UAV with Complementary Low-Cost Sensors," *IEEE Access*, vol. 3, pp. 599–609, 2015.

[34] Lin, L., and Goodrich, M. A., "UAV Collision Avoidance Using Reactive Strategies," *Journal of Field Robotics*, vol. 37, no. 5, pp. 814–832, 2020.

[35] Reynolds, C. W., "Flocks, Herds, and Schools: A Distributed Behavioral Model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[36] Olfati-Saber, R., "Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[37] Ross, S., and Melik-Barkhudarov, N., "Vision-Based Navigation for UAVs," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3745–3752, 2018.

[38] Wang, Q., and Zhang, H., "Vision-Based Obstacle Avoidance for UAV Swarms," *Journal of Unmanned Aerial Systems*, vol. 6, no. 2, pp. 89–97, 2020.

[39] Mnih, V., et al., "Human-Level Control through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[40] Schulman, J., et al., "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[41] Theraulaz, G., and Bonabeau, E., "A Brief History of Stigmergy," *Artificial Life*, vol. 5, no. 2, pp. 97–116, 1999.

[42] Beckers, R., et al., "From Local Actions to Global Tasks: Stigmergy and Collective Robotics," *Artificial Life IV*, pp. 181–189, 2000.

[43] Olfati-Saber, R., Fax, J. A., and Murray, R. M., "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[44] Ren, W., and Beard, R. W., "Consensus Seeking in Multiagent Systems Under Dynamically Changing Interaction Topologies," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 776–781, 2007.

[45] Low, K. H., and Chen, J., "A Virtual Navigator Model for UAV Swarm Coordination," *IEEE International Conference on Robotics and Automation*, pp. 1234–1240, 2019.

[46] Zhang, X., and Liu, Y., "Dynamic Path Planning with Virtual Navigator for UAV Swarms," *Journal of Intelligent and Robotic Systems*, vol. 105, no. 3, pp. 45–56, 2022.

[47] Consolini, L., et al., "Leader-Follower Formation Control of Nonholonomic Mobile Robots with Input Constraints," *Automatica*, vol. 44, no. 5, pp. 1343–1349, 2008.

[48] Wang, L., and Liu, J., "UAV Formation Control Using Leader-Follower Strategies," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 6, pp. 3210–3221, 2019.

[49] Lewis, M. A., and Tan, K. H., "High Precision Formation Control of Mobile Robots Using Virtual Structures," *Autonomous Robots*, vol. 4, no. 4, pp. 387–403, 1997.

[50] Ren, W., and Atkins, E., "Distributed Multi-Vehicle Coordinated Control via Local Information Exchange," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 10, pp. 1002–1033, 2008.

[51] Balch, T., and Arkin, R. C., "Behavior-Based Formation Control for Multi-Robot Teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[52] Reynolds, C. W., "Interaction with Groups of Autonomous Characters," *Game Developers Conference*, 2000.

[53] Ren, W., "Consensus Strategies for Cooperative Control of Vehicle Formations," *IET Control Theory and Applications*, vol. 1, no. 2, pp. 505–512, 2007.

[54] Dong, X., et al., "Time-Varying Formation Control for Unmanned Aerial Vehicles: Theories and Applications," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 340–348, 2016.

[55] Zavlanos, M. M., and Pappas, G. J., "Distributed Formation Control with Permutation Symmetries," *IEEE Conference on Decision and Control*, pp. 2894–2899, 2007.

[56] Mesbahi, M., and Egerstedt, M., *Graph Theoretic Methods in Multiagent Networks*, Princeton University Press, 2010.

[57] Li, Y., and Chen, H., "Reactive-Greedy-Reactive in Unmanned Aerial Vehicle Routing," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1595–1598, 2016.

[58] Zhang, J., and Zhao, T., "RGR Protocol for Efficient Routing in FANETs," *Ad Hoc Networks*, vol. 81, pp. 123–134, 2018.

[59] Bitencourt, J. F., et al., "Bee-Inspired Routing Protocols for UAV Networks," *Journal of Network and Computer Applications*, vol. 69, pp. 76–84, 2016.

[60] Karaboga, D., and Ozturk, C., "A Novel Clustering Approach: Artificial Bee Colony (ABC) Algorithm," *Applied Soft Computing*, vol. 11, no. 1, pp. 652–657, 2012.

[61] Chen, M., et al., "EPOS: Energy-Efficient Planning and Optimization for UAV Swarms," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2598–2611, 2019.

[62] Liu, Q., and Zhang, Y., "Energy-Aware Task Allocation for UAV Swarms Using EPOS," *Journal of Intelligent and Robotic Systems*, vol. 103, no. 2, pp. 34–45, 2021.

[63] Zhang, K., and Yang, Z., "Graph Attention Networks for UAV Swarm Control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 4012–4023, 2020.

[64] Wang, H., and Li, J., "Decentralized Actor-Critic for Multi-UAV Coordination," *IEEE International Conference on Robotics and Automation*, pp. 5678–5684, 2021.