Research Report

Communication and Networking Systems for Autonomous Aerial Networks

Saksham Khandelwal
24b0965@iitb.ac.in

October 2025

Abstract

This report documents the research and experimental work conducted during October 2025 as part of the ongoing exploration of mobile ad-hoc communication systems and their applications in autonomous aerial networks. The primary focus was on understanding and comparing routing protocols such as BATMAN (Better Approach To Mobile Adhoc Networking), OLSR (Optimized Link State Routing), AODV (Ad hoc On-Demand Distance Vector), and DSDV (Destination-Sequenced Distance-Vector), along with investigating AI-enhanced routing strategies.

Practical work included installing and configuring ROS 2 and Gazebo to simulate communication between two drones using the DDS (Data Distribution Service) middleware through ROS's publish–subscribe mechanism. Further, experiments were conducted to establish peer-to-peer communication between two laptops over a shared network using IP and port-based message exchange. To strengthen the theoretical foundation, networking principles across all OSI layers were studied in detail, emphasizing how transport protocols (UDP/TCP) and middleware layers interact in real-time systems.

Finally, study extended to evaluating different peer-to-peer networking paradigms-Direct, Ad-hoc, Mesh, and MANET — analyzing their feasibility for drone swarm communication. Several hands-on experiments were performed to configure and test Wi-Fi Direct and Ad-hoc networks on Ubuntu-based systems, assessing connectivity, latency, and range across multiple laptops. The outcomes provide a comprehensive understanding of decentralized communication models and their potential applicability in autonomous drone networks.

Introduction

The objective of this month's research was to develop a strong conceptual and experimental understanding of mobile ad-hoc networks (MANETs) and their potential use in autonomous drone communication systems. Modern drone operations typically rely on centralized ground stations for telemetry and control. However, as swarm-based and cooperative behaviors become more common, there is a growing need for decentralized, peer-to-peer communication systems that can operate autonomously without fixed infrastructure.

To address this, the study began with an extensive review of existing ad-hoc routing protocols—**BATMAN**, **OLSR**, **AODV**, and **DSDV**—along with an exploration of emerging AI-assisted routing approaches. The **AODV protocol** was analyzed in depth, focusing on route discovery, maintenance, and sequence number management mechanisms that enable dynamic, self-organizing topologies.

On the experimental side, **ROS 2** and **Gazebo** were installed and configured to simulate communication between two drones using the **Data Distribution Service (DDS)** middleware, which forms the basis of ROS 2's publish–subscribe model. This involved studying how DDS handles discovery, transport (RTPS/UDP), and real-time data exchange between distributed agents. Additional experiments involved direct communication between laptops using IP and port configurations, enabling real-time message exchange across a shared local network.

To gain a deeper understanding of how such communication frameworks operate, the OSI model was studied comprehensively—examining the roles of the Application, Presentation, Session, Transport, Network, Data Link, and Physical layers. This provided insight into how network layers interact in both infrastructure-based and ad-hoc systems.

Finally, the focus shifted toward comparing various peer-to-peer architectures—**Direct**, **Ad-hoc**, **Mesh**, and **MANET**—evaluating their advantages, limitations, and use cases. Experiments were conducted to set up and test Wi-Fi Direct and Ad-hoc connections between multiple laptops running Ubuntu, followed by range and latency evaluations to assess their suitability for drone swarm communication.

Overall, the month's work bridged theoretical understanding and experimental validation, contributing to the broader goal of enabling reliable, infrastructure-independent communication in drone networks.

1. Week 1: Study of Mobile Ad-hoc Routing Protocols

During the first week, the focus was on gaining a conceptual understanding of Mobile Ad-hoc Networks (MANETs) and exploring various routing protocols designed for such dynamic and decentralized environments. MANETs consist of mobile nodes that communicate without relying on fixed infrastructure, where each node acts as both a host and as a router. Several prominent routing protocols were studied to understand their design philosophies, operational mechanisms, and comparative advantages.

1.1. Overview of MANET Routing Protocols

1.1.1. BATMAN (Better Approach To Mobile Adhoc Networking)

BATMAN is a proactive routing protocol that simplifies network management by allowing each node to maintain knowledge only of the best next hop towards every destination, rather than the entire network topology. This distributed approach minimizes complexity and enables scalability in medium-sized networks.

1.1.2. OLSR (Optimized Link State Routing)

OLSR is another proactive protocol that improves the efficiency of classical link-state routing by introducing *Multi-Point Relays (MPRs)*. MPRs reduce redundant message flooding during topology dissemination, lowering overhead while maintaining up-to-date routes. It relies on periodic Hello and Topology Control messages for maintaining connectivity.

1.1.3. AODV (Ad hoc On-Demand Distance Vector Routing)

AODV is a reactive routing protocol that constructs routes only when required by a source node, minimizing unnecessary control message overhead. It combines the on-demand nature of dynamic path creation with distance-vector concepts, where each node maintains information about the next hop and hop count to a destination.

1.1.4. DSDV (Destination-Sequenced Distance Vector Routing)

DSDV is a proactive distance-vector routing protocol that enhances the classic RIP approach by incorporating sequence numbers to prevent routing loops. It periodically exchanges full or incremental routing updates to maintain consistency across nodes.

1.1.5. AI-Enhanced Routing

AI-based routing introduces machine learning models to predict node mobility, optimize route selection, or adapt to dynamic network conditions. These methods can improve reliability and reduce delay but require computational and data overhead.

1.2. Detailed Study of AODV

AODV (Ad hoc On-Demand Distance Vector) routing was studied in detail as a representative reactive protocol. It dynamically discovers routes only when necessary, ensuring efficient use of bandwidth and memory. The protocol employs three types of control messages: Route Request (RREQ), Route Reply (RREP), and Route Error (RERR).

1.2.1. Operation of AODV

Route Discovery. When a node requires a route to a destination, it broadcasts a **RREQ** packet containing the destination address and sequence number. Neighboring nodes either reply with a **RREP** if they have a valid route or rebroadcast the RREQ further. This continues until the destination or an intermediate node with a fresh route is found.

Route Reply. Upon receiving a RREQ, the destination (or an intermediate node with an active route) sends a **RREP** along the reverse path. Each node on this path updates its routing table with the next hop, hop count, and lifetime information, establishing a forward route to the destination.

Route Maintenance. When a link break is detected (e.g., due to node mobility), a **RERR** message is sent to upstream nodes, notifying them to invalidate affected routes. If communication is still required, the source initiates a new route discovery process.

1.2.2. Key Features of AODV

- **Loop-free routing:** Maintained via destination sequence numbers.
- **On-demand operation:** Reduces overhead by creating routes only when required.
- **Supports unicast and multicast:** Can handle communication to one or multiple nodes.

1.2.3. Advantages and Disadvantages of AODV

Advantages

- Efficient in highly dynamic networks due to on-demand route setup.
- Reduced memory usage since inactive routes are discarded.
- Simple and modular design, making it easy to simulate and implement.

Disadvantages

- **Initial delay:** Route discovery introduces latency before data transfer.
- **Flooding overhead:** Broadcasting RREQs can cause congestion in dense networks.
- **Scalability limits:** Performance degrades in very large or frequently changing topologies.

2. Week 2: ROS 2 Simulation and Communication

2.1. Overview of Activities

The primary objective for this week was the configuration of the ROS 2 and Gazebo simulation environments. After familiarization with the fundamental concepts, a simulation was developed using a premade SDF file of a quadcopter, from which the essential model and movement mechanisms were extracted. A ROS 2 communication simulation was subsequently developed in Python, comprising:

- A communication node to read and publish data from a text file.
- A real-time communication node to publish data live.
- A movement node to actuate the drone based on instructions.

It is intended that the ROS Python scripts, along with usage instructions, will be included in an appendix to this report.

2.2. Analysis of the ROS 2 Communication Backbone

To gain a deeper understanding of the system, an exploration of the underlying communication mechanisms of ROS 2 was conducted. ROS 2 is built upon the DDS (Data Distribution Service) middleware standard and the RTPS (Real-Time Publish–Subscribe) protocol, which runs over UDP.

2.2.1. System Architecture

- A **Node** is an operating system process or thread that uses a ROS client library (`rclpy`/`rclcpp`) to create publishers, subscribers, services, and actions.
- ROS 2 uses a middleware abstraction layer called **RMW** (ROS Middleware Interface), which typically relies on a DDS implementation such as CycloneDDS or FastDDS.
- DDS implements **discovery**, **serialization**, and the actual **message transport**, typically using RTPS over UDP.

2.2.2. Communication Flow

1. **Publisher Side:** When a node calls `publisher.publish(msg)`, the message is serialized and passed down to the RMW layer. DDS serializes it (usually in CDR format), finds all matching subscribers, and sends the data via RTPS over UDP (or via shared memory if supported).
2. **Transport Layer:** RTPS manages the actual delivery. It may use multicast for discovery and unicast for data transfer.
3. **Subscriber Side:** The DDS middleware receives and deserializes the message, and the ROS executor calls the user-defined callback function.

2.2.3. Discovery and Matching

When a node starts, DDS announces its presence using multicast packets. All participants (publishers and subscribers) share metadata about their topics, types, and QoS settings. Matching happens automatically if both ends share the same topic name, message type, and compatible QoS.

2.2.4. Quality of Service (QoS)

QoS policies determine how messages are delivered:

- **Reliability:** Reliable or best-effort delivery.
- **Durability:** Whether messages persist for late subscribers.
- **History/Depth:** Number of past samples to keep.
- **Deadline, Lifespan, Liveliness:** Advanced timing and availability constraints.

2.3. Layers of the Communication Stack

The data flow from application to the network can be modeled in layers:

1. **Application Layer:** The ROS 2 Python/C++ nodes that call `publish()` and `subscribe()`.
2. **RCL / RMW Layer:** Interface that connects ROS 2 client libraries to the DDS middleware.
3. **DDS Layer:** Handles discovery, QoS enforcement, serialization, and participant management.
4. **RTPS Protocol:** Defines how discovery and data exchange occur over the network.
5. **UDP Transport:** Lightweight transport mechanism that actually moves packets between hosts.

The hierarchy can be summarized as:

$$\text{ROS 2} \rightarrow \text{DDS} \rightarrow \text{RTPS} \rightarrow \text{UDP} \rightarrow \text{Network Layer}$$

2.4. DDS Domain ID and Network Isolation

DDS introduces the concept of a **Domain ID**, which defines a logical communication group. Only participants (nodes) within the same domain can discover each other and exchange data. This allows for the easy isolation of multiple simulations or robot fleets on the same physical network. For example, nodes with `ROS_DOMAIN_ID = 42` communicate freely, while remaining invisible to nodes with different Domain IDs.

3. Week 3: Drone Communication Networks and Peer-to-Peer Alternatives

3.1. Overview

Following the ROS 2 simulation setup, this week's work focused on understanding real-world drone communication systems. The investigation started with the existing setup based on **ExpressLRS (ELRS)** and extended to more general **peer-to-peer (P2P)** and **mesh** networking alternatives that enable distributed communication.

3.2. Analysis of the Current Centralized Setup (ELRS)

In the current configuration, each drone is equipped with an ELRS receiver module, while the ground station acts as the transmitter (TX module). Communication follows a **centralized one-to-many structure**.

Characteristics:

- Only the ground station can communicate with all drones.
- Drones cannot directly exchange data with each other.
- The link operates as a serial-style data stream rather than an IP-based network; hence, DDS (and ROS 2 networking) cannot run directly over it.

3.2.1. ExpressLRS (ELRS): Technical Overview

- **Purpose:** An open-source, ultra-low latency, long-range wireless link for control and telemetry.
- **Underlying technology:** LoRa (Long Range, Low Power) or FLRC modulation.
- **Frequency bands:** 2.4 GHz or 868/915 MHz (region dependent).

3.2.2. ELRS Operational Details

The ground station's TX module sends digital control packets using LoRa modulation, which are received and decoded by the drone's RX module. Telemetry data (e.g., GPS, battery) is sent back to the transmitter, achieving bidirectional communication.

Typical parameters:

- **Latency:** 3–7 ms.
- **Range:** From a few hundred meters up to tens of kilometers.
- **Data rate:** Configurable, suitable for control and telemetry but not high-bandwidth video.

3.2.3. Limitations of ELRS for Swarm Communication

Despite its popularity for RC control, ELRS has several limitations for decentralized swarm applications:

- No built-in support for multi-node or peer-to-peer networking.
- Small packet sizes (typically a few dozen bytes).
- Not designed for high-bandwidth data (e.g., images, video).
- Requires custom firmware modifications to potentially enable drone-to-drone communication.

3.3. Transitioning Toward Peer-to-Peer and Mesh Communication

While the ELRS setup is centralized, the long-term goal is to enable **distributed peer-to-peer communication** between drones. In such a network, drones can share position, velocity, and sensor data directly, allowing for coordination, formation flight, and swarm behavior through fully decentralized data exchange.

3.4. DDS Requirements for Distributed Communication

As established, ROS 2 relies on DDS over UDP/IP. For DDS to function across drones, each drone must have a valid and reachable **IP address**, appropriate UDP ports (7400–7500) must be open, and all nodes must share the same **ROS_DOMAIN_ID**.

3.5. Methods for Creating Peer-to-Peer IP Networks

The primary challenge is connecting multiple drones so that each has a reachable IP address. Several practical methods were studied and are compared below.

Table 1: Comparison of Peer-to-Peer Networking Methods

Method	Description	DDS Capable?	Pros	Cons
Wi-Fi Ad-Hoc	Drones join a shared SSID without router	Yes	Simple, low latency	Setup can be tricky on Linux
Wi-Fi Direct	One drone acts as Group Owner (hotspot)	Yes	Works offline	Single master device
Wi-Fi Mesh (802.11s)	Multi-hop, self-healing network	Yes	Scalable, robust	Requires compatible chipsets
MANET (BATMAN-adv)	Routing at Layer 3 over Wi-Fi/LTE	Yes	Works on any interface	Adds routing overhead
ELRS/LoRa + SLIP/PPP	IP over serial radio link	Limited	Very long range	Low bandwidth, complex setup
LTE + VPN	Each drone connects via LTE	Yes	Global reach	Internet dependency

4. Week 4: Foundations of Networking and Peer-to-Peer Communication

4.1. Fundamental Networking Concepts

Networking refers to the process of connecting multiple devices (nodes) to exchange data. Data is divided into **packets**, each containing source and destination addresses and a payload. These packets are routed through network layers to their destination.

4.2. The OSI Model

Networking is conceptually organized using the 7-layer OSI model:

Table 2: The 7 Layers of the OSI Model

Layer	Name	Purpose and Examples
7	Application	Application-level data (HTTP, DDS, MQTT)
6	Presentation	Data encoding, encryption, compression
5	Session	Management of communication sessions
4	Transport	End-to-end communication (TCP, UDP)
3	Network	IP addressing and packet routing (IPv4/IPv6)
2	Data Link	MAC addressing, local frame delivery
1	Physical	Raw bit transmission (Ethernet, Wi-Fi signals)

4.3. Transport Protocols: TCP vs. UDP

Two primary transport protocols were examined:

Table 3: Comparison of TCP and UDP

Feature	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Type	Connection-oriented	Connectionless
Reliability	High (ensures delivery and order)	Low (best-effort, may lose packets)
Use Cases	Web, file transfer, email	Gaming, video streaming, robotics, DDS

4.4. Wireless Technologies for Drone Networks

For drone swarms, Wi-Fi Ad-Hoc or Mesh networks are typically preferred because they enable direct drone-to-drone communication, operate without base stations, and offer low latency and high data throughput.

Table 4: Comparison of Wireless Physical Layers

Technology	Pros	Cons
Wi-Fi (2.4/5 GHz)	Low latency, high throughput	Limited range, interference
LTE / 5G	Large coverage, high data rate	Requires SIM, higher latency
Custom RF (LoRa)	Long range, low power	Very low bandwidth

4.5. Practical Methods to Establish Peer-to-Peer Networks

Several configurations for drone-to-drone communication were explored:

4.5.1. Wi-Fi Ad-Hoc (IBSS) Mode

All drones join a common SSID and channel, forming a flat, router-less network.

Advantages: Simple setup, supports broadcast/multicast needed for DDS discovery.

Disadvantages: Manual IP configuration required, limited scalability.

4.5.2. Wi-Fi Mesh (802.11s) Mode

An advanced mode where each node relays packets, enabling multi-hop communication.

Advantages: Automatic routing, scalability, self-healing topology.

Disadvantages: Requires compatible hardware (802.11s support).

4.5.3. Wi-Fi Access Point (Star Topology)

One device acts as a central Access Point (AP). While not truly P2P, it allows devices on the same LAN to communicate.

Pros: Easy to set up, compatible with all Wi-Fi devices.

Cons: Centralized bottleneck, not robust for swarm applications.

5. Week 5: Practical Implementation and Setup of Wi-Fi Ad-Hoc Networks

5.1. Understanding Ad-Hoc Mode in Wi-Fi

Conventional Wi-Fi networks operate in **Infrastructure Mode**, where a central **Access Point (AP)** manages all connections. In contrast, **Ad-Hoc Mode**—also known as **IBSS (Independent Basic Service Set)**—eliminates the access point entirely. Devices connect directly to each other, peer-to-peer, with no centralized coordination. This decentralized model is useful for applications like drone swarms where nodes must communicate directly without relying on fixed infrastructure.

5.2. The Network Stack in Ad-Hoc Mode

Ad-hoc mode specifically affects the **Data Link Layer**, where devices negotiate peer associations. All higher layers (IP, UDP/TCP, Application) remain unchanged and function normally over the ad-hoc link.

Table 5: Layer Functionality in Ad-Hoc Mode

Layer	**Behavior in Ad-Hoc Mode**
Physical (PHY)	Wi-Fi NICs operate on the same frequency/channel.
Data Link (MAC)	Frames are addressed directly to each peer's MAC address.
Network (IP)	Each node has a unique static IP in the same subnet.
Transport (TCP/UDP)	Provides reliable (TCP) or real-time (UDP) communication.
Application	Supports DDS, ROS 2, or custom telemetry over UDP.

5.3. Procedure for Setting Up Ad-Hoc Mode on Linux

The following procedure demonstrates how to establish an ad-hoc connection between two Ubuntu laptops:

5.3.1. Step 1 — Identify Wireless Interface

```
iw dev
```

Note the wireless interface name (e.g., `wlp2s0`).

5.3.2. Step 2 — Stop Network Manager (Temporarily)

```
sudo systemctl stop NetworkManager
```

5.3.3. Step 3 — Create Ad-Hoc Network (Laptop 1)

```
sudo iwconfig wlp2s0 mode ad-hoc
sudo iwconfig wlp2s0 essid "MyAdhoc"
sudo iwconfig wlp2s0 channel 6
sudo ifconfig wlp2s0 192.168.10.1 netmask 255.255.255.0 up
```

5.3.4. Step 4 — Join Ad-Hoc Network (Laptop 2)

```
sudo iwconfig wlp2s0 mode ad-hoc
sudo iwconfig wlp2s0 essid "MyAdhoc"
sudo iwconfig wlp2s0 channel 6
sudo ifconfig wlp2s0 192.168.10.2 netmask 255.255.255.0 up
```

5.3.5. Step 5 — Verify Connectivity

```
# From Laptop 2, ping Laptop 1
ping 192.168.10.1
```

Successful replies confirm that packets are being exchanged directly.

5.3.6. Step 6 - Using Netcat to estabilish Communication

5.4. Pending Work

The final task involved configuring ad-hoc networks on multiple laptops and validating direct peer-to-peer packet exchange using `ping` and `netcat`, connectivity consistency, and compatibility of different Wi-Fi chipsets in IBSS mode.
Following this, an exploration of **Wi-Fi Mesh (802.11s)** would be done:

- Ad-hoc (IBSS) provides single-hop peer communication.
- Mesh (802.11s) adds routing capabilities to enable multi-hop paths automatically.

References

[1] What is Networking

[2] Ad hoc On-Demand Distance Vector routing (**AODV**)

[3] ROS 2 Installation and DDS.

[4] ExpressLRS

[5] Basics of Networking and its different layers and types

[6] Setting up Ad-Hoc Network between laptops

[7] Ad hoc On Demand Distance Vector (AODV) Routing Research Paper

[8] Introduction of Mobile Ad hoc Network (MANET)