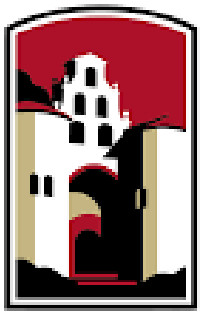


SAN DIEGO STATE UNIVERSITY

DRONE NETWORK SIMULATOR USER MANUAL

Version 1.0 | December 2025

Created by:
Aidan Kiswoto, Alberto Serrano, Guadalupe Zarate, John Guerrero, Kevin Lee, Nolan Conrad, Reem Awad, Rhilo Sotto, Wilson Cao

OVERVIEW

This simulator is a browser-based 3D visualization and testing environment for unmanned aerial vehicle (UAV) network experimentation. It implements a four-layer networking stack: physical (PHY), medium access control (MAC), network, and application layers, with Wi-Fi radio models (802.11ac, 802.11n, Wi-Fi Direct). The platform supports OLSR routing, energy-aware PHY modeling, and multiple mobility and formation patterns suitable for research, education, and algorithm prototyping. Note: This is a mostly finished project as per project project requirements.

Purpose:

This manual provides instructions for installing, configuring, and operating the UAV Network Simulation Platform. It outlines system capabilities and recommended workflows for academic, research, and engineering use.

Key Features:

- Real-time 3D visualization (WebGL)
- CSMA/CA MAC with collision detection
- Energy-aware physical layer modeling
- Multiple formation and mobility models
- Live KPI monitoring and data export
- Automated experiment execution

System Capabilities:

Capability Area	Description
Networking Stack	PHY, MAC, Network, Application
Wireless Models	802.11ac, 802.11n, Wi-Fi Direct
Routing Protocols	OLSR
Simulation Scale	1–100 UAVs
Visualization	Real-time 3D (WebGL)
Metrics	KPIs and exportable datasets

TABLE OF CONTENTS

1. INSTALLATION & QUICK START

1.0 System Requirements

Hardware Requirements:
- CPU: Intel Core i5 (8th gen) / AMD Ryzen 5 or better
  Recommended: Intel Core i7 / AMD Ryzen 7 for 75+ drones
- RAM: 8 GB minimum, 16 GB recommended for large networks
- GPU: Dedicated GPU with WebGL 2.0 support recommended
  Integrated graphics (Intel UHD, AMD Vega) acceptable for N<50
- Storage: 500 MB free disk space for installation
- Display: 1920x1080 resolution or higher recommended

Software Requirements:
- Operating System: Windows 10/11, macOS 10.15+, or Linux (Ubuntu 20.04+)
- Node.js: Version 18.0 or higher (LTS recommended)
  Download from: https://nodejs.org/
- Git: Optional but recommended for version control
  Download from: https://git-scm.com/

Supported Browsers:
- Google Chrome 90+ (recommended for best performance)
- Mozilla Firefox 88+
- Microsoft Edge 90+
- Safari 15+ (macOS only)
- Note: Chrome provides best WebGL performance

1.1 Preparing the Development Environment

Before installation, the user must open the project folder named “CompE560_DroneProject” inside an integrated development environment and launch a terminal from within that folder. All installation and execution steps rely on the terminal being properly positioned in that directory.

Users may opt for operation via *online server* should the below implementation be too difficult or time consuming. The codebase is hosted and operational at: https://compe560project1.netlify.app/

Visual Studio Code (VS Code) is recommended for this project.

1. Download Visual Studio Code from the official website and complete installation using default settings.
2. Launch Visual Studio Code.
3. Select the menu option “File,” then select “Open Folder.”
4. Browse to the location of the project files and select the folder named “CompE560_DroneProject.”
5. Confirm that the folder appears in the Explorer panel on the left side of the interface.

Opening a Terminal:

Using the VS Code Integrated Terminal (Recommended)

1. With the CompE560_DroneProject folder open in Visual Studio Code, select “Terminal” from the top menu and then select “New Terminal.”
2. A terminal window will appear inside the Visual Studio Code interface.
3. Confirm that the terminal is currently operating within the CompE560_DroneProject directory. The directory path displayed above the prompt should end with “CompE560_DroneProject.”

Using the System Terminal (Alternative Method)

Windows (PowerShell):

1. Open PowerShell from the Start menu.
2. Navigate to the CompE560_DroneProject folder by entering the full folder path.

macOS (Terminal):

1. Open the Terminal application.
2. Navigate to the CompE560_DroneProject folder by entering the appropriate directory path.

Linux (Ubuntu/Debian):

1. Open the Terminal application.
2. Navigate to the CompE560_DroneProject folder by entering the appropriate directory path.

Once the terminal is operating inside the correct folder, users may proceed to installation.

1.2 Installation Steps

Step 1: Install Node.js

Windows:
1. Download installer from nodejs.org
2. Run installer, accept defaults
3. Verify installation within the terminal and type:
   powershell:
   node --version
   npm --version

   Expected output: v18.x.x or higher

macOS:
1. Install via Homebrew (recommended):
   brew install node
   Or download installer from nodejs.org
2. Verify in terminal: node --version

Linux (Ubuntu/Debian):
1. Install via package manager:
   sudo apt update
   sudo apt install nodejs npm
2. Verify in terminal: node --version

Step 2: Obtain Project Files

Option A: Git Clone (Recommended)
Note: Type the following in the terminal window
powershell:
git clone <repository-url>
cd CompE560_DroneProject

Option B: Download ZIP
1. Download ZIP from repository
2. Extract to desired location
3. Open terminal in extracted folder

Step 3: Install Dependencies

Note: Type the following in the terminal window
powershell:
npm install

This process:
- Installs React, Three.js, React Three Fiber, Vite, UUID
- Takes 2-5 minutes depending on internet speed
- Creates node_modules/ folder

Common warnings during installation (safe to ignore):
- Deprecated package warnings
- Optional dependency failures
- Peer dependency warnings

Step 4: Start Development Server

Note: Type the following in the terminal window
powershell:
npm run dev

Expected output:
VITE v5.x.x ready in 1234 ms

➜ Local: http://localhost:5173/
➜ Network: http://192.168.x.x:5173/
➜ press h to show help

The server will continue running. Leave the terminal open.

Step 5: Open in Browser

1. Navigate to http://localhost:5173/ by command + click on link
2. Wait 2-3 seconds for initial load
3. You should see:
   - White ground grid with axis labels
   - HDR skybox (realistic lighting)
   - Control panel on left
   - Empty 3D scene (no drones yet)

Option 2: Open using link
Link: https://compe560project1.netlify.app/

1.4 First Simulation Tutorial (5 Minutes)

This tutorial demonstrates core features in a simple scenario.

Phase 1: Static Network Test (2 minutes)

1. Spawn Network
   - Node Count: 25
   - Seed: 12345 (for reproducibility)
   - Click "Spawn Drones"
   - Result: 25 drones appear at fixed random positions

2. Apply Formation
   - Formation dropdown: Select "Grid"
   - Click "Apply Formation"
   - Result: Drones smoothly transition to 5×5 grid pattern
   - Spacing: 6 meters between adjacent drones

3. Start Simulation
   - Click ▶ Play button
   - Observe for 20 seconds
   - Watch right panel "Routing Table" populate

4. Monitor Convergence
   - First 10 seconds: Routes being discovered
   - After 15 seconds: PDR should reach 90-95%
   - Latency: Stabilizes around 5-15 ms
   - Energy: Slowly decreases (99% → 98%)

Phase 2: Mobile Network Test (3 minutes)

5. Reset Simulation
   - Click ⟳ Reset button
   - Scene clears, KPI charts reset

6. Spawn Mobile Network
   - Node Count: 50
   - Formation: Leave as "Random Waypoint" (default)
   - Speed: 5 m/s
   - Click "Spawn Drones"

7. Start Mobile Simulation
   - Click ▶ Play
   - Drones move continuously to random waypoints
   - Links appear/disappear as topology changes

8. Observe Mobility Effects
   - PDR: Lower than static (70-85%)
   - Latency: Higher variance (10-30 ms)
   - Event Log: Shows route changes
   - Energy: Drains faster due to movement

9. Try Different Speeds
   - Pause simulation
   - Speed slider: Try 2 m/s (slow)
   - Resume: PDR improves
   - Try 10 m/s (fast): PDR degrades

10. Formation Transition
    - While running, click Pause
    - Select "V-Formation"
    - Click "Apply Formation"
    - Drones transition to V shape
    - Click Play
    - Observe PDR temporarily drop, then recover

1.5 Understanding the Output

Key Performance Indicators (Bottom Panel):

Avg Queue Fill:
- Definition: Average packets waiting in MAC queues
- Range: 0 to queue capacity (default 50)
- Target: <20 packets (green zone)
- What it tells you: Congestion level

Avg Energy (%):
- Definition: Mean battery level across all drones
- Range: 0% to 100%
- Depletion rate: ~0.1-0.5% per minute (load-dependent)
- What it tells you: Network lifetime

Right Panel Information:

Neighbors Table (MAC Layer):
- Shows: One-hop adjacent drones
- RSSI (dBm): Signal strength (-40 to -90)
- SNR (dB): Signal quality (5 to 35)
- MCS: Modulation scheme (0-9, higher=faster)

Routing Table (Network Layer):
- Shows: Multi-hop routes to all reachable drones
- Next Hop: Where to forward packets
- Hop Count: Number of intermediate nodes
- Metric: Route quality score (lower=better)

2. USER INTERFACE & CONTROLS

2.1 Control Panel Overview (Bottom - Right)

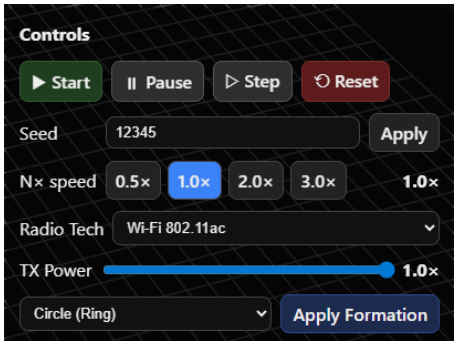


Figure 1: The control panel is located in the bottom right corner of the GUI

Simulation Control Buttons:

[▶ Play] Button
- Function: Start or resume simulation time
- State: Changes to [⏸ Pause] when active
- Behavior: Initiates physics updates, network traffic, mobility
- Shortcut: Spacebar
- Note: First press may have 1-2 second delay for initialization

[⏸ Pause] Button
- Function: Freeze simulation time
- Effect: Drones stop moving, packets stop transmitting
- Use cases: Apply formation, inspect state, take screenshots
- Shortcut: Spacebar (toggle with Play)
- Note: KPI charts stop updating but routing tables remain

[↺ Reset] Button
- Function: Clear all simulation state
- Effect: Removes all drones, clears KPI data, resets time to 0
- Warning: This action cannot be undone
- Shortcut: R key
- Use when: Starting new experiment, changing major parameters

[→ Step] Button
- Function: Advance exactly 1 frame (when paused)
- Frame duration: ~16.67 ms (60 FPS target)
- Use cases: Debugging, frame-by-frame analysis, packet tracing
- Shortcut: Right arrow key
- Requirement: Simulation must be paused first

Figure 2: The Speed Multiplier Button Interface

Speed Multiplier Button:
- Range: 0.5× to 3.0×
- Default: 1.0× (real-time)
- Values: 0.5×, 0.75×, 1.0×, 1.5×, 2.0×, 3.0×
- Effect: Speeds up or slows down simulation time
- Use 0.5×: For detailed observation, complex networks
- Use 3.0×: For faster experiment completion
- Note: Does not affect FPS or rendering speed

Drone Spawning Controls: (Note: system may lag past 75-100 drones)

Node Count Input:
- Range: 1 to 100 drones
- Recommended ranges:
  * Small (10-25): Quick tests, learning, debugging
  * Medium (25-50): Realistic scenarios, balanced performance
  * Large (50-100): Scalability testing, may reduce FPS
- Default spawn area: ±30m horizontal, 5-15m altitude
- Note: Positions random unless seed specified

Random Seed Input: (Note: not entirely functional)
- Purpose: Reproducible experiments
- Effect: Fixes initial positions and waypoint sequences

Figure 3: Button to add drone to simulation

[+ Add Drone] Button:
- Action: Creates N drones at random or seeded positions
- Initialization: Assigns IDs (0 to N-1), initializes network layers
- Visual: Drones appear instantly in 3D scene
- Note: Can spawn multiple times, but old drones remain (use Reset first)

Mobility Controls:

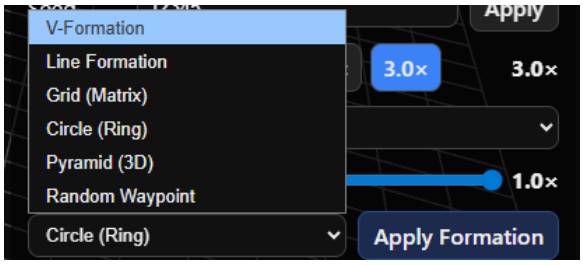


Figure 4: Formation options featured within the dropdown menu

Formation Dropdown:
- Random Waypoint (Default): Continuous random movement
  * Drones pick random target within bounds
  * Move at constant speed
  * Pick new target upon arrival
- V-Formation: Classic wedge shape
  * Leader at front center
  * Two trailing arms at 45° angles
  * Spacing: 6m between adjacent drones
- Line: Horizontal row
  * Drones aligned along X-axis
  * Equal spacing: 6m
- Grid: 2D matrix layout
  * Rows and columns (e.g., 5×5 for N=25)
  * Spacing: 6m in both dimensions
- Circle: Ring formation
  * Radius calculated for even spacing
  * All drones at same altitude
- Pyramid: 3D layered stack
  * Square base, ascending layers
  * Each layer smaller than below

[Apply Formation] Button: (Note: Drones do not move after formation is applied)
- Function: Transitions drones to selected pattern
- Animation: Smooth movement over 3-5 seconds
- Note: Drones become stationary in formations (except Random Waypoint)
- Use while paused: Formation applied, resume to see settled state
- Use while running: Observe transient disruption to network

2.2 Tech Profiles

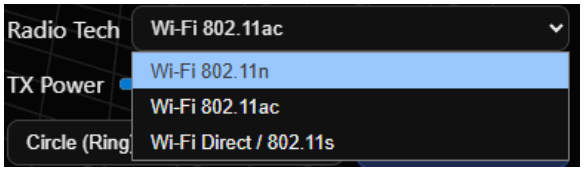


Figure 5: Wi-Fi Tech Profiles

Wi-Fi Technology Selector:

wifiAC (802.11ac):
- Frequency: 5.0 GHz
- Data Rate: 866 Mbps (peak)
- TX Power: 150 mW
- Effective Range: 25 meters
- Receiver Sensitivity: -82 dBm
- Best for: Maximum throughput
- Drawback: Highest power consumption, shorter range than 2.4 GHz

wifiN (802.11n):
- Frequency: 2.4 GHz
- Data Rate: 150 Mbps (peak)
- TX Power: 100 mW
- Effective Range: 35 meters
- Receiver Sensitivity: -85 dBm
- Best for: Balanced performance and efficiency
- Note: Most common UAV profile

wifiDirect (Wi-Fi Direct):
- Frequency: 2.4 GHz
- Data Rate: 250 Mbps (peak)
- TX Power: 80 mW
- Effective Range: 30 meters
- Receiver Sensitivity: -88 dBm
- Best for: Energy-constrained scenarios, short range
- Use case: Dense swarms with limited battery

Changing Tech Profiles:
- Requires: Simulation reset (clears existing network state)
- Effect: All drones use same technology
- Note: Cannot mix technologies in single simulation

Figure 6: TX Power slide located on Control Panel

TX Power Slider: (Note: not entirely functional and adjustment may not affect simulation behavior)
- Range: 10% to 100%
- Default: 100% (maximum power)
- Effect: Scales transmit power and range proportionally
- Examples:
  * 100%: wifiAC @ 150 mW, 35m range
  * 60%: wifiAC @ 90 mW, ~26m range
  * 30%: wifiAC @ 45 mW, ~18m range
- Trade-off: Range vs. battery lifetime
- Use 30-50%: For energy studies, dense networks
- Use 100%: For sparse networks, maximum connectivity

Routing Protocol:

OLSR (Optimized Link State Routing):
- Type: Proactive (table-driven)
- Behavior: Maintains routes to all nodes continuously
- Messages:
  * HELLO: Every 2 seconds (neighbor discovery)
  * TC (Topology Control): Every 5 seconds (topology broadcast)
- Overhead: Higher (continuous updates)
- Latency: Lower (routes pre-computed)
- Best for: Static or slow-moving networks

2.3 Information Panels (Right Side)

Neighbors Panel (MAC Layer Adjacency):

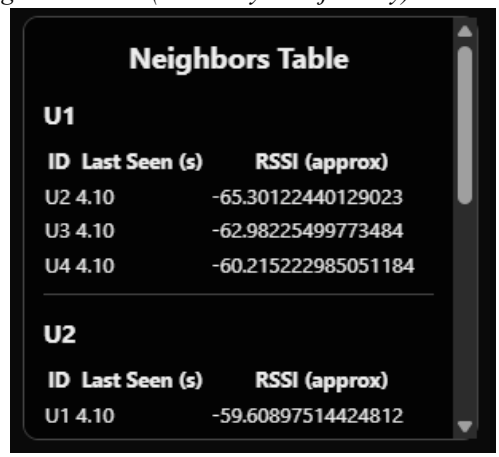


Figure 7: The Neighbors Table is located in the top right corner of the GUI

Purpose: Shows one-hop wireless links for selected drone

Columns:
- Neighbor ID: Drone identifier (0 to N-1)
- RSSI (dBm): Received Signal Strength Indicator
  * Range: -40 dBm (very strong) to -90 dBm (very weak)
  * Typical: -50 to -70 dBm for good links
  * Below -80 dBm: Unreliable, frequent drops

Selecting a Drone:

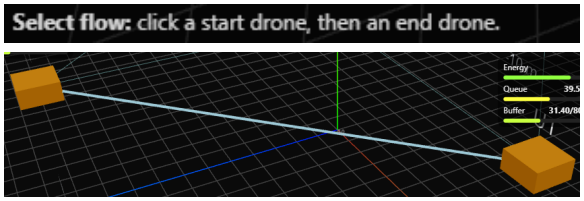


Figure 8: After selecting two drones, the connection between them will illuminate

- Method: Click drone in 3D scene
- Visual feedback: Outline or highlight
- Panel updates immediately with that drone's neighbors

Empty table means:
- No routes discovered yet (wait 10-20 sec)
- Network partitioned (increase TX power or density)
- Routing protocol not converged

ACK Statistics Panel:

Node	txQ	retry	ACK-Rcv	ACK-Sent	Collision
U1	1	0	8	7	0
U2	1	0	4	7	0
U3	1	0	7	6	0

Figure 9: The ACK Statistics panel, located at the bottom of the window

Shows number of acknowledgments sent
Retry and received by each neighboring drone

Purpose: MAC layer acknowledgment tracking
Helps verify correct basic CSMA/CA behaviour
By confirming successful transmissions,
Acknowledgments and retries in busy channels.

Metrics:
- Total TX: Packets transmitted by selected drone
- ACKs Received: Confirmations from receivers
- Timeouts: Packets not acknowledged within 50ms
- Success Rate: (ACKs / Total TX) × 100%
  * Target: >90%
  * <80%: Indicates collision or interference issues

* <50%: Severe congestion or sparse network

Use cases:
- Diagnosing MAC layer performance
- Identifying congested nodes
- Validating collision avoidance

Mobility Event Log:

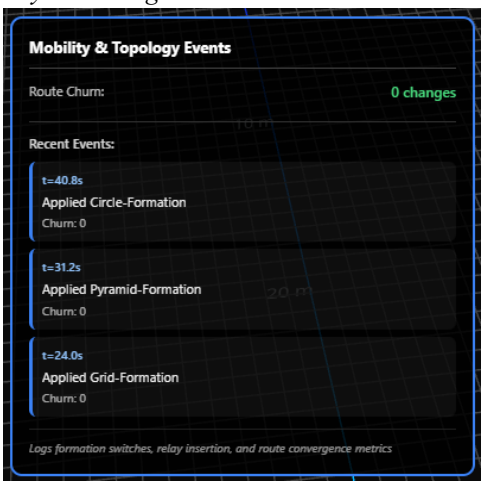


Figure 10: The Mobility & Topology Events information is available via the Mobility Events button at the bottom of the window

Purpose: Timestamped record of topology changes

Event Types:
- Formation Change: "Applied V-Formation at t=45.2s"
- Route Update: "Node 5: Routing table updated (8 changes)"
- Link Break: "Link lost: Node 3 → Node 12"
- Neighbor Timeout: "Node 7: Neighbor 15 timed out"
- Energy Critical: "Node 22: Battery below 20%"

Features:
- Auto-scroll: Latest events at bottom
- Timestamps: Simulation time in seconds
- Filter: (Future feature) Show only specific event types

2.4 KPI Charts and Data Export

Data Export Options:

Figure 11: Export option buttons located at the top left of the display

[Export CSV] Button:
- Format: Comma-separated values
- Columns: time, pdr, latency_ms, avg_queue, avg_energy, hop_count
- Sample rate: 1 Hz (one row per second)
- Example row: 10.5,0.89,18.3,5.2,97.4,1.8
- Use for: Excel, Python pandas, MATLAB analysis

[Export PNG] Button:
- Captures: Current state of all four KPI charts
- Resolution: 1920×1080 pixels
- Format: PNG image
- Use for: Reports, presentations, documentation

Live Time-Series Graphs:

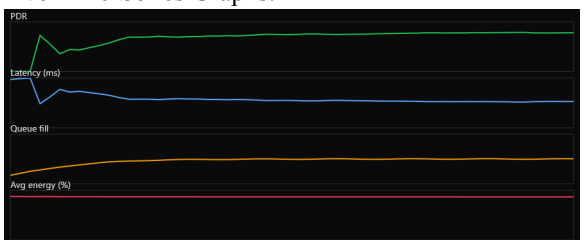


Figure 12: Output results via the PNG

PDR Chart (Packet Delivery Ratio):
- Y-axis: 0.0 to 1.0 (0% to 100%)
- X-axis: Simulation time (seconds)
- Update: Every 1 second
- Calculation: (Delivered packets in last 1s) / (Sent packets in last 1s)
- Color: Green (>0.9), Yellow (0.7-0.9), Red (<0.7)

Latency Chart:
- Y-axis: 0 to 100 ms
- X-axis: Simulation time (seconds)
- Metric: Mean end-to-end delay of delivered packets
- Includes: Queuing + transmission + propagation delays
- Spikes indicate: Route changes, congestion, or retransmissions

Queue Fill Chart:
- Y-axis: 0 to queue capacity (default 50 packets)
- X-axis: Simulation time (seconds)

- Metric: Average queue occupancy across all drones
- Zones:
  * 0-20 (Green): Healthy, low congestion
  * 20-40 (Yellow): Moderate load
  * 40-50 (Red): Near capacity, drops imminent

Average Energy Chart:
- Y-axis: 0% to 100%
- X-axis: Simulation time (seconds)
- Metric: Mean battery level across all drones
- Slope: Steeper for high TX power, fast mobility

2.5 3D Scene Interaction

Drone Visualization:

Figure 12: Information provided about each drone

Appearance:
- Model: Stylized quadcopter (four rotors)
- Label: ID number displayed above (e.g., "U5", "U12")
- Size: Proportional scaling (consistent across all drones)

Color Coding (MAC State):
- Green: Idle (radio on, listening, not transmitting)
- Yellow: Carrier Sensing (detecting channel activity)
- Red: Transmitting (actively sending packet)
- Blue: Receiving (packet being received)
- Note: Colors flash rapidly during active communication

  * Queue: Packets in transmit queue (0-50)
  * Buffer: Packets waiting for routes (0-100)
- Position: Fixed offset above drone

Selection:
- Method: Click on drone
- Effect: Right panels update to show that drone's data
- Visual: Outline or glow effect on selected drone

Links Visualization:

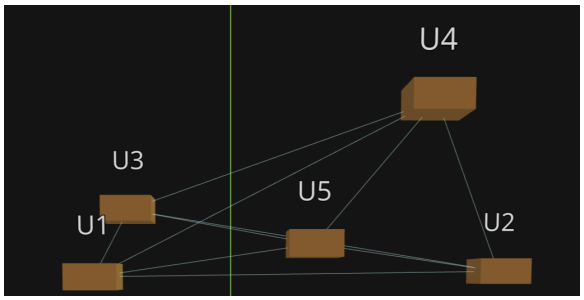


Figure 13: Drones linked together in physical space

Appearance:
- Color: Cyan (light blue)
- Width: 1-2 pixels
- Style: Solid line
- Range: Only shown for distances ≤ effective range

Link Meaning:
- Represents: MAC-layer bidirectional communication link
- Criteria: RSSI > sensitivity threshold, SNR > 5 dB
- Dynamic: Appear/disappear as drones move
- Note: Link presence doesn't guarantee packet delivery

Ground Grid and Environment:

Grid:
- Type: Cartesian coordinate system
- Spacing: 10 meter intervals
- Axes: X (red), Y (green), Z (blue)
- Labels: Numeric markers every 10m

Camera Controls:

Orbit (Rotate View):
- Method: Left-click + drag
- Effect: Rotates camera around center point
- Speed: Proportional to drag distance
- Constraint: Cannot flip upside down

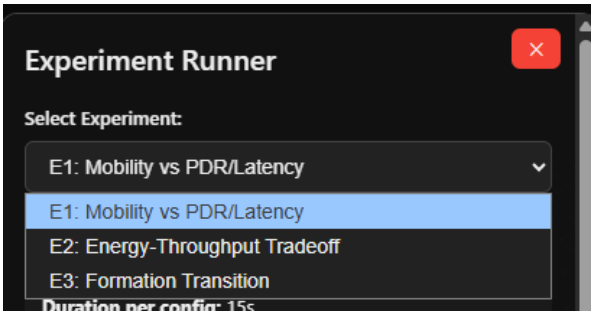
Pan (Translate View):
- Method: Right-click + drag
- Effect: Moves camera and target together
- Use: Recentering view on drones

Zoom:
- Method: Mouse scroll wheel
- Effect: Moves camera closer/farther from target

3. EXPERIMENTS & CONFIGURATION

3.1 Three Core Experiments

Figure 14 (right): After selecting "Experiments" in the top left, the captured drop down will appear, allowing you to select the experiment of your choice.



E1: Mobility vs PDR/Latency - Formation vs random waypoint at N=5, 25, 100
E2: Energy - Throughput Tradeoff - TX power 20%/40%/60%/80%/100%, measure lifetime vs PDR
E3: Formation Transition - 3 phases: waypoint(60s)→transition(30s)→hold(60s)

3.2 Key Metrics

Metric	Target	Description
PDR	>0.90	Packet delivery ratio (delivered/sent)
Latency	<50 ms	End-to-end delay
Queue	Minimize	Packets to be processed
Hop Count	Minimize	Average intermediate nodes
Energy/Bit	Minimize	Battery consumption per bit
Route Churn	<5/s	Routing table changes per second

4. TROUBLESHOOTING & KEY INFORMATION

4.1 Common Issues

Low FPS: Reduce drones (N=25), lower speed (0.5×), close other tabs
No Movement: Check Play pressed, speed>0, formations are stationary
No Routes: Wait 20s for convergence, check TX power >30%, increase density
Low PDR: Reduce speed (2-5 m/s), increase TX power (60-100%), add drones
Crashes: Don't exceed 100 drones, clear cache, restart server (Ctrl+C)

4.2 Console Debugging (F12)

```
nodes[0]  // inspect drone
nodes[0].networkLayer.getAllRoutes()  // view routes
nodes[0].neighborTable  // check neighbors
nodes[0].networkLayer.getStats()  // get metrics
```

4.3 Network Architecture

Application → Traffic generated every 8-12s, 400 bytes
Network → OLSR routing, node IDs (not IPs), TTL=64 hops
MAC → CSMA/CA, ACKs, queues (50 pkts), exponential backoff (16-1024)
PHY → Wi-Fi profiles, RSSI, path loss, energy model

4.4 Packet Structure

```
{
 packetId: "a3f2b1c4",   sourceId: 5,        destId: 12,
 packetType: "DATA",     ttl: 64,           hopCount: 0,
 timestamp: 1702345678901, sequenceNum: 42,    sizeBytes: 400
}
```

4.5 Key Formulas

Path Loss: $PL(d) = 40 + 10*n*\log10(d) + X_\sigma$ (n=2.0-3.2, σ=2-6dB) [1]

RSSI: $P_{tx}(dBm) - PL(d)$ [2]

SNR: RSSI - NoiseFloor(-95dBm) [3]

Energy: $E(mWh) = \Sigma(P_{state} * t_{state})$, Battery=5000mWh [4]

Power States: TX=80-150mW, RX=50mW, Idle=10mW, Sense=15mW, Sleep=1mW

4.6 Default Parameters

Range: 25-35m | Beacon: 1s | HELLO: 2s | TC: 5s | Neighbor timeout: 3s
Queue: 50 pkts | Buffer: 100 pkts |

Traffic: 1 pkt/8-12s | Pkt size: 400 bytes

4.7 Essential File Information

Scene3D.jsx - Core simulation, physics, node spawning
trafficGenerator.js - Application data generation
networkLayer.js - Network layer interface
routing.js - OLSR implementations
macChannelAccess.js - CSMA/CA MAC protocol
queueManager.js - Packet queuing
neighborTable.js - RSSI, SNR, MCS calculations
phyEnergyModel.js - Power consumption tracking

4.8 Developer Tools

```
sceneControls.spawnDrones(count)
sceneControls.applyFormation(type)  // 'v','line','grid','circle','pyramid'
sceneControls.setSpeed(mps)         // 0-15
sceneControls.setTxPower(percent)   // 0.1-1.0

nodes[i].networkLayer.sendPacket(destId, payload)
nodes[i].networkLayer.getRoutingTable()
nodes[i].networkLayer.getStats()

import { getSamples, addSample, clearSamples } from './hooks/kpiStore'
```

More Information:

Document Version: 1.0 | December 2025
Support: See README.md, SYSTEM_OVERVIEW.md, EXPERIMENT_GUIDE.md
Contributing: Follow ESLint rules (npm run lint), test with npm run dev

5. References

References:

[1] “Log-distance path loss model,” Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Log-distance_path_loss_model

[2] “Received signal strength indicator,” Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Received_signal_strength_indicator

[3] The Things Network, “RSSI and SNR,” TheThingsNetwork Documentation, 2024. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/rssi-and-snr/

[4] D. Minoli, “Modeling Overall Energy Consumption in Wireless Sensor Networks,” arXiv preprint arXiv:1112.5800, 2011. [Online]. Available: https://arxiv.org/pdf/1112.5800

6. Statement of Work

5.1 Aidan Kiswoto

Focused mainly on the GUI on the simulation. I worked with the Physical, Link/MAC, Network and Mobility teams to display specific factors using buttons such as Traffic statistics, Acknowledgements and Mobility Events. I also worked on modelling the specific aspect ratio of each tab as well as making each tab scrollable for a better user-friendly experience. I worked closely with John to create the best possible GUI that worked with each team.

5.2 Alberto Serrano

Mobility: worked and tested how the drones took 2D and 3D formations, created equations that would help drones calculate how far away they are from their formation position, created equations that would complete formation logging such as steady state and churns, and was a part of the live presentation.

5.3 Guadalupe Zarate

I worked on the Link/MAC layer of the drone network simulation alongside Kevin Lee, contributing to the design and implementation of core CSMA/CA mechanisms, including acknowledgements, collision handling, and neighbor table generation. Using models from the physical and network layers, we helped develop and refine algorithms for channel sensing, packet transmission, and ACK-based reliability. Much of my effort focused on validating the accuracy of the simplified models so that the simulation behavior felt consistent with real-world CSMA/CA systems. In addition to the core technical work, I created a small set of testing drones for visual and exploratory purposes, adding an intuitive and engaging way to observe system behavior, even though these visuals were not central to the protocol design. I also assisted with testing multiple simulation runs, reviewing results, and presentation materials to clearly communicate our work and findings. I am sincerely grateful to all of my project team members for their collaboration and support throughout this project. I would especially like to thank Kevin Lee for his close collaboration within our CSMA/CA subgroup, whose insights and teamwork were invaluable during the development and validation of the protocol. I am also very thankful to Nolan Conrad and John Guerrero, who designed and implemented the network layer and GUI, playing a central role in bringing the project together.

5.4 John Guerrero

GUI Implementation. I ensured GUI properly reflects layers behaviors and analytics of each layer incorporated in the simulation. I contributed to the energy model, tech profiles, buffer, queue, ensuring behaviors are consistent throughout simulation. Collaborating with the PHY team to create an energy model that would make sense and mimic real-time energy depreciation as much as possible. Collaborated with the Link/MAC team to ensure that all the implementation of CSMA/CA mechanisms, ACK, etc reflected on the GUI properly. Collaborated with the Mobility team to create a movement model that covers obstacle interaction (partially complete), formation changes, and creation of a 3D space for topology. Contribution to presentation slides and Q&A preparation.

5.5 Nolan Conrad

I specialized in the Network Layer for our project, and focused on designing and implementing the OLSR routing protocol, along with Rhilo Sotto, to ensure efficient and reliable communication between nodes in the system. In addition to my networking responsibilities, I contributed to the general effort across multiple components of the project, helping integrate features and resolve a multitude of issues that we faced. I also played a key role in overseeing the review process for our early designs, ensuring that we were in functional shape, and aligning with project objectives. Beyond technical development, I assisted in creating and refining the documentation for both our in-class presentation and our final user manual, helping translate complex system behaviors into clear and accessible explanations for the end users.

5.6 Reem Awad

I developed and maintained the physical layer code for the project, including a Wi Fi based channel model with distance based packet loss, interference and noise events, and capture logic. I implemented logging and experiment support to track RSSI like metrics, packet success and loss, and throughput.

5.7 Rhilo Sotto

Network layer. I worked on general simulator testing and ensuring rubric requirements were met. Focused on formatting and organizing the presentations and presenting for the final presentation. Formatted and proofread the user manual, and made the table of contents.

5.8 Wilson Cao

Physical layer, involved with implementing the physical layer with a focus on modeling battery drain and energy consumption. This work was integrated with the Link and MAC layers to support realistic communication behavior within the network. I took part in presenting the technical presentation summarizing the design, implementation, and results.

5.9 Kevin Lee

Link/MAC team. I worked on the CSMA/CA protocol and collisions/acknowledgements. I ensured the results showed decent performance and similar outputs compared to real world applications through testing and research. I worked on the slides and presented them to the class. Tested the experiments and gathered the results of multiple runthroughs to be put on the final presentation.

END OF MANUAL