Interpretable Multi-Agent Coordination Algorithms for Heterogeneous (Urban/Suburban/Rural) Autonomous Mobility Networks

Authors: Rivaldo Lumelino, Alexandr Voronovich

Our mission

Develop and analyze interpretable multi-agent coordination across different network environments

Urban · Suburban · Rural · V2X · MQTT

8 autonomous drones.

Each drone moves, sends messages, senses neighbors, and makes decisions

Network controls message delays, drops, and queueing

Drones generate human-readable explanations for every action

Motivation / Problemhat we do

Communication can be slow, delayed, or lost

Different environments have different challenges

- **Urban**: congestion / interference
- **Suburban**: moderate network quality
- **Rural**: weak or unreliable signals

Agents must still avoid collisions even when communication fails

Humans must understand why agents take actions

Key Problems:

How do agents communicate in poor networks?

How do they adapt to different environments?

How do we explain agent decisions in human language?

Our Approach

Communicate

- Share position & movement intention messages
- Use delay-based delivery + message drops
- Support real-world protocols: V2X and MQTT

Sense

- Each drone has a 5-meter sensor range
- If messages fail, drones fall back to sensor data

Explain (Interpretability)

- "Reduced speed because it had no communication."
- "Turned sharply to avoid another drone."
- "Using received position updates for navigation."

Environments

- We simulate 5 network types:
- Urban
- Suburban
- Rural
- V2X
- MQTT

Improved in the Code

1. Standardized Network Layers **(V2X + MQTT)**

We implemented two realistic communication models:

- **V2X**: near-real-time, almost no loss
- **MQTT**: IoT-style, queued and delayed delivery

Why it matters:

- Allows comparison of real-world communication protocols

2. Agent Interpretability Engine

We added human-readable logs:

- Message reception
- Slow-down decisions
- Sensor fallback
- Large-turn detection

These logs make drone behavior understandable.

Improved in the Code

3. Sensor-Based Safety Backup

If communication fails:

- Drone slows down by 80%
- Uses sensor range (5m) to detect nearby drones
- Avoids collisions without messages

This simulates "local autonomy" under network failure.

4. Message Delay + Queue Simulation

Message queue behaves like real network congestion:

- Delayed deliveries
- Messages dropped based on probability
- Queue growth measured

5. Collision Detection & Metrics Architecture

We built:

- Collision counter
- Message success rate
- Queue size logging
- Baseline for adding more metrics later

Network Type	Message Loss (%)	Delay Range (steps)	Speed of Communication	Reliability	Expected Behavior of Drones
V2X	1%	1-1	Very Fast	⭐Very High	Smooth coordination, almost no fallback to sensors
MQTT	10%	3-8	Slow	Medium	Messages pile up in queue, drones often rely on delays
Urban	5%	1-3	Fast	High	Occasional misses but overall good coordination
Suburban	10%	2-6	Medium	Medium	Slower updates,occasional confusion
Rural	30%	5-15	Very Slow	Low	Many drops, drones mostly use sensors + safe mode

Thank you