URBAN DELIVERY MAS

DECENTRALIZED MULTI-AGENT COORDINATION FOR DYNAMIC, PRIORITY-AWARE LAST-MILE LOGISTICS

PROFESSOR : ANDREA OMICINI

STUDENT : ARVIN LAJANI

UNIVERSITY OF BOLOGNA CAMPUS CESENA

Problem Statement

- **Urban Last-Mile Logistics Challenges**
- **Dynamic Environment Perturbations:**
- Dynamically arriving customer orders
- Shifting priority tiers (Standard vs. Express)
- Fluctuating traffic congestion
- Battery charge degradation
- Hardware failures

Centralized Optimization Limitations:

problems	impact
Single Point of Failure	Server crash paralyzes entire fleet
Computational Inflexibility	Any perturbation forces full re-optimization
Telemetry Overhead	Continuous high-frequency data streaming
Local Disturbance Rigidity	Can't accommodate reactive self-preservation

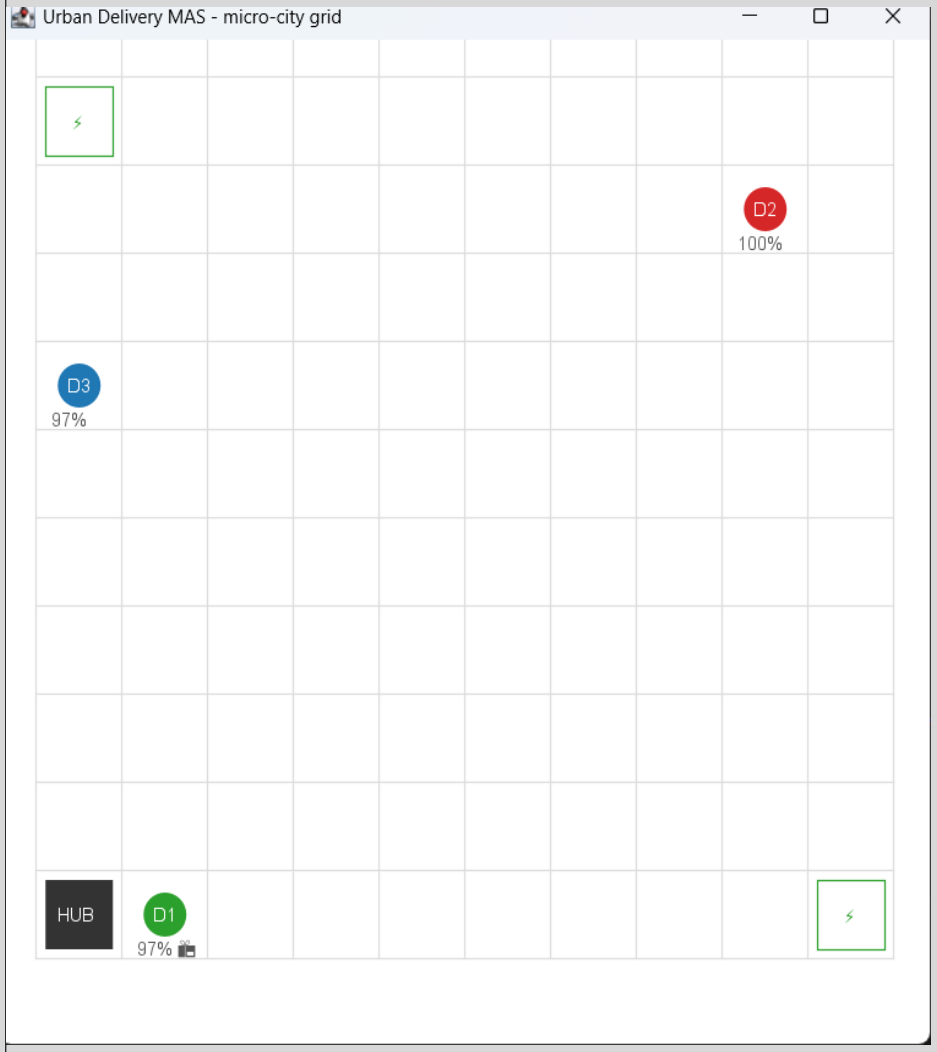
System Architecture

Three-Layer Decoupled Architecture
VIEW LAYER (Swing GUI) Minimalist 2D visualization of grid
COGNITIVE REASONING LAYER Jason 3.3.0 / AgentSpeak(L) Agents: BDI States: Beliefs, Desires, Intentions
ENVIRONMENT PHYSICS LAYER Java 21 Environment API: Grid actions, state transitions, percepts

Component Responsibilities

Component	Instances	Responsibility
Distribution Hub (hub.asl)	1	Broadcasts CFPs, collects bids, awards tasks, manages re-delegation. Zero global control over drones.
Drone Agent (drone.asl)	3 (drone1-3)	Computes context-sensitive bids, navigates grid, manages battery cycles, handles preemption, drops intentions when critical.
DeliveryEnv (Java)	1	Maintains 10×10 grid, executes atomic actions, publishes percepts, injects disturbances
DeliveryGUI (Java)	1	Real-time Swing visualization of positions, paths, battery indicators.

Theoretical Background



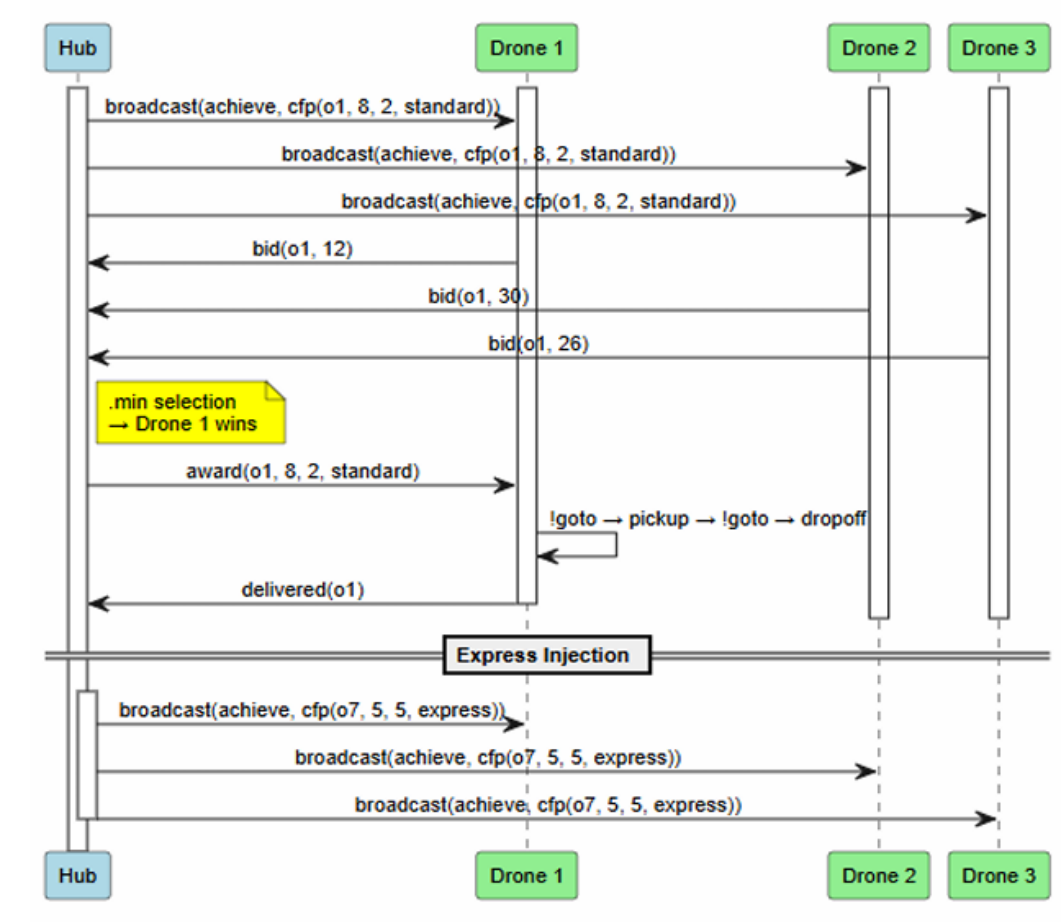
BDI AGENT

BELIEFS (B)
Informational state
pos(X,Y), battery(B), carrying(Id)

DESIRES (D)
- Motivational state
- Fulfill orders, maintain safe battery

INTENTIONS (I)
- Deliberative state
- Active committed plans: !deliver, !recharge

Asynchronous Task Allocation Protocol



Phase 1: Call for Proposals (CFP)

Manager broadcasts task details (e.g., location, priority) to the network.

Phase 2: Bid Submission

Contractor agents compute execution costs based on private state (proximity, battery, workload) and submit bids.

Phase 3: Bid Evaluation & Awarding

Manager evaluates bids using an objective function (e.g., cost minimization), awarding the task to the best candidate and rejecting others.

Phase 4: Task Execution

Winning contractor commits the task to its local intention structure and executes it.

Concurrency Control: Atomic Annotations

- *Ensuring transactional integrity during multi-agent task allocation.*
- **The Problem: Race Conditions in Auctions**
  - Multiple bid messages (bidrec) arrive asynchronously from different drones.
  - Without synchronization, concurrent belief updates or interleaved plan executions could lead to double task allocation or dropping valid bids mid-evaluation.
- **The Solution: @atomic Execution**
  - Annotation syntax: @try_award_plan[atomic]
  - **Critical Section:** Forces the Jason reasoning engine to execute the entire award plan as an **indivisible transaction**.
  - **Thread Safety:** Suppresses intention switching—no other event or plan can interleave until the winner selection and .abolish() memory cleanup complete.
- **System Benefit:** Guarantees deterministic, race-condition-free task allocation in a decentralized multi-threaded environment.

Key Architectural Decisions

Decision	Rationale
Achievement Goal Performatives	tell updates beliefs but may not trigger reasoning. achieve forces immediate deliberation
Atomic Plan Execution	Prevents race conditions during concurrent bid evaluation
Prohibitive Bidding (99999)	Ensures N responses per CFP → Clean .min() selection
Decoupled Environment	No cognitive logic in Java code → Fully transparent reasoning

Conclusion

- **Emergent Spatial Task Division**
  - Natural partitioning through localized bidding
  - No central trajectory optimization
- **Dynamic Fault Recovery**
  - Belief-triggered intention dropping
  - Safe re-delegation and self-preservation
- **Real-Time Priority Preemption**
  - Context-sensitive plan selection
  - Zero task abandonment