ACTS

Autonomous City Traffic Simulation

Decentralized Control

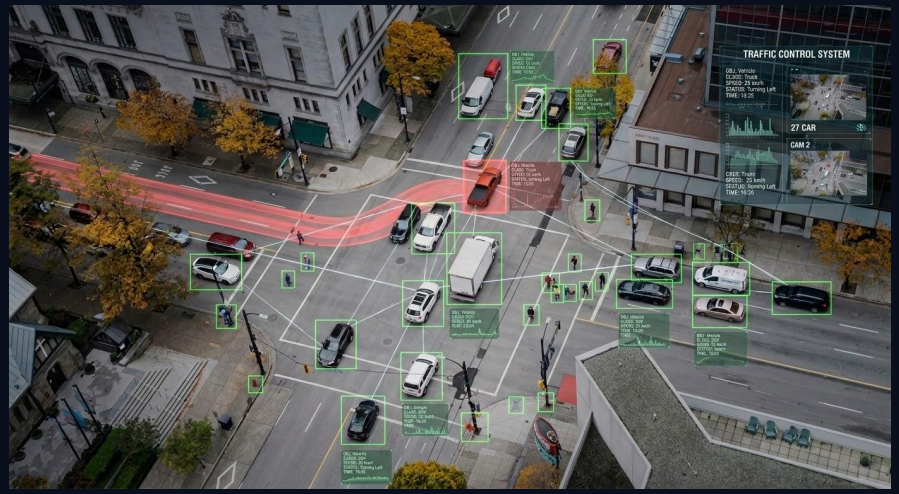
Built with Python (Mesa & NetworkX) to manage urban traffic without a central coordinator.

Consensus & Logic

Smart traffic lights negotiate green states dynamically using Lamport clocks and conflict matrices.

Testing Sandbox

An ideal simulation environment for validating complex distributed systems architectures.



Federico Morsucci · Cristian Morbidelli

Distributed Systems Course · A.Y. 2025–2026

ANALYSIS

The Context and the Problem

1. The Context

Classic urban traffic control relies on rigid timers or a central server, failing to adapt in real time to the flow of vehicles.

2. The Problem

A central server creates bottlenecks and a Single Point of Failure (total breakdown). Furthermore, fixed timers do not allocate green lights based on actual demand.

3. The Goal

Create a simulation where intersections are autonomous. Traffic lights act as independent IoT nodes, communicating via Redis without a central server.

INFRASTRUCTURE

Simulation and the Trade-off

Hybrid Architecture (Simulator + Network)

Spatial and temporal engine managed by **Python (Mesa & NetworkX)**, with JSON communications over an external **Redis** broker isolated on Docker.

The Simulation Trade-off

Time advancement coordinated centrally by Mesa (**step()**) for control and testing. The negotiation logic, however, is **100% distributed** and "Network-Ready".

Deployable on Microcontrollers

The negotiation code at each intersection does not share memory and the algorithm is ready for execution on physically distributed hardware.

INFRASTRUCTURE

Integration Architecture

Mesa Runtime Framework

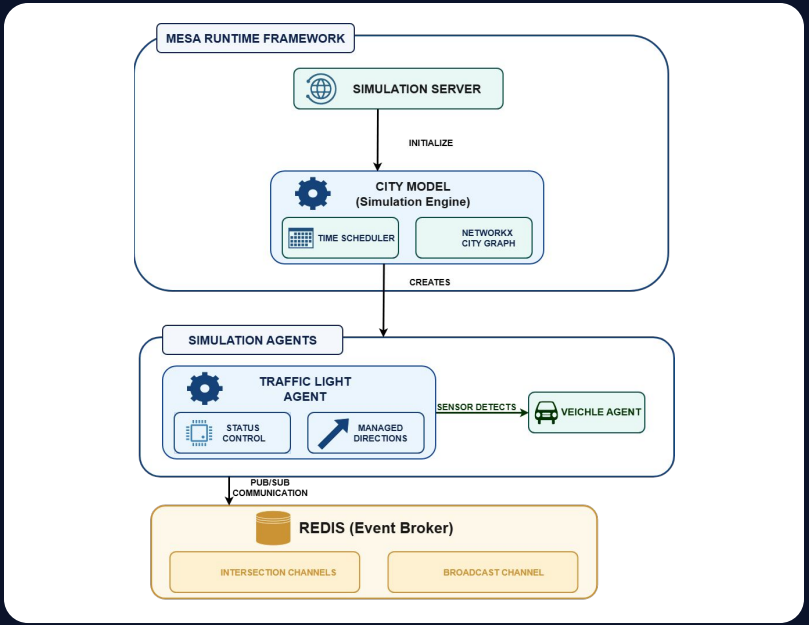
Core engine managing the simulation server, time schedulers, and the NetworkX city graph topology.

Simulation Agents

Autonomous vehicle agents and smart traffic lights acting as independent decision-making nodes.

Redis (Event Broker)

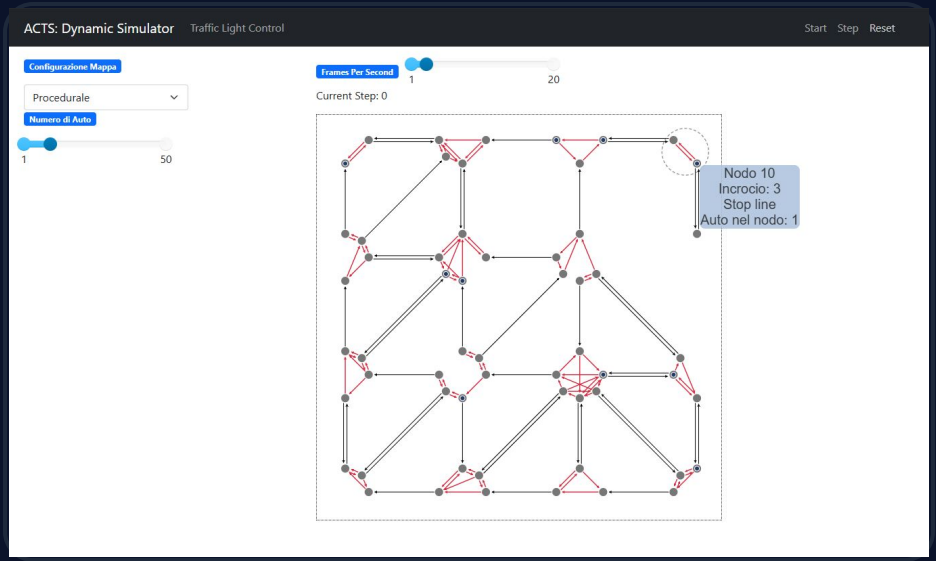
Decentralized message broker managing pub/sub communications across intersection and broadcast channels.



Federico Morsucci · Cristian Morbidelli

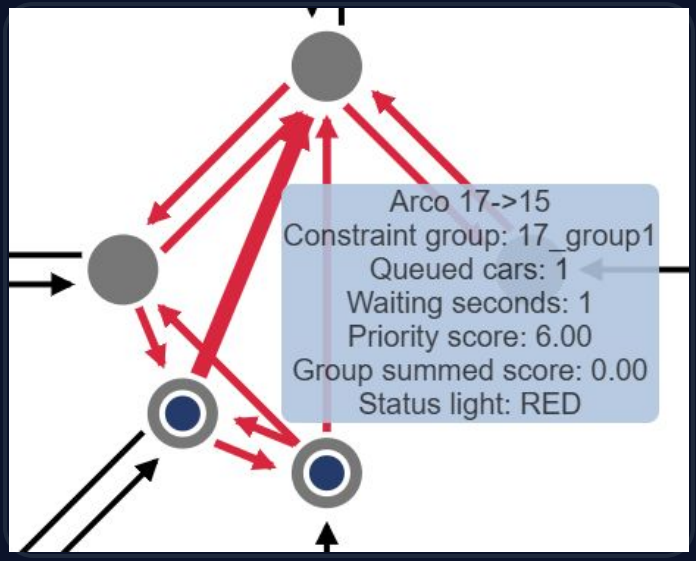
SIMULATOR

The Control Interface



Control Panel

Dynamic map configuration and global real-time traffic flow monitoring.



Node Details

Local analysis of each intersection: car queues, waiting times, and calculated passage priorities.

LOGIC

The Distributed State and Decision Mathematics

State Management

The system state is temporarily partitioned within each traffic light (**TrafficLightAgent**).

State Breakdown:

- **Local:** queues and waiting times
- **Distributed:** Lamport Clock, permission maps, and arrival estimates

The Dynamic Score

Priority is not based on a fixed timer, but is recalculated at each step using an optimized algorithm:

$$S = (Q \cdot 5 + W) + \left(\sum_{i=1}^{n} \frac{Wave_i \cdot \omega}{1 + \frac{ETA_i}{10}} \right)$$

Where **Q** is the queue, **W** the wait, **Wave** the incoming traffic flow, and **ETA** the estimated arrival time.

Federico Morsucci · Cristian Morbidelli

LOGIC

The Negotiation Protocol (Mutex)

No Supervisor

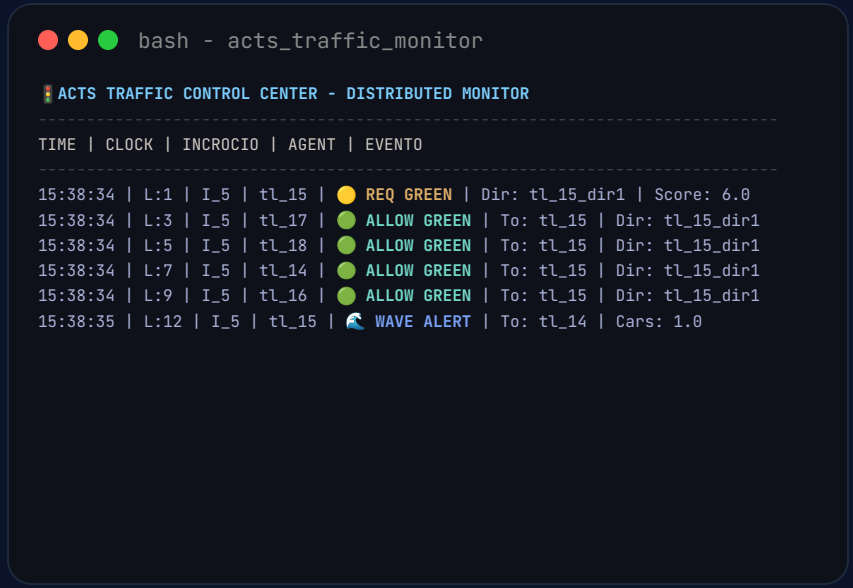
The network is decentralized. Traffic lights request green (**REQ GREEN**) from adjacent nodes based on a *Score* and a logical clock.

Network Consensus

Neighboring nodes grant permission (**ALLOW GREEN**) only if there are no hazards, verifying local conflict matrices.

Proactive Propagation

Traffic lights send downstream alerts (**WAVE ALERT**) to signal arriving cars, creating spontaneous and synchronized "green waves".



MONITOR

Real-Time System Logs Tracking

bash - acts traffic monitor

```
🚦 ACTS TRAFFIC CONTROL CENTER - DISTRIBUTED MONITOR
------------------------------------------------------------------------------------------------------------------------
TIME | CLOCK | INCROCIO | AGENT | EVENTO
------------------------------------------------------------------------------------------------------------------------
15:38:36 | L:21 | I_1 | tl_3 | 🟡 REQ GREEN | Dir: tl_3_dir0 | Score: 2.1 | ReqClock: 20
15:38:36 | L:23 | I_1 | tl_5 | 🟢 ALLOW GREEN | To: tl_3 | Dir: tl_3_dir0 | ReqClock: 20
15:38:38 | L:39 | I_1 | tl_4 | 🟡 REQ GREEN | Dir: tl_4_dir0 | Score: 9.0 | ReqClock: 38
15:38:39 | L:45 | I_1 | tl_4 | 🟡 REQ GREEN | Dir: tl_4_dir0 | Score: 12.0 | ReqClock: 44
15:38:40 | L:63 | I_1 | tl_5 | 🟢 ALLOW GREEN | To: tl_4 | Dir: tl_4_dir0 | ReqClock: 60
15:38:41 | L:121 | I_1 | tl_4 | 🟡 REQ GREEN | Dir: tl_4_dir0 | Score: 21.0 | ReqClock: 60
15:38:42 | L:134 | I_1 | tl_3 | 🟢 ALLOW GREEN | To: tl_4 | Dir: tl_4_dir0 | ReqClock: 60
15:38:42 | L:136 | I_1 | tl_4 | 🌊 WAVE ALERT | To: tl_3 | Cars: 1.0
```

PROPERTIES

Concurrency, Safety and Liveness

Tie-Breaking (Concurrency)

If two traffic lights request the green light at the same instant and with the same score, the conflict is resolved using **Lamport Clocks**: the one with the lower clock wins, preventing deadlock. In case of a further tie, the **alphanumeric ID** of the node is evaluated.

Safety (Physical Safety)

To guarantee the absence of accidents, revoking the green light forces a mandatory transition through yellow and a cooldown period where the intersection is completely red (**All-Red Phase**), ensuring that the area is physically clear.

Liveness (Starvation)

To prevent main roads from blocking secondary ones indefinitely, a **MAX_GREEN_TIME** is integrated. Once the limit expires, the agent forcibly yields the green light, guaranteeing a bounded wait (**Bounded wait**).

RELIABILITY

Fault Detection and Tolerance

Timeout Pattern (Heartbeat)

Being a controllerless system, anomaly detection is local. Each traffic light monitors the idle time (**quiet_time**) of its neighbors.

Health Check

Once a certain silence threshold is exceeded, the agent sends a **HEALTH_CHECK** query. If it does not receive an **ALIVE_SIGNAL** in response, it infers the failure.

Graceful Degradation

The failed node is mathematically isolated and the intersection enters emergency mode (**Failsafe**, flashing yellow), preventing infrastructure blockage.

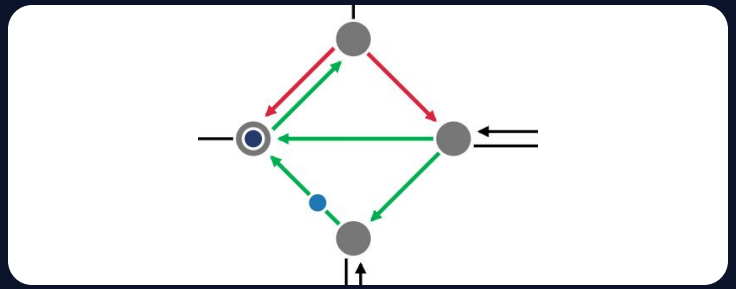
Signal Phasing and Independent Directions

Controlled Direction

The intersection is not a monolithic block. The traffic light manages separate objects called **Controlled Directions**, each with its own queues and waiting times.

Signal Phasing

The system implements phase control: trajectories sharing the same **phase_index** are not in conflict and can proceed simultaneously.



DEMO

Practical Demonstration and Scenarios

Basic Behaviors

Logical validation through preset simulations (demo) in the dashboard:

Arrival Order: Demonstrates asynchronous negotiation: whoever occupies the intersection first gets priority, alternating in complete safety.

Fairness: Prevents starvation caused by continuous flows; thanks to **MAX_GREEN_TIME** the agent forcibly yields the green light to secondary roads.

Synchronization and Conflicts

Robustness of the distributed system in critical traffic scenarios:

Green Wave (Traffic Wave): As vehicles pass, a **TRAFFIC_SIGNAL** (with ETA) is issued to preemptively activate subsequent green lights.

Lamport Conflict: Resolves simultaneous arrivals at the same tick by comparing **Lamport Clocks** and, in case of a tie, the alphanumeric node IDs.

Conclusions

IoT Decentralization

ACTS demonstrates in the field that a **decentralized IoT architecture** is capable of managing urban traffic by reacting in **real time**, eliminating the rigidity of pre-programmed timers and centralized server bottlenecks.

Field Validation

The environment proves to be a solid **testbed** for the application of **distributed paradigms** in complex urban scenarios.