



MedNet

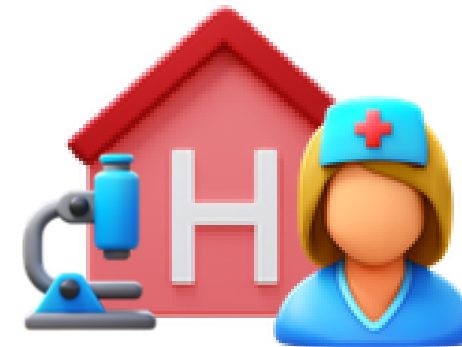
Multi-agent emergency system

Intelligent Systems Engineering course
Alma Mater Studiorum - University of Bologna
A.A. 2025-2026

Pier Costante Babini
Marco Morelli

The domain: Medical Emergency Network

- Control centre, fleet of ambulances, network of hospitals.
- Patient life cycle: call → triage → transport → care → discharge.
- World modelled as a 30×30 territorial grid.
- Each hospital perceives only its own state: beds, position, specialisations.
- Five scenarios isolate specific phenomena:
 - default (demo, 5 patients + walk-in burst)
 - e2e (complete life cycle)
 - preemption
 - divert (reroute)
 - severity (bid weights)



Why a multi-agent system?

- No agent knows the network's global capacity **in advance**.
- The **state changes** while a decision is being made (walk-ins).
- Priority **conflicts** over scarce resources (beds, equipment, doctors' time).
- A multi-agent system is essential to optimise resource use and improve real-time decision-making.

In a nutshell: the domain is a strong MAS case study, no information needed for a centralised decision lives in one place, and that information changes during the decision process itself.

Agents, roles and hospital network

Agent	No.	Responsibility
Patient	5	Reports a single emergency
Control center	1	Receives calls, dispatches ambulances, runs network CNP
Ambulance	3	Moves, performs on-site triage, transports
Hospital	3	Bids for admissions, reserves/releases beds
Triage nurse	3	Secondary triage, priority queue, runs in-hospital CNP
Doctor	6	Bids, derives and executes care plan, discharges
Equipment manager	3	Arbitrates shared equipment, runs watchdogs

Table 1: MedNet agents.

- Default configuration: 24 agents / 3 hospitals.
- Naming convention encodes role and placement (e.g. doctor_h3_trauma_surgery).

Goals and main contributions

- Contract Net Protocol reused across **two levels** of negotiation.
- **BDI agents** in Jason/AgentSpeak(L) integrated with a Java environment.
- Dynamic, decoupled discovery of agents/services via the **Directory Facilitator**.
- **FIPA ACL** communication.
- Shared **tuProlog knowledge base** (clinical & organisational knowledge).
- **STRIPS planner** for dynamic care plans.
- Cooperative **preemption** and replanning.
- **Renegotiation** during transport.
- **Mutual exclusion** via leases and watchdogs.
- **RBAC** enforced on every action.
- Automated 3-level test suite with Continuous integration on GitHub Actions.

Three-level architecture

- **BDI agents (Jason/AgentSpeak)**: all decision logic, when to bid, refuse, interrupt.
- **Java environment**: everything physical, grid, beds, equipment, clock, scenarios.
- **Prolog theories**: shared declarative knowledge, queried by both other levels.
- The **STRIPS planner** derives care plans that are dynamically created.

BDI in brief:

- **Beliefs**: what the agent holds to be true of the world and of itself.
- **Goals (desires)**: the states of affairs the agent would like to bring about.
- **Intentions**: the courses of action the agent has committed to and is actually executing.

Three-level architecture

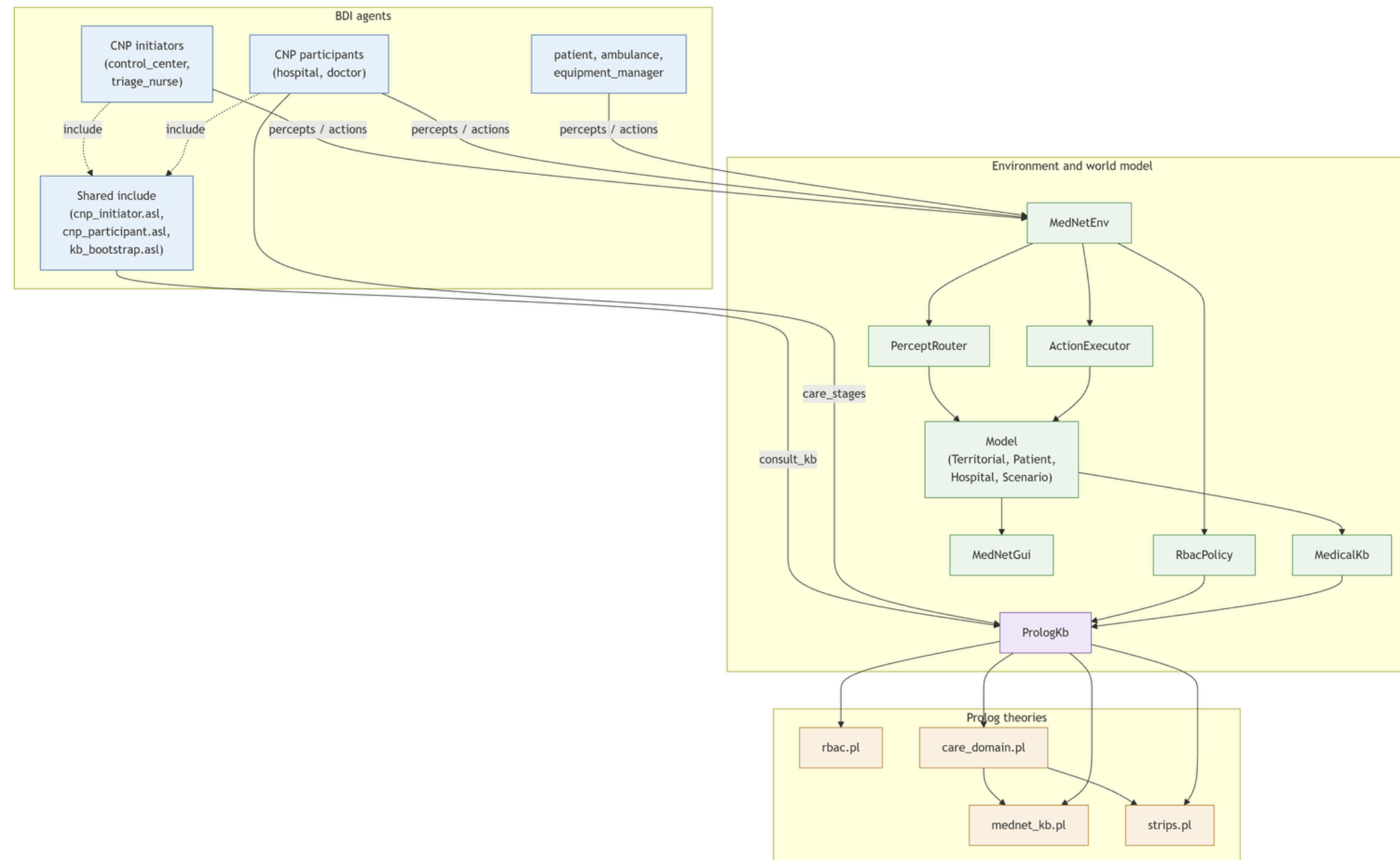


Figure 1: MedNet architecture.

The single gateway: Percept/Action + RBAC

- **PerceptRouter**: builds the percepts an agent is entitled to see, based on role and position → situatedness.
- **ActionExecutor**: first authorises (RBAC), then validates arguments/target, then applies the effect.
- Immediate re-perception after a successful action, no waiting for the next tick.

In brief:

- **Percepts**: literals the environment makes observable to a specific agent.
- **Actions**: the only way an agent can change the world (17 actions).
- **Messages**: asynchronous FIPA ACL exchanges between agents.

The bridge to Prolog

- Prolog engine based on **tuProlog**.
- **Four theories:**
 - clinical/organisational knowledge
 - RBAC policy
 - STRIPS planner
 - care domain.
- The tuProlog engine performs **inference**, providing a centralised knowledge base for every agent decision.
- The Java side queries default codes and clinical constraints, ensuring decisions rest on up-to-date knowledge.

The journey of a patient (end-to-end)

1. **Activation**: patient reports the emergency.
2. **Dispatch**: control centre assigns the nearest free ambulance.
3. **On-site triage**: preliminary assessment, patient loaded.
4. **Network negotiation**: CNP among hospitals.
5. **Transport & delivery**: reservation becomes occupancy.
6. **Secondary triage**: true severity code revealed.
7. **In-hospital negotiation**: CNP among doctors.
8. **Care**: plan derived and executed stage by stage.
9. **Discharge**.

The journey of a patient (end-to-end)

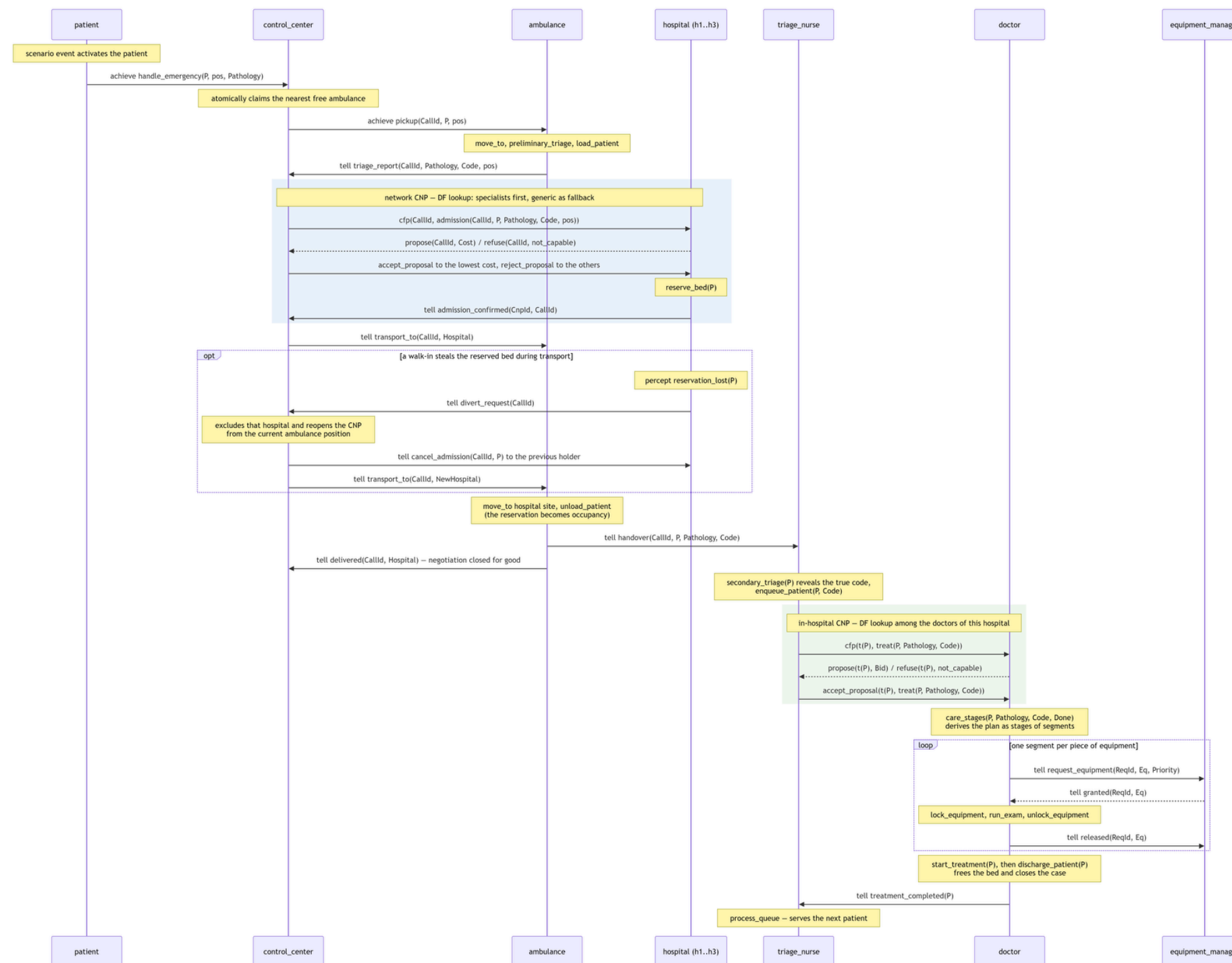


Figure 2: the journey of a patient.

Preview

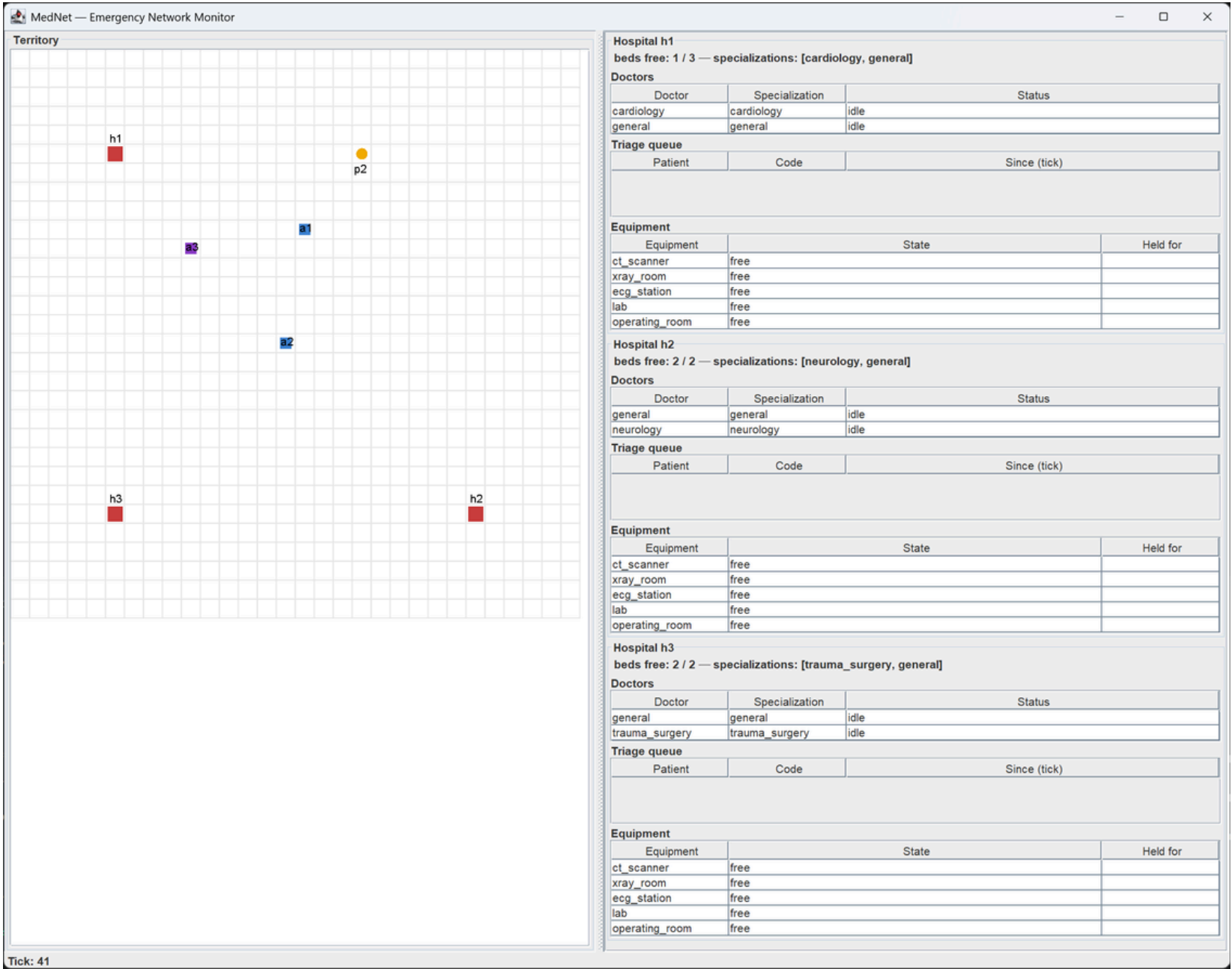


Figure 3: Program overview.

Contract Net Protocol: two levels

- **Common library** code (cnp_initiator.asl / cnp_participant.asl) used for both network and hospital negotiation.
- Five **FIPA acts**: cfp → propose / refuse → accept_proposal / reject_proposal.
- One round per identifier at a time; cleanup always runs before the callback.
- **Initiators**: control_center (network level), triage_nurse (hospital level).
- **Participants**: hospital agents (network level), doctor agents (hospital level).

The bid formulas (Inter and Intra Hospital)

Hospital (network) level:

$$Bid = \underbrace{(|H_x - x| + |H_y - y|)}_{\text{travel time}} \cdot W_d(\text{Code}) + \underbrace{\frac{(Cap - Free) \cdot 100}{Cap}}_{\text{ward load}} + \underbrace{P(\text{Code})}_{\text{specialisation mismatch}}$$

Code	W_d	P
red	30	2000
yellow	20	800
green	10	300
white	10	300

Figure 4: Inter-hospital bid formula, weights of the admission bid.

- **Doctor level (hospital):** bid = 0 if specialisation matches, 10 otherwise; availability is a hard constraint, not a cost.
- **No free beds:** the context fails and the library automatically refuses on the hospital's behalf (no fallback bid).

Dynamic behaviour I: preemption

- Decision taken only after a genuine "no winner" outcome.
- **Victim**: the least critical patient among those the protocol allows to preempt (red is never a victim).
- Cooperative **release of equipment**, nothing is torn away mid-exam.
- Preemption is a decisive strategy where a less critical patient can be **temporarily suspended** to handle a more urgent case.
- Equipment and staff are reassigned during preemption, ensuring optimal and reactive resource use.
- Preempted patients re-enter the queue, guaranteeing fair, severity-based handling.

Dynamic behaviour II: renegotiation

- A **walk-in** can steal a reserved bed.
- The hospital requests a **divert**; the centre reopens the negotiation from the ambulance's current position.
- The old bed must be released back before the ambulance is rerouted.
- **Delivery closes the negotiation** definitively.
- Renegotiation is triggered when external events affect bookings, ensuring effective management of resources and patients.

Dynamic behaviour III: mutual exclusion

- **Physical level:** single-owner lock (reentrant for the owner).
- **Cognitive level:** arbiter agent with a priority queue (min over priority, request time).
- An exam starts only once the physical lock is held, the grant alone is not enough.
- **No deadlock** by construction: no segment ever holds two pieces of equipment while waiting for a second.
- **Two watchdogs:** grant expiry and lease expiry.
- Mutual exclusion ensures only one agent can access shared resources at a time, preventing conflicts and guaranteeing efficient use.
- The system's design rules out deadlock, so no equipment remains permanently locked and unavailable.

Planning: Prolog + STRIPS

- **Declarative action schemas:** action(Name, Preconditions, Add, Delete).
- The doctor declares only the initial state and the goal (discharge), the plan is derived.
- **Parallel stages** (independent equipment segments) emerge naturally from the domain, not hard-coded.
- The STRIPS planner allows care plans to be derived and **dynamically updated** as the situation evolves.

Replanning after preemption

- Completed exams **survive the cancellation** of interrupted jobs (durable exam_completed/2 percept).
- The regenerated plan is strictly shorter than the original.
- No dedicated "resume" code, replanning is simply the planner given a **different initial state**.
- Completed exams are preserved, enabling optimal replanning with fewer exams left to perform.

Security: RBAC

- **Authorisation policy** declared in Prolog, not in Java.
- 7 roles × 17 actions, checked by a single question before every action.
- **Fail-closed** by construction: no explicit rule authorising an action → access denied.
- RBAC authorises the role, not the object (target validity is checked separately by the executor).
- Every denial is logged, and end-to-end tests assert that denials are absent in a correct run.
- The authorisation policy is declared in Prolog to guarantee secure, controlled access.

```
31     permitted(ambulance, move_to).  
32     permitted(ambulance, preliminary_triage).  
33     permitted(ambulance, load_patient).  
34     permitted(ambulance, unload_patient).
```

Figure 5: Ambulance permitted actions in Prolog. [Source](#).

Technology stack

Component	Version	Role
Java	21	Language of the environment and of the world model
Gradle	9.7.1	Build, MAS execution, separation of the test tasks
Jason	3.3.0	AgentSpeak(L) interpreter, messaging infrastructure, DF
tuProlog	4.1.1	Embedded inference engine for the four theories
JUnit	6.1.3	Test framework
AssertJ	3.27.7	Fluent assertions

Figure 6: MedNet technology stack.

Repository structure mirrors the three architectural levels (agents / Java / Prolog).

Testing on three levels + CI/CD

- **Unit:** world model, theories, planner, RBAC matrix, driven by manualClock.
- **Protocol:** real MAS with mock agents, isolates negotiation logic from physics.
- **End-to-end:** liveness, correct ordering, and absence of safety-net interventions.
- **GitHub Actions** CI on every push + **Conventional Commits** enforced via Husky/commitlint.
- Our approach combines unit, protocol and end-to-end testing, ensuring the system's reliability and effectiveness.



Conclusions and future work

The project provided an opportunity to gain hands-on experience with the course topics and practical technologies by implementing a multi-agent emergency system.

- Two-tier resource allocation (hospitals and doctors) both solved by negotiation, with the same protocol.
- Preemption, renegotiation and mutual exclusion as direct answers to the constraints of the domain.
- **Future developments:**
 - Obstacle-aware movement
 - Externally provided scenarios
 - An optimal planner
 - Stochastic, distribution-driven walk-ins

The background features four large, thick blue arcs positioned in the corners of the slide, creating a modern, abstract frame. The arcs are part of larger circles that extend beyond the visible edges of the slide.

Thank you for your attention