Connected Mobility SS25 - Assignment 1 - FMI Emergency Scenario - Zombie Apocalypse

Kilian Mio
Bisma Baubeau
Neel Henry Mandal

Abstract

This project models an emergency scenario within the FMI building using a simulated zombie apocalypse, where connectivity between nodes is established through physical contact. The primary objective is to analyze congestion points within the building and examine the dynamics of transmission and movement. Although the ONE simulator was initially designed for opportunistic network simulations rather than dynamic crowd scenarios, we adapted and extended it to suit our needs. As a result, we developed a novel simulation scheme tailored for complex emergency evacuations, offering significant performance improvements and broader applicability to similar high-stakes scenarios.

1. Description

It's a typical day at the TUM FMI building: students wander the hallways, study in the Magistrale, and attend lectures in seminar rooms and lecture halls. Suddenly, a student transforms into a Zombie in one seminar room, setting off immediate panic. While clearly fictional, this scenario provides a powerful lens to examine emergency dynamics. Individuals react instinctively to their immediate environment instead of following a coordinated evacuation plan. In such situations, predicting crowd behavior becomes significantly more complex, as each person's movements can influence many others. The number of potential interactions grows quadratically with the number of individuals, leading to emergent behaviors that are difficult to anticipate or control.

In this project, we model such an emergency scenario within the FMI building using a simulated Zombie outbreak. The system consists of two types of agents: Humans, who aim to evacuate the building, and Zombies, who roam freely and initiate panic or infection through direct contact. Importantly, infection is not modeled as a passive or probabilistic transmission. Instead, it occurs through active, contact-based interaction: when a Human and a Zombie come into proximity, both execute behavioral logic that may transform the Human.

We adapted the ONE (Opportunistic Network Environment) simulator to implement this scenario, initially developed for simulating mobility and data transfer in delay-tolerant networks (DTN). While not intended for dense, contact-driven crowd simulations, the ONE offered a usable foundation. We made minimal but targeted modifications to support reactive agent behaviors and physical contact-based interactions—effectively hacking the simulator to approximate our use case without extensive changes to its core.

Given the complexity of modeling numerous agents with interaction-based logic in a confined architectural space, it was immediately clear that a straightforward, large-scale simulation would suffer severe performance issues. Rather than allowing the simulation to become infeasible, we proactively developed a reproducible scheme to partition the simulation into smaller, manageable sub-simulations. This approach maintains behavioral realism while significantly improving computational efficiency, making systematic testing and iteration feasible even under complex conditions.

2 Analysis

When studying human behavior in emergency situations, we must account for various influencing factors. These include stress tolerance, physical fitness, aggression levels, social awareness, personal relationships, observational skills, and many other cognitive or emotional traits—some of which may be difficult to quantify or even identify in advance. The typical scientific approach would involve controlled studies to isolate and analyze the impact of each such factor, forming the basis for detailed behavioral rules in a simulation.

In general, each individual in an evacuation can be modeled as an agent with both an internal state and access to a shared external environment. At each time step, the agent evaluates its current state and surroundings to decide on an action. Actions can include movement, communication attempts, or interactions with the environment. The internal state may capture aspects like emotional condition, stress level, fatigue, or current goals. This contrasts with the external environment, which, though perceived differently by each agent, is consistent across all agents regarding its underlying structure and contents.

Formally, let $\mathcal{A}$ be the space of possible actions, $\mathcal{W}$ the state of the shared environment, and $\mathcal{S}_n$ the internal state of agent n. The decision-making process can be framed in terms of the conditional probability distribution:

$$p_{w,s,n}(a) = Pr[A = a | W = w, S_n = s]$$

This expression defines the probability that agent n will choose action a given the environment is in state w and the agent's internal state is s. In a fully specified model, this distribution could be learned empirically, through behavioral data, or constructed heuristically to reflect plausible decision-making behavior in humans under stress.

However, in this work, we adopt a more pragmatic approach. Due to time constraints and the high complexity of modeling realistic human behavior in emergencies, we use a simplified rule-based model. Our focus is not on accurately replicating human psychology, but rather on demonstrating the practical feasibility of implementing agent-based behavior within the constraints of the ONE simulator.

By establishing a minimal yet functional agent decision model, we aim to explore how far one can push a DTN simulator toward representing crowd dynamics, interactions, and environment-aware behavior. This proof-of-concept lays the groundwork for future work where richer behavioral models could be integrated incrementally.

3 Design

To simulate realistic yet computationally tractable crowd dynamics, we implemented a simplified agent model tailored to the constraints of the ONE simulator. Each agent is represented as an autonomous node with a minimal internal state and access to partial environmental awareness. While this abstraction lacks the nuance of complete behavioral modeling, it allows us to explore emergent phenomena such as panic spread and congestion. We consider two distinct agents: humans and zombies. Zombies pursue the nearest human, while humans prioritize fleeing from nearby zombies as an immediate objective and attempt to reach an exit as their long-term goal. When a zombie catches a human, an infection may occur after a brief struggle. If successful, the human dies and subsequently revives as a zombie, as illustrated in Figure 1.

The following subsections outline the key design decisions and their implementation within the ONE simulator.

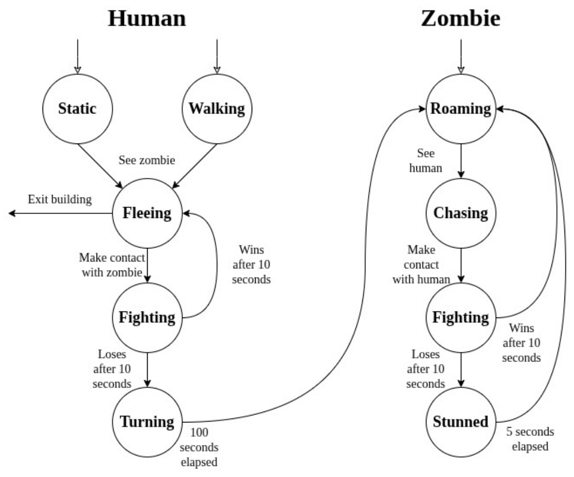

Figure 1. State machine of a moving entity in the simulation

Each subsection addresses a distinct aspect of the agent model — Mobility, Connectivity, and Simulation Transitions. This reflects the modular structure of the simulator itself. This alignment streamlined the development process and ensured that our extensions remained compatible with the simulator's core architecture.

3.1 Mobility

Agent movement in our simulation depends on their current state (see Figure 1). Humans and zombies follow different movement patterns that reflect the specific behaviors relevant to a zombie outbreak. The aim is not to model detailed daily routines—such as a student attending class or a staff member moving between offices—but to focus on the behavior that emerges after a human encounters a zombie. Therefore, we ensure that initial human positions are plausible, while subsequent behavior is driven by the unfolding simulation dynamics.

Some transitions between states rely on proximity-based perception. To model this, we define a perception radius of 10 meters for both humans and zombies. Within this range, an agent becomes aware of the presence of others and may react accordingly — for instance, a human may begin fleeing upon perceiving a nearby zombie. This simple model is sufficient for our purposes, as it approximates the range at which individuals might realistically recognize a threat in an indoor environment such as the FMI building. It also avoids the complexity of modeling occlusion or line-of-sight while preserving the essential dynamics of local interaction.

To facilitate movement, the simulation space is partitioned into convex, obstacle-free subregions, as described in Section 3.3. This ensures that any waypoint within a subregion is reachable via a straight-line path, simplifying pathfinding and reducing computational complexity.

Agents have a speed uniformly distributed between 0.5 and 1 m/s and update their movement behavior every meter, allowing them to react rapidly to nearby changes—such as the sudden appearance of a zombie or the movement of other agents.

Human-Static models someone attending a class, working somewhere, or waiting for something. As such, there is no movement.

Human-Walking models someone moving to another place. A destination is selected, and the human goes towards it until it is reached. Then, a new destination is selected, and so forth. In this movement mode, there is no collision with other humans.

Human-Fleeing models a human trying to escape the zombie outbreak. They try to reach an exits while avoiding zombies and collisions with humans. The direction of the movement $\vec{D}$ is calculated as follow:

$$\vec{D} = \sum_x \frac{\vec{F_x}}{\|\vec{F_x}\|}$$

with

- x and entity (exit, human, or zombie),
- $\vec{F_x}$ the associated attraction/repulsion force applied on the human fleeing.

In detail, with $\Delta\vec{P_x}$ the vector going from the fleeing human to the entity x:

- For exits:
$$\vec{F_x} = k_e \cdot \frac{\Delta\vec{P_x}}{\|\Delta\vec{P_x}\|}$$
- For other humans:
$$\vec{F_x} = -k_h \cdot \frac{1}{\|\Delta\vec{P_x}\|^{a_h}} \frac{\Delta\vec{P_x}}{\|\Delta\vec{P_x}\|}$$
- For zombies:
$$\vec{F_x} = -k_z \cdot \frac{1}{a_z^{\|\Delta\vec{P_x}\|}} \frac{\Delta\vec{P_x}}{\|\Delta\vec{P_x}\|}$$

For the choice of constants, our goal was to have (more or less):

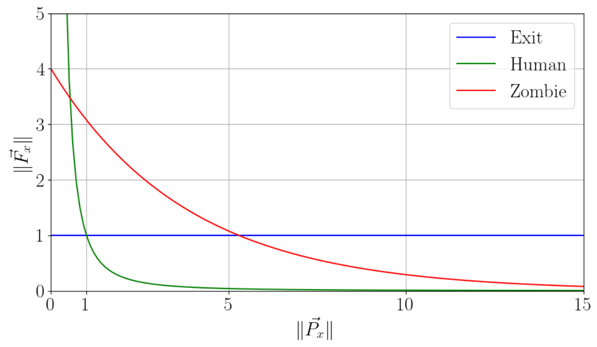

Figure 2. Relative attraction/repulsion strength in relation to the distance to the entity

- Collision avoidance with human with dominance of human repulsion under 0.5 meters (approximately) and no real significance above 2 meters;
- Zombie avoidance with dominance of zombie repulsion from 0.5 to 5 meters;
- General movement toward exit with significance of exit attraction above 1 meter, and domination above 5 meters.

These constraints led us to choose the constants as follows:

- $k_e = \begin{cases} 1 & \text{if chosen exit} \\ 0 & \text{otherwise} \end{cases}$
- $k_h = 1$ and $a_h = 2$ for all humans,
- $k_z = 1$ and $a_z = 1.3$ for all zombies.

When fleeing, a human chooses an exit depending on its priority: an exit with priority p_e has a probability $\frac{p_e}{\sum_{x \in \{\text{exits}\}} p_x}$ to be chosen.

Figure 2 shows that this choice complies with the constraints.

Human-Fighting models a human fighting with a zombie. Such a human doesn't move.

Human-Turning models a human turning into a zombie. Such a human doesn't move.

Zombie-Roaming models a zombie that is roaming, looking for humans. Similarly to Human-Walking, a destination is selected, and the zombie goes towards it until it is reached. The only difference is that, if an exit is in range, the zombie will choose it as a destination to allow exploration. Then, again, a new destination is selected, and so forth. In this movement mode, there is no collision with other zombies.

Zombie-Chasing models a zombie that is chasing a human. In this state, the zombie goes straight towards the closest human in range, who has no collisions with other zombies.

Zombie-Fighting models a zombie fighting with a human. Such a zombie doesn't move.

Zombie-Stunned models a zombie stunned by a human. Such a zombie doesn't move.

3.2 Connectivity

Our model uses connectivity between agents to represent physical contact rather than wireless communication. While the ONE simulator was originally built for opportunistic data transfer, we leveraged its contact detection mechanism to simulate direct interactions such as infection events and agent exits.

To support distinct types of contact events, we introduced two specialized `NetworkInterface` subclasses: `AgentInterface` and `ExitInterface`. In our setup, all agents are equipped with both interfaces, allowing them to engage in physical interactions with other agents (via `AgentInterface`) and to detect exits from the simulation (via `ExitInterface`). Exit nodes, in contrast, are assigned only an `ExitInterface`. Because the ONE simulator restricts connections to occur only between interfaces of the same type, this separation ensures that agent-to-agent and agent-to-exit interactions are processed independently and unambiguously.

The implementation details and behaviors of these interfaces are described in the following subsections.

3.2.1 Agent Interface

The `AgentInterface` manages connections between agents and determines whether a physical interaction should trigger an infection event. Each instance of this interface maintains a local state indicating whether its associated agent is currently a zombie. Upon establishing a connection with another `AgentInterface`, it evaluates the infection logic based on the zombie state of both parties.

If one agent is a zombie and the other is not, the interface probabilistically determines whether the human should be infected, simulating a brief struggle. This is implemented using a fixed infection probability and accompanied by short movement pauses to reflect the temporary nature of these interactions. If an infection occurs, the human agent is marked as deceased and revived as a zombie, triggering a state change in its interface and updating its identifier to reflect the transformation.

This logic is implemented within a custom subclass of `SimpleBroadcastInterface` and overrides the standard connection behavior. The full implementation can be found in the `AgentInterface` class in our simulation code[1].

3.2.2 Exit Interface

The `ExitInterface` models interactions between agents and exit points in the simulation. Unlike physical contact between agents, a connection to an exit represents a departure from the current simulation space. This mechanism is central to our strategy for managing simulation complexity, which is discussed in the following section 3.3 on simulation transitions.

The interface enforces a fixed throughput constraint to reflect the limited capacity of physical exits: only one agent may exit per second. This constraint is implemented by tracking the time of the last successful connection and rejecting any new connections until the interval has elapsed.

When an agent establishes a connection with an `ExitInterface`, the connection is logged, and the agent is effectively removed from the simulation. Since the ONE simulator does not support dynamic node removal, we achieve this through a small workaround: the agent is deactivated and excluded from further interactions. This process is discussed in more detail in the following section 3.3 on simulation transitions. Exit nodes themselves are stationary and equipped only with this specialized interface. In contrast, agents are equipped with both `AgentInterface` and `ExitInterface`, allowing them to participate in both types of interaction. The full exit behavior is implemented in the `ExitInterface` class[2].

3.3 Simulation Transitions

Given the performance limitations inherent in simulating large-scale, densely interactive agent populations within the ONE simulator, which is single-threaded and not optimized for continuous, spatially embedded interactions, we adopted a modular simulation strategy. Rather than executing a single monolithic simulation, we partitioned the

[1] See `AgentInterface.java` in the project repository.

[2] See `ExitInterface.java` in the project repository.

environment into a directed acyclic graph (DAG) of interconnected sub-simulations. Each node in the DAG represents an individual room or section of the environment, and edges correspond to one-way exit points through which agents may transition.

In our simplified scenario, rooms are modeled as rectangular, obstacle-free regions, but this spatial abstraction can be generalized to arbitrary layouts. The key constraint is that transitions between areas must form a DAG to ensure simulations can be scheduled without circular dependencies as shown in Figure 3. This structure enables the independent simulation of each room, with simulations at the same DAG level able to execute in parallel, significantly improving scalability.

We introduced a mechanism for logging and replaying inter-room transitions to implement this architecture within the ONE simulator. This is achieved through the `ExitReport` class, which acts as a `ConnectionListener` and tracks when an agent establishes a connection with an exit node (represented as a stationary host with an `ExitInterface`). Upon such a connection, the `ExitReport` logs an `ExitEvent` that encodes the agent's group ID, the type of agent (human or zombie), and the simulation timestamp.

These events are serialized and stored after each sub-simulation, and are later replayed in downstream simulations using the `ExternalEvents` mechanism provided by the ONE simulator. Specifically, each `ExitEvent` triggers the activation of a dormant agent at the corresponding entrance of the following simulation segment.

Due to a limitation in the ONE simulator—namely, the lack of support for dynamically removing hosts from the simulation—we introduced a custom `Activatable` interface, implemented by both `AgentInterface` and `ExitInterface`. While the simulator internally tracks host activity, it does not expose a method to toggle this state. Our implementation fills this gap, allowing hosts to be explicitly deactivated when they exit a room. These inactive hosts are assigned the `NoMovement` model to ensure they no longer interfere with other hosts through the `ApocalypseControlSystem`.

To streamline the pipeline and eliminate manual configuration, we also developed a standalone utility that automatically generates settings files for each simulation segment, which can be used as a base for configuring the simulation. This tool, implemented in the `apocalypseSettingsGenerator` package, uses a CSV-based description of the building layout as input. The `BuildingGraphImporter` class parses this CSV, which defines rooms, their connections, dimensions, and the number of human agents. Then, using depth-first traversal, the `MainZombieApocalypse` class generates the corresponding settings files, assigns globally unique IDs to

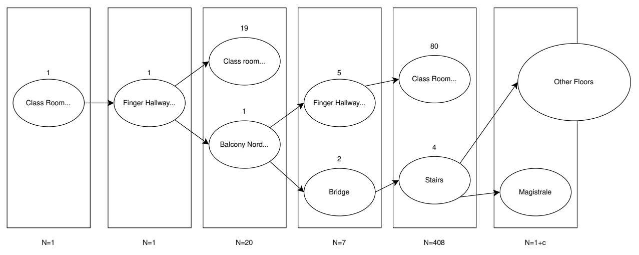

Figure 3. Graph simulating the FMI

all agents and transition nodes, and ensures proper linkage between simulation segments according to the DAG. This automation ensures consistency, reduces human error, and makes it easier to scale or modify the simulation environment.

4 Results

The speed of both humans and zombies is randomly assigned between 0.5 m/s and 1 m/s. The probability of a human turning into a zombie on contact is set at 20%, in which case a human will be immobilised for 100s before resuming movement. In the case of survival the zombie will pause for 5s instead. The total simulation time depends on the size of each room, ranging from 1500 seconds for small spaces like the balcony to 5000 seconds for larger areas such as the magistrale.

To model realistic evacuation behavior, exits that do not lead toward the magistrale—and thus not toward the building's main exit—are assigned a low priority value of 1. In contrast, exits that provide a direct path to the magistrale are given a high priority value of 100.

The outbreak begins with a single zombie placed in a designated "outbreak room" containing 15 students. The remaining rooms are populated with humans in proportions similar to those found in the FMI building. Room dimensions have been adjusted to realistically reflect the layout of the FMI. For simplicity, all rooms of the same type (e.g., classrooms) share identical sizes and entry/exit configurations.

The following factors are measured:

- How many people exit the Magistrale by simulation length
- which rooms and branches are untouched in the simulation

Table 1 shows that humans have a fairly easy time escaping zombies in the Magistrale due to zombies having a similar speed to humans. From 10.000 s simulation time onwards, there's an average of 3 extra humans escaping for every further 5.000 s simulated. Due to the realistic modeling of human knowledge of which exits,

Runtime in 1000 s	5	10	150	20	25	30	35	40	45	50
Humans exited	5	18	21	27	30	33	37	38	40	41

Table 1. Number of humans left by runtime in seconds

lead to the magistrale as show in Figure 3, only 25 % of rooms, which lead off of the first branch in the graph, that don't lead to the exit, are visited by the occupants of the "outbreak room". In the second branch, all humans choose the route leading outside, and the other fingers, as well as the other classrooms, don't experience any infection. Results showed that humans have a high probability of not leaving rooms if no zombies are present. Therefore, some humans stay on the stairs that lead to the other levels.

The objective of the project — to run a large-scale crowd simulation — was successfully achieved. We hope that the developed framework may provide a foundation for creating more complex and detailed crowd simulations in the future.

Acknowledgements

We used several AI-assisted tools during the preparation of this report to support both writing and implementation efforts. GitHub Copilot was employed to accelerate coding tasks and improve convenience during development for trivial tasks such as reading and writing into files, though it did not directly influence code quality. For writing, tools such as Grammarly, LanguageTool, and ChatGPT were used to enhance the clarity, consistency, and overall quality of the text.

While these tools contributed to productivity and refinement, all content was thoroughly reviewed and validated by the authors. We remain fully responsible for the correctness, coherence, and originality of the final work.

References

[1] I. M. Author. Some related article I wrote. *Some Fine Journal*, 99(7):1–100, January 1999.

[2] A. N. Expert. *A Book He Wrote*. His Publisher, Erewhon, NC, 1999.

[3] A. N. Expert. Some related conference article I wrote. In *Congress on Evolutionary Computation (CEC 2004)*, Portland, Oregon, 2006.

[4] N. H. M. Kilian Mio, Bisma Baubeau. Zomby apocalypse fork of The ONE Simulator. https://github.com/Mikilio/the-one, 2025. Accessed: 2025-06-11.