ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 37

Settore Concorsuale: 01/B1 - INFORMATICA

Settore Scientifico Disciplinare: INF/01 - INFORMATICA

MULTILEVEL MODELING AND SIMULATION: METHODOLOGIES AND APPLICATIONS

Presentata da: Luca Serena

Coordinatore Dottorato

Ilaria Bartolini

Supervisore

Moreno Marzolla

Co-supervisore

Gabriele D'Angelo

Esame finale anno 2025

Abstract

Multilevel modeling and simulation is a methodology that involves the hierarchical decomposition of complex systems into modular, cooperating components, each representing specific aspects of interest. The motivations behind this approach can vary widely, including the ability to leverage existing software, the need for different levels of abstraction and granularity, or the opportunity to better organize model development by separating semantically distinct aspects.

This thesis presents a comprehensive study of multilevel modeling and simulation techniques, providing an overview of their use in scientific literature and discussing the design principles and key issues involved.

The analysis of the current state of the art reveals that while multilevel modeling is widely used across various scientific fields, little effort was directed toward formal and methodological aspects. As a result, there are no precise standards for the design of such models, and ambiguities exist starting with the terminology. To address this gap and better define potential approaches for building a multilevel framework, the thesis proposes some categories of design patterns that address some critical aspects that may be encountered during the development phase, and a metamodel crafted to enhance multilevel modeling clarity and effectiveness.

The proposed design principles were ultimately applied to create multilevel simulations for IoT applications, with a focus on developing a decentralized, efficient sensor data marketplace. This scenario leverages technologies like LoRa and blockchain to support secure and scalable data exchange. Given

the complexity of factors involved — including mobility, physical message propagation, data storage and trading — multilevel modeling has proven to be exceptionally effective in managing the complex interactions and dependencies among these components.

Contents

Part I

Background

Chapter 1

Introduction

Modeling and simulation (M&S) is a wide family of methodologies employed in various fields to replicate and study the behavior of complex systems. The representations are centered on replicating the temporal evolution of real-world entities using mathematical or logical models, delineating the behavior of the system in response to certain events.

This methodology finds application across various scenarios and purposes, including:

- Investigations on highly complex systems, where it would be hard to study analytically the interactions among the parties involved;
- Assessing the impact of structural, organizational, or environmental changes on a system;
- Exploring the correlation between simulation inputs and outcomes to understand which are the most important variables and their interactions;
- Identifying potential threats to the regular operation of a system, such as vulnerabilities, bottlenecks, or scalability issues, to anticipate potential problems and deploy improvements ahead of time;

- Testing critical application scenarios, where it is desirable to run tests on digital twins before dealing with real systems, in order to avoid severe damages to things or people (e.g., medical applications, the resilience of buildings to major events such as flooding or emergency evacuations).

Regardless of the goals of the investigation, and the modeling methodologies employed, simulated systems are populated by a collection of possibly heterogeneous entities, which perform some role or task during the simulation and are characterized by specific attributes that denote mutable or non-mutable properties. In a simulation, the system continuously evolves its state, defined as the collection of all relevant information and variables that define the system's condition at a given instant. The state changes occur as a result of events, which are triggered by the behavior of some entities or by some inherent property of the model.

When dealing with systems of a certain complexity, it may not be feasible to conduct simulations using a monolithic architecture. In such cases, a multilevel approach can be employed, with the system being represented as a collection collaborating of sub-models, each one in charge of reproducing a certain "level" of the simulation. Multilevel modeling is an increasingly popular methodology in M&S studies due to its flexibility, allowing developers to select the most suitable modeling paradigm for each level of simulation. Additionally, developers can integrate existing task-specific simulators into the complex model, which saves time and reduces implementation costs. However, multilevel modeling is only a generic approach to simulation, as it encompasses a wide range of techniques that can be used to represent a system, including considerations on how to break down a system into various levels, the simulation paradigms employed, and the integration of different components. In the scientific literature, there are few standards and guidelines for designing and implementing these simulators, and many of them apply only to specific use cases or types of models. This lack of clarity starts with nomenclature, which is often ambiguous and inconsistent.

1.1 Research Methodology

The aim of the research has been to conduct a comprehensive study on multilevel modeling, examining the current state of the art and considering both methodological and application aspects.

The first step involved examining how this approach is utilized in the scientific literature. Specifically, the review consisted of 127 papers focusing on human behavior and mobility. This selection was made to restrict the domain since multilevel modeling techniques are used across a variety of scientific fields and to consider topics that can be understood without specialized knowledge in a particular scientific area. It emerged that while often terms are used ambiguously, with various expressions employed interchangeably despite potential distinctions, the multilevel approach for M&S is widely used by researchers from all over the world, and the number of papers using these techniques over the years is increasing, as shown in Figure 1.1. This trend has also been influenced by the arrival of COVID-19, which in recent years has led researchers to study transmission mechanisms using multilevel approaches, as shown in Figure 1.2.

Regarding methodological aspects, an analysis of the scientific literature revealed that multiple metamodels and formalisms are proposed for M&S, but little effort was devoted to the multilevel aspects. In particular, while certain formalisms describe interactions between modeling components within a complex framework, none of them is universally applicable across all modeling paradigms and problem semantics. For instance, certain metamodels assume the inclusion of multiple scales of detail or the employment of an agent-based approach, while others do not consider structural issues of multilevel modeling, such as criteria for invoking sub-models or policies for ensuring global consistency of the state of the system. The subsequent step was to identify general development principles applicable to any type of multilevel domain, regardless of the type of the models employed, their mechanisms of instantiation and the modes for data exchange.

Analogously, it emerged that while many design patterns are proposed for

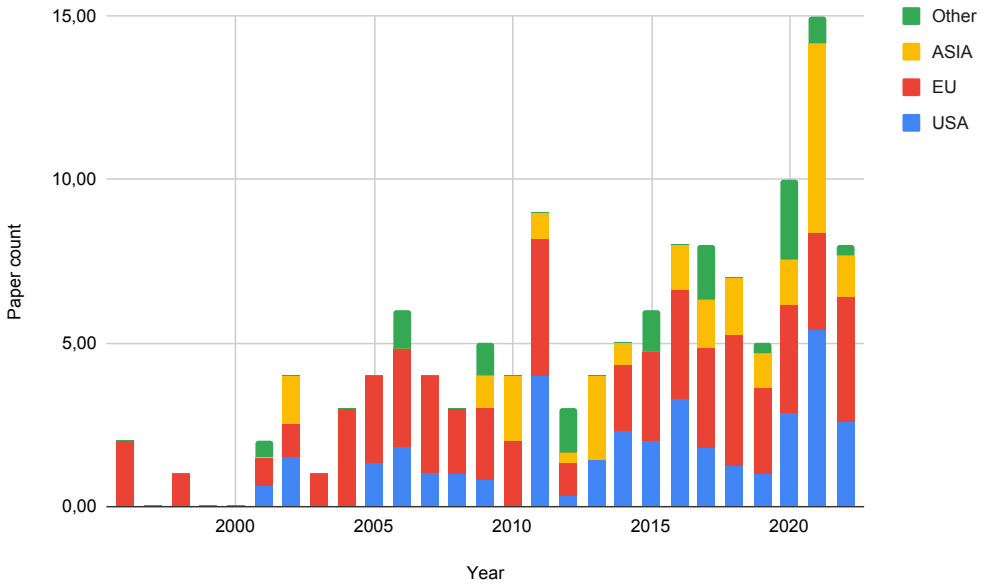


Figure 1.1: Paper counts by year and geographic area of the authors.

software development, few solutions are tailored for multilevel M&S. Therefore, several critical points have been identified in the design of multilevel models, such as the hierarchical structuring of various sub-models and their modes of interaction. The proposed solutions, which in part came directly from the software engineering domain, were then summarized into multiple sets of design patterns.

Finally, the methodological knowledge acquired was exploited to develop multilevel simulators that can investigate actual scenarios of interest. In particular, the focus has been directed towards smart territories, where Internet of Things (IoT) applications make use of technologies such as sensors, wireless connectivity, and blockchain to provide services. The considered case study was a decentralized crowdsensing architecture where vehicles act as data collectors and transfer data from sensing devices to networked access points. While the presented architecture is based on existing proposals, the novelty of the work is the simulation of the multiple factors of interest involved (i.e. wireless transmission, vehicle mobility, blockchain storage) within a

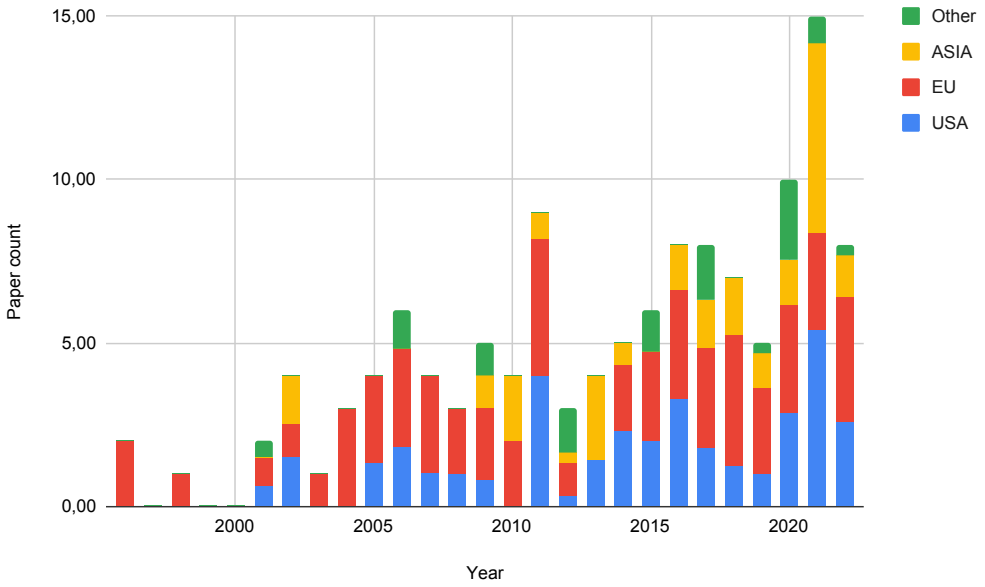


Figure 1.2: Breakdown of research topics by year.

single complex model, according to the principles of multilevel M&S.

1.2 Outline of the Thesis

The thesis is organised as follows. Chapter 2 is an overview of multilevel modeling and simulation. Chapter 3 reviews how the methodology is employed in the scientific literature. In Chapter 5 a novel formalism for the development of multilevel models is presented. In Chapter 4 some categories of design patterns are proposed to address recurrent issues that can be encountered in the development phase. In Chapter 6 some agent-based models are presented. Chapter 7 introduces an architecture for gathering and trading sensor data and shows how the behavior of the proposed framework can be reproduced through multilevel M&S. Finally, Chapter 8 provides some conclusive remarks.

Chapter 2

Background

This chapter introduces multilevel modeling, defining exactly what this expression means and which are the motivations for its use. The Chapter is organized as follows. Section 2.1 discusses the various types of modeling paradigms. Section 2.2 clarifies the use of the terminology. Finally, Chapter 2.3 examines benefits and challenges of the adoption of multilevel M&S.

2.1 Types of Models

M&S includes a wide variety of methodologies, as the various models can be characterized by different features and can adhere to different simulation paradigms. To comprehensively characterize models, they can be classified across three critical dimensions:

1. ***Time Management*** (continuous, discrete, mixed): This dimension is about how time is represented within the sub-model. In continuous models, time evolves in a seamless, uninterrupted manner. Conversely, in a discrete model time is split into distinct steps or intervals. Discrete models can either be *time-stepped* if time is divided into fixed intervals of uniform length or *event-driven* if changes in the system are triggered by specific events or conditions and time progresses based on such occurrences. Finally, mixed models amalgamate both continuous and

discrete elements in their temporal representation. Analogously to time, also the management of the space follows similar principles. The locations of interest, if any, can be represented in the simulation space by either discrete or continuous points.

2. ***Execution policy*** (sequential or parallel): This dimension describes how models are executed. Sequential execution follows a linear progression, where tasks are completed one after the other, using a single core. On the other hand, parallel execution involves simultaneous task execution, potentially enhancing computational efficiency by utilizing multiple processing units. The cost of parallelism is a more complex management of various execution processes, as multiple activities need to be synchronized to be consistently performed. A further option is Parallel and Distributed Simulation (PADS), where computational resources from various computers can be leveraged. The speedup that can be achieved through parallelization might vary considerably from model to model, so assessing whether this approach may be advantageous or not depends on the semantics of the problem.

3. ***Level of detail*** (low, high, adaptive): This dimension refers to the number of state variables that are used to represent the state of a model. Coarse-grained models employ a minimal set of state variables, offering a generalized view of the system and focusing on computational efficiency. Conversely, fine-grained models use an extensive set of state variables, enabling a comprehensive representation of the system under investigation, at the cost of a greater use of computational resources. Finally, adaptive models dynamically adjust their level of detail based on the system's complexity or specific conditions, allowing for a flexible and responsive representation.

Another crucial aspect concerns predicting the evolution of the models' state. While in deterministic models the evolution of the state of the system is entirely predictable from the initial conditions, stochastic models are

influenced by random decisions. Moreover, since experiment reproducibility is essential in simulation studies, it is always desirable for the generation of random numbers to be linked to a seed rather than volatile elements. In this way, other researchers will be able to replicate the experiments and assess the correctness of the results.

On the other hand, M&S paradigms refer to a set of principles and methodologies that guide the design and implementation of a simulation. They encompass the fundamental approach adopted to represent and analyze systems in a simulated environment. As far as simulation models are concerned, several paradigms are available, among which the most important are:

Agent-Based Models (ABMs) where the system is portrayed as a set of heterogeneous self-governing entities, whose behavior is defined by the features of the agent and by the interaction with other agents. With this methodology, it is possible to individually represent the behavior of each actor within the system under investigation, with a level of granularity that is contingent on specific design choices. Also, the connection between agents and the simulated environment holds significant importance. The expression "situated agents" is often used to indicate entities that engage with environmental objects to achieve their design objectives [1], to the extent that it would make no sense to describe their behavior outside of the environment in which they are located. An example of situated agents is vehicles in traffic models. In alternative scenarios, agents may not strongly connect with the environment, as the spatial aspect might be irrelevant to the simulation's purpose. In these cases, agents derive their behavior from interactions with other system entities. Peer-to-peer system studies, such as those focused on the interplay among various peers, exemplify this approach. For instance, in [2], an ABM is employed to assess the feasibility of cyberattacks on proof-of-work blockchains. In ABMs, the computational cost of the simulation depends on factors such as the number of agents,

the complexity of their behaviors, and the volume of interactions among the simulated entities.

Equation-Based Models (EBMs) where systems or phenomena are described by a set of differential equations that describe the relationship among various variables, capturing the dynamics, interactions, and dependencies among different elements of a system. EBMs come in various forms, including Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs), depending on the nature of the system and the dynamics involved. A particular type of EBMs are *compartmental models*, where the system's entities are divided into various compartments, each representing groups or categories of agents sharing common features, and the transitions between compartments are governed by a set of ODEs. Typically, EBMs are significantly less resource-intensive and time-consuming compared to ABMs. The efficiency in equation execution comes, however, at the expense of a reduced level of granularity, given that ABMs generally handles more aggregated representations.

Cellular Automata (CA) where the actors of the system are situated in a lattice of cells. The cells themselves act as agents in the system, and the evolution of the agents' states is influenced by the current state of the neighboring cells [3]. CA can exhibit either a continuous or discrete state space, while typically time advances over small discrete steps. CA find frequent application in complex systems investigations across various domains, such as physics, biology, social sciences, and computation. They are defined by simple local rules while demonstrating the potential to display emergent patterns and behaviors on a global scale.

System Dynamics (SD) where sets of differential equations describe the relations among the entities of the system, and feedback loops update the state variables over time [4]. These models can be thought of as a stock and flow diagram, where stocks represent accumulations

of quantities (e.g., populations or number of items sold), while flows determine the change rate of stocks over time. Mathematically, $(stock(t+dt) - stock(t))/dt = flow(t)$. SD helps analyze how changes in one part of a system can affect other components, allowing for a deeper understanding of the system's behavior and the possible long-term implications of different choices or interventions. This methodology was applied in first place in the World3 model [5], where five key sub-models (i.e. population, food production, industrialization, pollution, and consumption of nonrenewable natural resources) are integrated to encompass the most significant interactions between the Earth and human population.

Monte Carlo methods where stochastic simulations are employed to estimate the likelihood of specific scenarios or combinations of events to occur. This approach involves running repeated simulations with random inputs to generate a range of possible outcomes, from which probabilities can be inferred. Thus, the accuracy of these methods depends heavily on the randomness of the input sampling process. Monte Carlo simulations are particularly useful in estimation and optimization problems, other than in contexts where analytical methods are hard to apply due to the system complexity.

2.2 Definition of Multilevel Modeling

When representing a system of significant complexity, it is often convenient to implement a model in a manner that multiple views of the system are provided or various sub-models are aggregated to effectively collaborate. As the nature of these approaches may vary considerably, it is desirable to employ precise nomenclature that leaves no room for ambiguity. However, in the state of the art, the use of the terminology is quite inconsistent, as multiple expressions are often used interchangeably even though some significant differences exist. In this subsection, a definition of the technical terms is

provided:

Multilevel to generically indicate a framework where the system is depicted through semantically distinct models, each one describing a specific part [6]. By adopting this approach, it becomes feasible to utilize simulation strategies tailored to the considered scenario, enabling pre-existing task-specific models to collaborate within a more complex simulation environment. An example is epidemiological models, as frequently these types of studies consist of one model defining how a virus evolves within the body of a patient, while another model describes the dynamics of contagion at the population level [7].

Multiscale to indicate a framework where representations at different degrees of detail are integrated to simulate a system [8]. The term *multi-resolution* is sometimes used as a synonym. The goal of multiscale modeling is to strike a balance between the computational resources required for model execution and result accuracy. Opting for maximum detail is not always feasible, as it demands extensive execution time and also possibly generates vast amounts of data from fine-grained models that are hard to convey in the metrics of interest. Thus, multiscale representations enable a greater focus on the phases or spaces where the most important events occur. Secondary events, on the other hand, can be modeled more approximately, often without individually considering the behavior of all actors in the system but using aggregate representations. In a typical multiscale approach, micro and macro models alternate, with the option to include intermediate scales, such as mesoscale models. An example is traffic modeling, where micro models describe the behavior of all the vehicles in the critical areas, while the surrounding zones deal with the aggregate macro representation. However, multiple scales may not only imply different levels of detail in representations but can also influence the semantics of the system. For example, in DNA studies the highest level of resolution may involve quantum mechanics, while gradually decreasing granularity modelers

might deal with molecular mechanics and polymer physics [9]. Multiscale models can be further classified into two types: sequential and concurrent. Sequential multiscale models use information computed at a lower scale to solve problems at a higher scale, while concurrent multiscale models simultaneously compute quantities from both macro and micro scales during the overall model's computations, capturing interactions between scales [10].

Multi-layer to indicate a framework portraying the system's actors through a multilayered graph, which is a data structure comprising nodes, edges, and layers and whose interpretation of the layers depends on the implementation of the model [11]. A special case of multilayer graphs is multiplex graphs, which are a sort of network of networks that feature nodes distributed across multiple layers, each distinguished by distinct connections [12]. These models are often used to represent human relationships, where each layer represents some type of social connectivity (e.g., friendship, co-working, family).

2.3 Benefits and Challenges

There are several reasons to adopt a multilevel approach in simulations. Firstly, this paradigm allows the use of existing software, saving time and costs both in the implementation phase and during testing. Assuming trust in the accuracy of the available code, specific code for the relevant scenarios can be utilized, which is assumed to have been thoroughly tested and proven effective in reproducing certain system dynamics. Simulating everything from scratch often puts developers in the position of having to model certain types of systems they are not experts in, risking significant errors due to limited knowledge of the subject. Of course, this is a double-edged sword because, in the case of bugs or modeling errors, it becomes more challenging to identify and correct mistakes made by third parties, especially if the application's source code is not available. Secondly, even in cases where it

is necessary to build a system from the ground up due to the unavailability of effective simulators for the considered use case, multilevel modeling can be an efficient approach for organizing the development of the code as well. Organizing the model into various building blocks helps to more effectively divide tasks among developers. Furthermore, unlike monolithic architectures where a single model takes care of describing the whole system, in multilevel frameworks it is possible to commit changes to local portions of the code without affecting the other components of the complex models, thus making it easier to carry out upgrades and tests. Finally, the ability to represent a complex system at different levels of detail allows for a balanced tradeoff between accuracy and computational efficiency. Depicting a system at the highest level of detail is not always practical, as it can demand excessive resources and lead to prohibitively long execution times.

However, multilevel modeling also involves some concerning issues that must be addressed. Firstly, identifying optimal partitioning - whatever the definition of optimal may be - is not always straightforward. Some of these issues must be tackled during the design phase, where the various compartments of the complex model need to be defined. This involves determining what aspects should be represented in detail, what should be depicted more coarsely, and what can be neglected. In the case of multiple levels of detail, placing the most important events on a spatiotemporal timeline is not always easy, and often empirical rules are the only way to define transitions from a micro to a macro representation and vice versa. Another critical point concerns how to couple different sub-models, especially when dealing with different simulation paradigms or when there are different mechanisms for managing the progression of time. While standard interfaces for interoperation across simulators have been proposed [13], they tend to be quite large and cumbersome to implement. Metamodels and design patterns can provide guidelines that developers can follow during the design phase to determine how the sub-models will be coupled. These guidelines may relate to the structuring of the various sub-models, typically organized hierarchically,

how information exchange occurs between different components, and under what conditions a transition from one level of detail to another takes place; then it is up to the developer to decide which approach is the most suited, after examining the various pros and cons of the different proposals. Another crucial aspect involves ensuring consistency among all system representations. There are several instances where the risk of inconsistencies arises, such as when i) there is a partial overlap among various sub-models, ii) certain system actors transition from one sub-model to another, iii) the input for one sub-model is based on the output from other sub-models. Transitions between discrete and continuous models, as well as shifts from individual-based to aggregated representations, pose particularly significant challenges. Finally, a last issue concerns the generation of (pseudo-)random values, as it is often necessary to use one or more of these streams during the execution of a model. A common practice involves generating pseudo-random numbers from a single initial seed to ensure result repeatability. However, in a parallel or distributed setting each model has its random stream. Therefore, it is necessary to ensure that (i) the random streams do not overlap, meaning the initial seeds (or the pseudo-random generator) are chosen to guarantee independence among the random streams, and (ii) the outcome of the multilevel model remains unaffected by the order in which the sub-models are executed.

Chapter 3

Related Works

A part of the work presented in this chapter has been published in [14] and is reported here for the reader's convenience. Multilevel modeling, in its various forms, is employed in a wide range of applications, particularly in the scientific domain, where numerous studies have been conducted in fields such as chemistry, physics, biology, engineering, epidemiology, and sociology. However, given that many of these domains require specific knowledge of the addressed subjects, the state-of-the-art analysis focuses on topics related to mobility and human behavior.

3.1 Epidemic Modeling

Among the investigation areas that have been considered, epidemiology is the topic where most of the multilevel modeling works have been proposed, especially in recent years following the unfortunate advent of COVID-19. These studies aim to investigate the dynamics of disease spread, enabling predictions about how the epidemic will evolve and assessing the effectiveness of preventive measures such as mobility restrictions and vaccinations. Identifying epidemic peaks enhances the efficiency of risk assessment and resource planning, facilitating optimized allocation of resources such as hospital beds, medical supplies, and personnel.

From a semantic standpoint, there are multiple crucial aspects to model this type of system, leading to various combinations of sub-models found in the scientific literature. Given that pathogens spread through contact or physical proximity, human mobility plays a fundamental role in these scenarios, both at the microscopic level to model the local evolution of contagion, and at the macroscopic level to investigate virus propagation over a large scale.

The most recurrent multilevel modeling framework in the epidemiological area is to couple a *within-host model* that describes the evolution of viral agents inside the human body and a *between-host model* that reproduces transmission dynamics among individuals.

Multiple works are based on this methodology to investigate the diffusion of diseases such as Ebola [15, 16], Influenza [17], cholera [18], toxoplasmosis [19, 20], HIV [21], COVID-19 variants [22, 23], as well as non-human diseases like scrapie herd infection [24] or host-parasite coevolution [25].

Within-host models usually consist of a set of ODEs that describe the host-pathogens interactions, immune system responses, and the impact of therapies. Several within-host models exist, tailored to the specific requirements of different diseases. Despite the diversity, all these models typically include fundamental features such as the quantification and density of infected and healthy cells, mortality rates, shedding rates, and transmission rates of the pathogen.

Between-host models, on the other hand, usually adopt a compartmental approach, meaning that the population is divided into distinct and non-overlapping categories based on certain characteristics or states, in this case, the state of health of the people. Transition functions describe how individuals move between compartments over time, reflecting the dynamics of the modeled process. This modeling strategy is particularly suitable for large populations, where it becomes feasible to approximate epidemic trends by overlooking transmission dynamics specific to individuals or small groups. However, at a smaller scale, stochastic transmission events and population characteristics can significantly influence outcomes. In such cases, alternative simulation

paradigms like ABMs may be preferable [26], even though its use in multilevel frameworks is uncommon. The most popular scheme for between-host analyses is the SIR model, where individuals are either *susceptible*, meaning they are currently healthy but at risk of infection, *infected* or *recovered* [27]. The model is defined by the following set of ODEs.

$$\begin{cases} \dfrac{dS}{dt} = -\beta \cdot \dfrac{S \cdot I}{N}, \\ \dfrac{dI}{dt} = \beta \cdot \dfrac{S \cdot I}{N} - \gamma \cdot I, \\ \dfrac{dR}{dt} = \gamma \cdot I, \end{cases} \tag{3.1}$$

Transition rates between categories might depend on different factor parameters; in SIR, β depends on the number of contacts between infected and susceptible individuals, other than on the inherent contagion factor of the virus. On the other hand, the value of γ, often in the context of multilevel modeling, can be inferred from within-host systems and depends on how long the immune system typically takes to rid itself of pathogens. Of course, this model has certain limitations, such as a small number of compartments and the absence of births and deaths, resulting in a constant decrease in the number of susceptible individuals over time, and the inability to represent the mortality rate in the case of severe diseases. To cope with these issues, many alternative models have been proposed, as illustrated in Figure 3.1. In SIRD, people labeled as *infected* can transition to either *recovered* or *deceased*, in SIRV *susceptible* individuals can progress to either *infected* or *vaccinated* state, in SEIR there is a phase where individuals are *exposed* to the virus, meaning they have contracted the disease but they are still not experiencing symptoms [28], and finally in SIS people after healing are *susceptible* again, enabling the modeling of diseases that do not confer long-lasting viral immunity [29]. It is worth saying that the concepts of these models can be easily combined to create a more accurate representation of some diseases. Moreover, the transmission dynamics are significantly impacted by the nature of interactions among individuals. For example, in [30] the authors proposed a model where

social interactions at home, workplace, and public places are characterized by a specific exposure rate, defined as the probability of getting exposed while interacting with silent carriers.

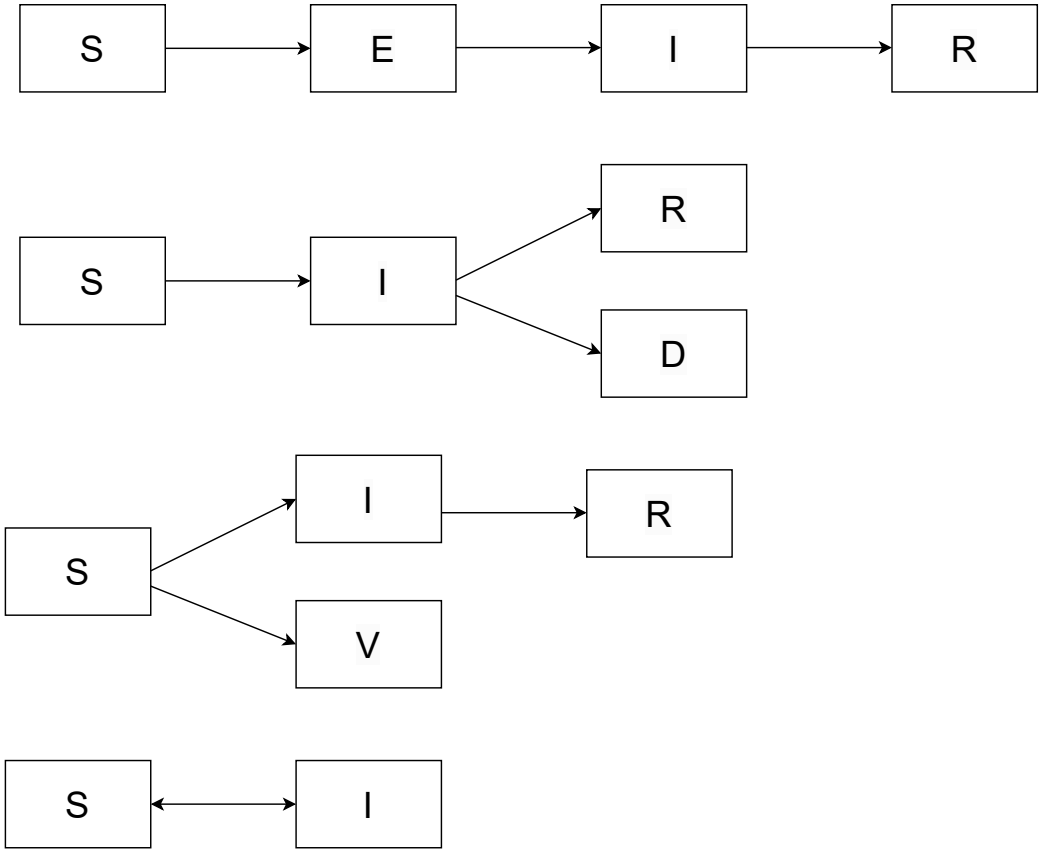


Figure 3.1: Graphical representation of compartmental models alternative to SIR. The meaning of the characters is the following: $S \implies susceptible$, $E \implies exposed$, $I \implies infected$, $R \implies recovered$, $D \implies deceased$, $V \implies vaccinated$

Another frequent approach in epidemiological multilevel modeling is to combine local compartmental models with macroscopic models describing the mobility of people over a large scale, in order to represent how pathogens spread over regions, countries, or continents. A globalized world unfortunately also entails a global diffusion of epidemic diseases, so information about human mobility plays a big role in the veracity of the model. Various models to represent the movement of people exist, and probably the most recurrent

scheme consists of having some type of graph-based data structure where nodes and edges represent either individuals and human relationships, or cities and routes. For example, in [31] an air traffic network is built based on real data from USA airports, in [32] a multi-layer network is used, with layers representing different means of transportation and nodes representing either central or peripheral cities, and in [33] the population is placed in a graph, where individuals move to different locations with some probability, and each node is characterized by a specific infection rate, recovery rate, and migration probability. In [34] a purely ABM approach is used, where micro agents represent individuals and meso agents represent groups of hundreds of people. Stochastic decisions are made for managing contagion between infected and susceptible individuals and for moving agents within and between cities. Multilayer graphs are also used for representing human interactions of different nature, like in [35] where multiple layers represent different social connections, such as the social and working environments of individuals [36]. Similarly, in [37] a multilayer network with inter-layer hopping (MNIH) is used, where individuals are the nodes of the graph, layers are different sub-networks and connections occur via both intra-layer and inter-layer edges. Temporal multilayer graphs are instead used in [38] to model the diffusion of sexually transmitted diseases, with two layers representing respectively steady and casual partnerships, while a SIS model is used to model the contagion aspects. Bipartite graphs are used by [39] to represent movements of individuals from their neighborhood to points of interest, based on anonymized mobile phone data.

Human mobility can be alternatively represented using an equations-based approach. In [40] a set of equations is used to describe a population of commuters moving over long distances (extra-urban) and a population of non-commuters acting over short distances (urban). In [41] a molecular-dynamics-based social-force model is employed to model passenger's movements inside an airplane during a flight. In [42] contagions occur stochastically when an infested and a susceptible individual are within the contact radius,

while in [43] a social-force model defines agents mobility, with the paper aiming at investigating over the effects of social distancing and mobility restrictions, considering both direct and indirect transmission. To achieve a realistic representation of inter-cities movements, one may take into account the composition of the population to generate mobility patterns that align with socio-economic factors. Following this approach, in [44] the population is represented as a nested hierarchy of sub-populations, defined as fractions of the overall population characterized by a common feature, such as ethnicity, gender, or age. In [45] a mobility model represents the bidirectional recurrent commuting flows that couple two sub-populations, while in [46] both small-scale commuting flow and long-range airline traffic are modeled based on actual airport traffic and connections between sub-populations. In [47] a multiscale framework is proposed, with an individual-based model that simulates contagion dynamics at the micro-scale, providing parameters to the macroscopic SIR model.

Finally, researchers may also want to investigate how certain social factors, such as awareness and the fear of the disease, influence the spread of the epidemic. In [48], the authors coupled the SIR model with a model that describes the risk awareness of the epidemic, concluding that the knowledge of contagion dynamics has a big impact on the diffusion of the virus.

In [49], a multilayer modeling approach is employed to capture both the epidemic contagion and the dissemination of opinions regarding social distancing, which can influence the infection rate. The overall adherence to social distancing rules is significantly influenced by social connections, as uninformed individuals can be swayed by either opinion based on their social neighborhood. Similarly, in [50] a SIS model is coupled with a graph structure representing a population made of different communities, which describes how dissemination processes of both pathogens and opinions over the epidemic are influenced by the interactions among communities. In this model, opinions are mainly influenced by the number of infected people, the dominant opinion within a community, and the dominant opinions within other communities.

3.2 Traffic Modeling

Traffic modeling predicts traffic flow by considering factors such as the influx of vehicles, drivers' behavior, road geometry, and traffic signals. These models play a crucial role in the design of new roads, intersections, traffic policies, and transportation facilities, offering assessments of their potential impact on traffic flow and safety. Additionally, these models can be employed to test traffic conditions during emergencies, aiding authorities in implementing effective plans for evacuations and traffic management.

Multiscale approaches are often used in traffic modeling, allowing for a greater emphasis on areas of higher interest. A typical framework consists of a microscale model describing in detail the behavior of each vehicle, a macroscale model dealing with the aggregate representation of the traffic flow, and in certain cases a mesoscale model describing individual vehicles through simplified flow dynamics [51]. The primary advantage of this methodology lies in its ability to strike a well-balanced trade-off between simulation accuracy and computational efficiency, by 'zooming into" certain critical areas such as locations prone to congestion or incidents, while simpler macro or meso models are applied in other regions, optimizing computational resources.

Microscale Models Microscale models describe individually the behavior of vehicles, defining their movements in response to various environmental factors, including road features (e.g., route signs, traffic lights, number of lanes) and the presence of other vehicles in the vicinity. This information guides decisions on maneuvers like acceleration, braking, and steering. The state of a vehicle encompasses factors such as current position, speed, acceleration, vehicle characteristics like maximum speed and acceleration capabilities, and driver characteristics like reaction time, aggressiveness, or adherence to traffic rules.

The time and space in which vehicles move can be depicted in different ways. Some models use continuous representations, like those relying on differential equations, while others opt for discrete representations, as in CA.

Alternatively, space can be described through a graph-based structure, where edges correspond to road segments and nodes to the intersections.

Sometimes micro-scale traffic models are themselves multilevel, being composed of various sub-models that describe various aspects of driver behavior [52]. In particular one can find:

Car-following models where the drivers' behavior depends on the preceding vehicle in the same lane [52];

Gap acceptance models which establish the minimum gap that drivers accept in road intersections, determining if a vehicle from a secondary street can enter the main road;

Speed adaptation models which adjust the speed of a car based on road features such as speed limits or road surface quality;

Lane-changing models which encode the decision to change lanes on a multi-lane road link, e.g., to overtake another vehicle or to comply with lane restrictions [53].

All these aspects can be linked together to carry out a more comprehensive description of vehicle behavior, depending on the modeler's requirements. In particular, some of the most used frameworks to model traffic at the micro scale are:

- **Intelligent Driver Model (IDM)**, where a set of ODEs is used to determine the positions and velocities of individual vehicles. Parameters include desired speed, safe time headway, maximum acceleration, and comfortable deceleration [54].
- **Nagel-Schreckenberg (NaSh) models**, which are CA where cells are either empty or occupied by a car. Vehicle velocity depends on the front vehicle and additional factors [55].
- **Agent-based models** where individual vehicles are modeled as situated agents in the micro environments, and behavioral rules define

travel decisions and how vehicles interact with the environment and with other agents [56]. A wide range of (single-level) ABMs is used for the simulation of traffic [57], transportation [58], and autonomous vehicles [59]. Noteworthy simulators in this domain include MATSim [60] and SUMO [61]. To handle a large number of agents, many ABMs use strategies such as *downscaling*, which consists of simulating the entire system's dynamics based just on a fraction k of the total population [62]. Downscaling is often necessary when dealing with a substantial number of vehicles. By carefully choosing k, execution times can be significantly reduced with minimal loss of accuracy.

Macroscale Models Macroscale traffic models provide a high-level perspective on traffic flow, focusing on the overall behavior of traffic as a collective rather than individual vehicles. These models use aggregate parameters to describe traffic conditions and are particularly useful for studying large-scale transportation networks such as highways or major road systems. Key parameters in macroscale models include:

- *Density*, defined as the number of vehicles per unit of road length at any given time, indicates how crowded the road is.
- *Space-mean speed*, defined as the average speed of vehicles in a specific section of the road.
- *Flow*, defined as the number of vehicles passing through a particular point in a set amount of time. It helps to assess the volume of traffic over a specific stretch of road.

Some of the most used frameworks to model traffic at the micro scale are:

- **Network Fundamental Diagram (NFD)**, which describes the relationship between the number of vehicles in a network and the average flow in that area [63].

- **Lighthill, Whitham and Richards (LWR) model**, where the behavior of traffic streams is expressed through the continuity equation and an assumed equilibrium speed-density relationship [64].
- **Payne-Whitham (PW) model**, which uses a set of differential equations to describe traffic, incorporating mean speed, density, and the number of vehicles in a specific road segment over a unit of time [65].
- **Aw-Rascle-Zhang (ARZ) model**, which involves a set of second-order, nonlinear hyperbolic PDEs to describe traffic density and velocity [66].

Multiscale Models The integration of models at different scales requires careful coupling in order to ensure consistency among the various representations, regardless of the sub-models involved. The most common method for linking micro and macro models is the aggregation and disaggregation of variables [67, 68, 69, 70]. In the context of this approach, aggregation means that variables representing single actors of the system are grouped, treating the combined result as a single data point [71]. Following aggregation, specific attributes and features of micro agents are going to be lost unless some mechanism is used to save and retrieve them. In contrast, disaggregation is the inverse process of breaking down aggregated data points into individual components with their unique characteristics. This process entails the release in the system of new information, which can be either generated synthetically or derived from previously saved data. In each case, the new information must be meaningful for the simulation purposes and consistent with the semantic constraints of the model. For example, if disaggregation is employed to populate a zone at the micro scale to study in detail some important phenomenon, individual velocities should be defined in a manner that ensures that their average aligns with the macro value, possibly introducing some random variation to reflect what happens in real traffic [72]. In [73], the transition from the micro to the macro scale is handled by an entity called "agent upstream", which has the role of computing flows and densities of

the aggregations of vehicles. Analogously, an "agent downstream" considers traffic flow parameters to create an appropriate number of vehicles when switching from the micro to the macro representation. Another approach to manage effectively the switch of resolution is to introduce a *transition cell* at the boundary zones of the model in order to transmit limit conditions and manage the shift from continuous to integer values [74]. In fact, during the disaggregation process, little inconsistencies could occur due to the conversion from real to integer values. For example, if the average number of vehicles in a zone is 80.5, a detailed micro simulation of that zone will result in either 80 or 81 vehicles, resulting in a small loss or gain. The accumulation of these rounding errors may lead to significant variations in the overall number of vehicles. A possible solution is to save and accumulate the fractional part until it reaches a unit, at which point a new vehicle is created [75]. A similar approach is proposed in [76], where "fictitious" vehicles positioned at the boundaries between scales receive the information and pass it on to the other vehicles.

In [77] and [78] the authors propose equations-based solutions to address both local consistency (i.e., agreement between models at different scales) and global consistency (i.e., overall preservation of traffic characteristics after several changes of scale). Local inconsistencies may arise when traffic variable values are not coherent during scale transitions, leading to issues such as violations in vehicle conservation, abrupt jumps of average speed, and unrealistic bottleneck effects. On the other hand, global inconsistencies may occur when information about traffic composition is lost during the conversion between different scales. For instance, properties computed in a micro model, such as route choices, vehicle features, or driving styles, might be lost in the transition to a macro model, and thus such information would be unusable when switching back to the micro model unless it was properly stored. In [79], the authors introduce a service-oriented middleware designed to facilitate collaboration between a macroscopic simulator and several microscopic ones. The middleware provides a common interface

that can be used by all the parts of the architecture, defining the modes of interaction, orchestrating the execution of the various components, and translating data between different scales. In [80] and [81] a tightly coupled system of partial and ordinary differential equations has been employed to model, respectively, traffic dynamics and the behavior of specific vehicles, such as autonomous cars or particularly slow and cumbersome vehicles.

While most of the approaches mainly focus on the microscopic and the macroscopic scales, other levels of resolution can also be considered. In mesoscale models, usually, vehicles are still handled individually using parameters from the micro model [82, 83], and their movements are simulated according to speed-density relationship or based on a macroscopic flow model, as proposed in [84]. Similarly, in [75] the *Underwood* model operates at the mesoscale, differing from NaSh model by how the vehicles handle their speed, which is dependent on factors like maximum speed, current concentration, and capacity of the road. In [85] the meso model is inspired by the macro NFD model, but with the additional capability of keeping track of single vehicles. Since mesoscale models still treat vehicles individually, there is no need for aggregation during the switch from the micro models. Consequently, the model coupling is generally more straightforward, although certain consistency constraints between scales may still need to be applied. On the other hand, in sub-microscopic scale representations, all the technical aspects that can potentially influence the driver's behavior are considered. For instance, in [86] the sub-microscopic level is depicted using a bond graph, where vehicle dynamics are presented in response to acceleration, braking, steering, considering longitudinal, lateral, yaw, and actuator dynamics of each vehicle. In [87], the authors proposed a framework that combines sub-micro, micro, meso, and macro scales. In this scenario, the sub-microscopic level involves an interactive exchange of information between a driver and a vehicle model, to determine driving strategies such as braking or steering. On the other hand, the mesoscopic scale is characterized by functions that establish the probability of having a vehicle within a certain space and speed range in

the unit of time.

A noteworthy variation of the previously mentioned approaches has been described in [88], where vehicles are assumed to be equipped with short-range communication technologies that enable data transfer with nearby vehicles. Traffic data from the surrounding zone can be collected and aggregated locally within a group of vehicles and then shared at a higher level among groups. This high-level processing is realized by some intelligent agents representing "traffic management centers". The model is based on a combination of ABM and CA, where agents represent vehicles and traffic management centers, and the CA represents the environment where agents operate.

In the process of creating the simulation environment, either synthetic or real information is employed depending on the research scenario. In certain works, the goal of the experiments is to investigate traffic dynamics of precise locations, so that geographical, environmental, and population data of specific areas must be considered. On the other hand, certain works adopt a more generic perspective, so that the testbed is a generic environment on which to perform tests. A more complex approach is presented in [89], where a synthetic population is created according to the characteristics of the area, and the daily activity plans are generated based on data from a survey done to a portion of the population.

3.3 Crowd Modeling

Crowd modeling is applied to analyze how individuals collectively behave in confined spaces or events, to understand how crowds move, interact, and respond to different stimuli. There are some similarities between crowd and traffic modeling, as in both fields the mobility aspects can be described at various levels of detail. Usually, an ABM describes entities at the micro level, and EBMs deal with aggregate metrics like density, flow, and average speed at the macro level. However, studies on crowd mobility differ widely in purpose, often combining the description of pedestrian movement with other aspects

of the system. In particular, in the state of the art, many multilevel models have been used to study pedestrians' response in emergencies like evacuation due to fire or earthquakes or to investigate interactions between pedestrians and means of transport.

Modeling Approaches Similarly to traffic modeling, a multiscale approach can be exploited for describing pedestrian movements by alternating microscale representations where the behavior of pedestrians is depicted individually to coarser-level scales dealing with groups of people or crowds.

At the microscopic level, the movements of pedestrians are modeled in relation to their surroundings, considering factors like destination, obstacle avoidance, and aversion to crowded areas. To encompass all these aspects, various approaches such as *social force models*, *centrifugal models*, and *agent-based approaches* can be employed [90]. Social force and centrifugal models introduce attractive/repulsive forces between objects in the environment and individuals, while the agent-based approach treats each person as an intelligent and autonomous entity. The macroscopic level, instead, refers to the aggregate dynamics of groups of people, dealing with averaged quantities such as density, momentum, and energy [91]. Finally, Mesoscopic models lie somewhat between the micro and macro level and consider groups of pedestrians neglecting detailed internal interactions, with the purpose of capturing group dynamics while still maintaining some control over individuals [92].

3.3.1 Emergencies

The simulation of emergencies and in particular those involving evacuations has gained particular interest from researchers, as a significant volume of research has been carried out to investigate them. Within these scenarios, a specific subset of agents, referred to as *leaders*, assumes the critical role of identifying secure exits and guiding others to safety. In [93] a strategic guidance model defines how leaders navigate towards exits, exploiting their knowledge of the environment to assess both risks and congestion, while a

pedestrian-following model delineates the behavior of followers, who respond to the leaders' guidance. Similarly, in [94] the leaders make strategic decisions on the route to take based on their macroscopic view of the environment, while the operative aspects regarding pedestrian movements are described at the microscopic level. In [95], a multiscale framework was proposed by the authors to simulate the response to a tsunami alert. At the micro level, people's behavior is described by a leader-follower model where agents are marked either as *leaders* or *followers*, based on their knowledge of how to respond effectively to a tsunami alert. On the other hand, at the macro level, the flow of pedestrians on the road network is represented by a LWR model.

A highly studied field concerns the evacuation of buildings in case of fires. In these scenarios, a multilevel approach can be employed in order to have separate models for describing respectively pedestrian mobility and fire/smoke propagation. In [96] a cellular automaton describes the evolution of the blaze inside of a building, while in [97] and [98] smoke and fire propagation are described by a set of equations. In [99] where Markov-chain models are used to mimic both smoke propagation and people evacuation, while in [100] the authors reproduce a cinema evacuation scenario by integrating a building model with a complete city/district model to simulate daily activities in the areas surrounding the building. A similar approach can also be employed to simulate the evacuation process resulting from other destructive events, such as floods. For instance, in [101] a continuous flood model describes the spread of water over a city, while an agent-based model mimics the behavior of pedestrians heading from dangerous regions toward safe areas. Similarly, in [102] the authors consider the extreme scenario of dam failure by combining an ABM that takes into account diversity in individual behavior and social characteristics for describing evacuation dynamics, and a hydrodynamics model based on diffusion equations. Both levels are kept synchronized: the hydraulic simulation is executed first, and the results are integrated into the ABM. It is worth observing that the temporal scale of the models is different (10 minutes vs 5 seconds) so that the water depth is updated only

after 120 steps of the evacuation model. Fluid dynamics are not necessarily employed to describe liquid propagation, as these equations can also be used to simulate the movement of individuals in conditions of overcrowding, a situation often encountered during evacuations or concerts. In [103] the authors integrate an ABM with Smoothed-Particle Hydrodynamics (SPH), a method where fluids are represented as a collection of particles that interact with each other according to a set of physical laws. To couple the two components, each agent is treated as a SPH particle subject to various types of forces that ultimately drive the speed and acceleration of agents. Finally, a multiscale approach is employed in [104], where the physical space in the evacuation model is either continuous or represented as a network. A continuous space representation is employed in areas demanding fine-grained resolution, allowing for a sophisticated depiction of pedestrians' behavior, with different types of agents reacting diversely to environmental stimuli. Each agent is described by a set of continuous attributes such as position, velocity, body frame width, and others. In regions where a high level of detail is not required, a network approach is adopted. Natural partitions like rooms or corridors become graph nodes and connectivity elements such as doorways serve as edges. Nodes and edges have a capacity defined as the number of people that a space partition can contain and the maximum number of agents that are allowed to traverse an edge at any given time.

3.3.2 Interactions with other means of transportation

Considering pedestrians' behavior in contexts where they share spaces with various media of transportation may help in evaluating and enhancing the safety of the urban environments, and identifying potential risks. In [105] the authors investigate railway crossing intersections, using a discrete-event model to represent pedestrians and a continuous model to represent trains. In [106] the risk of collisions between vehicles and pedestrians at road intersections is evaluated with a multilevel modeling approach, considering both micro-environment factors (e.g., the characteristics of the intersections like traffic

signs, vehicle flow, road features, and crosswalks) and macro-environment factors such as the urban design and land use of the areas surrounding the intersection. In [107] a queueing network model simulates vehicular traffic, public transport, and pedestrians in low-density conditions, while the movement of pedestrians under dense conditions is described by a force-based model. In [108] the purpose is to simulate crowd behavior in the context of public events. While a macroscopic model describes the arrival of people via shuttle bus, the movement of pedestrians inside the event areas is modeled at three different levels: *strategic* to define the target of the individuals, *tactical* to determine the route toward the destination and *operational* describing the actual walking behavior. The operational level is described at either microscale (for cramped and narrow areas) or mesoscale via a cellular automaton model, with transition zones managing the scale switching. Although agent-based models are the most frequently used approaches for crowd simulation, cellular automata are used as well. In [109] a cellular automaton represents people's movements inside the two ferry terminals, while a mesoscopic graph-based model simulates the journeys of ferries.

3.4 Urban Issues

While the movement of vehicles and pedestrians is a crucial aspect of city life, some simulation studies focus on other urban issues, such as the morphology of cities and the activities of their residents. While in this broad field the purposes of the research are the most diversified among the considered areas, a common trait is the importance of real geographical data due to the importance of representing the environment under test as accurately as possible.

3.4.1 Morphology of cities

Environmental considerations are becoming increasingly important in the planning of urban structures and infrastructure, as it is necessary to take

into account the impact of climate change and the more frequent occurrence of destructive events such as floods, droughts, or extreme heat. In [110] the authors provide a Urban Integrated Assessment Framework (UIAF), designed to evaluate the impact of the climate and economic change in the cities. The UIAF consists of a set of models that are coupled according to a three-level hierarchy. At the top, city models incorporate socio-economic changes and climate forecasts to analyze spatial patterns of a future population. City-zonal models downscale the above data to a finer spatial resolution, in order to estimate climate-related impacts using population, transport, and land-use models. Finally, zone-parcel models enable the simulation of the possible spatial pattern of housing development associated with the population prediction for each zone.

In [111], the authors address the issue of heat islands, referring to urban areas that exhibit significantly higher temperatures than their surroundings. The study employs three distinct models operating at various spatial scales, each considering different parameters like temperature and wind for the mesoscale and greenery, surrounding buildings, and pavement for the microscale.

Other research focuses on urban planning, aiming to assess the potential impact of new policies and infrastructure changes. For instance, Simmobility [112] is a simulator that integrates various mobility-sensitive behavioral models, allowing for comprehensive analyses that consider interactions among land use, transportation, and communication. Simmobility is characterized by three-time resolutions, ranging from fractions of seconds for short-term scale to days or even years for long-term resolution. At the long-term scale, the simulator represents strategic activities like job or house relocations, land development, or vehicle purchases. The mid-term level includes daily activity scheduling, route, destination, and departure time choices, encompassing a *pre-day model* for deciding the daily schedule of an agent and a *within-day model* for transforming the activity schedule into effective decisions and execution plans. Finally, the short-term scale takes care of traffic and pedestrian

mobility, modeling the movements of people, vehicles, and commodities.

Some papers have a specific focus on urban planning, such as in [113] and in [114], where city expansion dynamics are simulated considering respectively the use case of Wuhan and Auckland. In [113] the development process of an urban area is simulated by a three-scale framework: a macro SD model to predict the demand of new urban land use, a meso model to take into account intercity interaction to determine the areas for potential urban expansion, and a microscale model to represent neighborhood interactions through logistic regression. The three scales are then incorporated to form a cellular automaton describing the evolution of land use in the region. In [114] *government agents* manage the urban development at the macro scale according to zonal requirements and the needs of *resident agents*, which try to find the best place to live at the micro level. The ABM is then coupled with an artificial neural network for supporting agents' decisions about which non-urban zones to consider for city expansion. Studies on the potential urban growth of the city of Wuhan have been carried out in [115] through a multiscale hierarchical framework. At the macro level the probability of change is defined in the whole study area, at the meso level the density of change is defined only in the extent of land-cover change from non-urban to urban, and at the micro level the intensity of change is determined locally, detailing the specific extent of the changes. Similarly, in [116] the authors present a modeling framework aimed at replicating land use and land cover change processes in Costa Rica. A SD model uses data at different aggregation scales, showing how local, regional, and national trends can have opposite effects and results. Finally, some research works use a pure ABM approach for describing land-use dynamics. In [117] CA are employed both at macro level to simulate land use and transport infrastructure, and at the micro level to reproduce pedestrians' movement. In [118] the authors combine different agent-based sub-models describing how environmental, social, and economic factors change the land-use dynamics. Sub-models interact according to the concept of "co-modeling", which means that micro models are considered

agents of higher-level models.

3.4.2 Activities of Residents

In [119], the authors analyze air pollution considering the urban scenario of Madrid. PM_{10} dispersion is calculated by Computational Fluid Dynamics (CFD) models, which receive important inputs from other sub-models. Specifically, the Weather Research and Forecasting (WRF) mesoscale model provides boundary conditions for what concerns meteorological variables such as wind speed and direction and surface heat fluxes, while a microscale traffic emissions model provides hourly PM_{10} emissions.

In [120] a regional regression model is combined with a local agent-based model to describe deforestation processes in Amazonia. The macro regression model considers various environmental, demographic, agrarian structure, technological, and market connectivity indicators, while at the local scale two types of agents exist: small farmers who prefer lands close to roads or urban centers and big farmers who look for large pieces of inexpensive land.

Land use research has also been carried out through multilevel regression techniques. For instance, in [121] the authors constructed a predictive statistical model to explore land use in the Philippines. The investigation incorporated data at multiple levels, including field-level details such as the type of cultivation and land characteristics, household-level information like the ethnicity of family components, and village-level metrics such as the percentage of the population with a specific origin.

Some works are dedicated to issues concerning smart territories, a topical subject due to the recent advent of IoT applications. In [122] human mobility is integrated in a simulation of IoT and smart territories, focusing on a smart market scenario where users can subscribe their interests to some products or services, receiving information about events and sales in the neighborhood. In this work, two simulators are employed at two different levels of detail. At a coarser level an agent-based simulator uses PADS and Descrete-Events Simulation (DES) methodologies to describe dynamics

over the whole territory, with agents being involved in sales, subscriptions, and geographical movements. On the other hand, a fine-grained simulator can be triggered when needed in order to describe in detail the specific interactions within the smart market, considering wireless communication issues, interactions, and movements. In [123] a further level is added, with a set of equations describing the flow of customers and the parking strategies in the neighborhood. Here a wrapper is used to handle the interactions among the models, coordinating the execution of the two fine-grained simulators and synchronizing their activities with the higher-level simulator. Parking activities have also been studied in [124], where an NFD model describes the vehicle flow, while at the microscale a parking algorithm mimics cruising-for-parking activities. In this model, the decisions are influenced by factors like cruising speed, parking duration, and, of course, parking occupancy.

3.5 Social Sciences

M&S is a popular instrument in social studies, as it provides a means to understand, analyze, and predict complex social phenomena. However, the general approach to carry out multilevel modeling investigations on social sciences differs remarkably with respect to the other areas that have been previously considered. In particular, the concept of "level" in social sciences often denotes a point of view of the system rather than a component of a model. While models used in social sciences differ significantly in purposes and implementations, a common trait is that the dynamics of individuals are considered in relation to other individuals or groups. Given the diverse nature of human relationships, a common strategy is to use multilayer graphs for modeling social connections. In this representation, nodes correspond to individuals, edges symbolize connections between them and the layers are the various environments where specific connections occur. This methodology is popular especially in investigations related to the diffusion of trends, news, and ideas, since multilayer graphs are well suited for representing information

across various social networks to which individuals may belong.

For instance, in [125] two separate layers are employed to represent Twitter and FriendFeed links. Similarly, in [126] two layers represent two types of human relationships, and a SIR-based model is used to mimic information propagation, where individuals are either unaware of the information, or spreading the information to the contacts, or no longer spreading information. In [127] different layers correspond to different types of relationships. A multiplex network (i.e., multilayer network where inter-layer edges can only connect nodes that represent the same actor) is used to investigate society structuring, proving the hypothesis that society is composed of several strongly-tied communities which are in turn connected to each other by weak connections.

In [128], online and "real-life" connections are represented across different network layers, while information propagation is described through a compartmental model where people are labeled either as *Ignorant*, *Spreader*, *Variation* (i.e., people who diffuse an altered version of the information), *Oyster* (i.e., people who received information but are not actively spreading it) or *Recovery* (i.e., people who no longer spread information) [129].

Another noteworthy research area in the field of social sciences is *population dynamics*, which encompasses factors such as susceptibility to diseases, homophily, migration, and gentrification. The latter refers to the influx of middle-class or wealthy people in popular neighborhoods, causing the rise of prices that make it hard for low-income residents to cope with the new cost of living. To model these scenarios accurately it is important to collect a significant amount of reliable data, which is then analyzed at both the individual and aggregate level, in order to find useful insights supported by statistical evidence.

Numerous studies have embraced this approach. The first category of papers is dedicated to exploring links between health conditions and other individual characteristics. In [130] childhood obesity is analyzed by considering information at the individual level (sex, age, ethnicity), zip-code level

(median household income, lifestyle classifications, urbanization), county level (median household income, urban-rural distribution). In [131] the authors study correlations between obesity and diabetes, taking into account the characteristics of both individuals and the geographical area (e.g., poverty, population density). In [132] malaria diffusion in children is analyzed considering risk factors at the individual level, household level (wealth, education, sources of drinking water), and community level (place of residence). In [133] the incidence of visceral leishmaniasis in the Brazilian city of Teresina is studied census tract level (socio-economic and demographic information) and district level (prevalence canine infection, and insecticide spraying). In [134] the intent is to evaluate the efficiency of "Avahan", a HIV prevention program in India, taking into account variables at both individual and district level. In [135] a multiscale geographic weighted regression model has been employed to link economic and social indicators to road fatalities in Texas. From this investigation, it turned out that one of the most important factors associated with road accidents at the local scale is the average time required to go to work, while at the regional and global scale, one of the most concerning factors is driving alone, which has a negative effect on road security. In [136] the authors analyzed the role of socio-economic and environmental influence on food choice, specifically on fruit and vegetable consumption. This study considers data at both the individual level (survey data on fruit and vegetable consumption, individual and social influences) and neighborhood level (supermarket and fruit and vegetable store density). In [137], the authors investigate how sexual orientation affects the earnings of US citizens, considering data at both the individual level and the contextual level (i.e., the presence or absence of state-level anti-discrimination laws, and which political party was governing the state during the considered period).

A second category of papers about population dynamics analyzes the effects of *relocation* of individuals. In [138] gentrification is analyzed considering the case study of Philadelphia. The model incorporates features at both (i) individual level, including ethnicity, education attainment, marital status,

and income, and at (ii) neighborhood level, such as the gentrification process of a local area and neighborhood stability. Similarly, in [139] correlations between gentrification and voter turnout are investigated by examining the social landscape of Atlanta. In this work, variables are considered at both individual level (age, gender, ethnicity) and neighborhood level (average instruction level, gentrification rate, percentage of owner-occupied housing units), while also taking into account cross-level interactions between gentrification and longstanding voters. In [140] a multilevel modeling approach is proposed to analyze the migration phenomenon within Norway. In this work various approaches are proposed, considering both individual and aggregated features such as age, job, family status, education level, etc., which define individual probabilities of migration. The paper proposes models of different complexity, culminating with the event history models, where decisions are carried out based on data about the events that occurred throughout the entire life history of the individuals. Slightly different is the approach adopted in [141] to model the social integration of migrant workers in China, where a two-level data structure is employed, considering both individual and city levels. Similarly to other works, individual variables include demographic, social, and occupational characteristics of migrants, while city variables concern the nature of a city, economic conditions, population structure, language and culture, and institutions. In [142], a study on human mobility considers features at the micro level (household and housing units), meso level (groups of micro-agents and urban sectors), and macro level (city characteristics, urban planning), where people's mobility is influenced by agents similarity and affinity between social groups and urban sectors. Research on population dynamics extends beyond the human species. For instance, in [143] the authors study *poikilotherms* (i.e., animals with fluctuating internal temperatures, such as lizards) with the goal of estimating mortality and fecundity rates. In the proposed approach, an individual-based model predicts the number of produced eggs based on the current adult population, while a compartmental model describes population dynamics through the means of a set of ODEs.

Lastly, in [144] a multiscale approach is proposed for social sciences analyses based on the well-known prey/predator model [145]; the variation in the number of predators and prey is defined by differential equations at the macro level, while the micro model is in charge of describing individual behavior of both preys and predators.

It is worth noting that ABMs are very popular in the social sciences [146], even if they are not frequently employed in multilevel frameworks. However, we believe that multiscale approaches could become increasingly relevant in the future, as the recent development of multiscale ABMs platforms such as LevelSpace [147] could boost the use of this methodology. LevelSpace is an extension of NetLogo [148], a widely used tool for implementing agent-based models. Through LevelSpace, developers can develop applications composed of multiple interacting NetLogo models that are structured hierarchically, facilitating the creation of multilevel simulators and the integration of existing models.

3.6 Others

This section is used to discuss the papers that did not fit into any of the previous categories. The work presented in [149] introduces a supply chain simulation framework. While many studies in this domain rely on discrete-event models, this paper advocates a multilevel methodology that incorporates both continuous and discrete models. The activities of a company are modeled at different scales. The operational level describes activities in single plants over a short time horizon, while tactical and strategic levels address high-level policies. The goal of [150] is to quantify the economic impact of climate change on different stages of the potato supply chain, considering issues such as drought and extreme weather. The investigation is performed through a pure agent-based approach, with five levels appearing in the simulation: the cultivation, shipping, and processing of potatoes, the retailing, and the logistics of the transportation of commodities. A machine learning model is

also integrated into the simulation for predicting potato prices, taking into account yearly trends and seasonality. In [151], the lifecycle of commodities is described by an agent-based modeling framework, with multiple temporal scales involved. The agents of the model correspond to all actors involved in the process of producing, consuming, and distributing commodities. Three-time scales are used: long-term planning represents strategic activities such as commodity flows and logistic network formation, mid-term planning represents tactical decisions such as shipment generation, logistic planning, and vehicle flows, and finally short-term planning represents operative choices, such as scheduling, routing, and dispatching decisions. In [152] the authors propose a simulation environment that combines agent-based modeling and SD, taking a two-company marketplace as an example. At a high level, information collected from the behavior of customers is aggregated to describe trends. Customer decisions are influenced by high-level information, such as product popularity, and also by the choices of their peers, guided by a neighborhood model. The perspective of the companies is modeled through continuous feedback loops employing a SD approach, depicting the capacity to attract and retain customers. Fundamental elements for the described framework are the Continuous Agent-Based Modeling (CABM) Builder, responsible for managing the inputs from the agents to build the equations to be solved, and the CABM solver, which solves the various equations to compute the simulation results. Finally, in [153] a multilevel approach is used to study emotional psychology: the authors propose a model that maps the interactions among mood, emotion, and characters, which are defined on three different levels and with a set of equations, into changes in mood and emotion.

Part II

Methodology

Chapter 4

Design Patterns

A part of the work presented in this chapter has been published in [154] and is reported here for the reader's convenience.

In software engineering, design patterns are general design solutions to recurring problems. Unlike algorithms, where the step-by-step solution to a problem is detailed, design patterns provide abstract descriptions of how to address certain categories of common problems in software development. They do not offer implementation details, as these may vary based on the context and the technologies employed, but they outline the key concepts to overcome the issue. Often, it is crucial to offer multiple solutions to a specific class of problems, as the proposed patterns frequently come with various pros and cons. Depending on the context and the developer's needs, greater emphasis can be placed on one aspect over another.

The concept of design patterns in software engineering first appears in [155], where the authors proposed solutions for various classes of problems in the context of object-oriented programming. In particular, three categories of design patterns were discussed:

- *Creational* patterns, which deal with object creation mechanisms, abstracting the instantiation process and making the system independent of how its objects are created, composed, and represented.
- *Structural* patterns, which deal with the composition of classes or

objects, forming larger structures while keeping the individual elements of a structure independent.

- *behavioral* patterns, which deal with the interaction and responsibility of objects, defining how they communicate and collaborate.

Other than the categories proposed in [155], many other design patterns have been studied by researchers in the last decades, such as strategies for managing concurrency [156] or architectural solutions [157].

As design patterns could lead to a comprehensive investigation of many aspects of software development, in [154] specific solutions for M&S have been introduced. In particular, the proposed design patterns were classified into the following categories:

- *Orchestration patterns*, which deal with the flow of execution of sub-models.

- *Structural patterns*, which describe how sub-components can be aggregated into complex models. Note that these patterns are taken directly from [155] since they are relevant for multilevel modeling besides general Object-Oriented programming.

- *Information exchange patterns*, which define how data can be transferred between models of different types, e.g., continuous and discrete-space models.

- *Multiscale patterns*, which are used to define how models employing different levels of details can be integrated.

4.1 Orchestration Patterns

GEMMA demonstrates how organizing various components within a complex model can, in most cases, be represented as a non-binary tree. The root of this tree may be either i) one of the various sub-models, ii) a wrapper

script responsible for executing other sub-models, or more abstractly in the case of peer-to-peer structures, iii) the user itself manually executing the various components. Orchestration patterns illustrate strategies for arranging the tree structure of the various sub-models, defining how the execution flow moves from a parent to a child within the tree. These solutions are not mutually exclusive and can be combined when needed to create more complex hierarchical structures.

4.1.1 Models' Controller

This pattern is characterized by a key entity, the *Controller*, that takes on the role of a wrapper script, orchestrating the execution of all the instances of the underlying component(s), and providing higher-level functionalities, which include managing information exchange between models and exercising semantic control over the model's state and input/output operations. Additionally, often the *Controller* serves as the interface for the user, sitting at the top of the hierarchy. In this case, it can contain all the important parameters of the model, providing users with the ability to easily modify the key variables for conducting experiments, enabling the testing of various system configurations without the need to tinker with the source code of the sub-models. Moreover, when a system parameter is shared among multiple sub-models, it is advisable to define it at the top of the hierarchy. This ensures that the parameter is consistently updated and subsequently passed down to the underlying components, preventing inconsistencies that may arise if not all components are updated uniformly.

The introduction of a Controller effectively centralizes the scheduling and management logic, thereby enabling a clear separation of concerns between functionality and implementation. Furthermore, this design allows for greater flexibility, as the addition of extra sub-models is made relatively straightforward, as only the Controller is involved. However, a potential drawback that must be acknowledged is the risk of the Controller becoming overly complex when dealing with a substantial number of incompatible sub-models.

An illustrative application of the Models' Controller pattern can be found in [158], where it was employed to investigate crowd evacuation through a multilevel model. In this work, a synchronization module is in charge of scheduling the execution of micro and macro scales while managing the exchange of information.

4.1.2 Director-Worker

The Director-Worker pattern occurs when in a hierarchical structure the execution flow is passed from a parent node of the tree to one of its children. While the Director includes some functionalities of the Controller, it differs in that the Director itself is a sub-model, whereas the Controller is an external entity outside the model. Additionally, the Director-Worker pattern can be combined with the Composite pattern (see Section 4.2).

It is worth noting that when dealing with preexisting models, the hierarchical connection between two components requires some change in the source code so that at least the Director must be adapted to schedule the instances of the Worker(s). Similarly to Models' Controller, if a Director is the root of the tree structure it must serve as the user interface so that preferably the root contains the system's parameters and conveys them to the underlying components.

Models' Controller and Director-Worker patterns can easily be combined, as shown in Figure 4.1. For example, in [123] a simulator positioned at the top of the hierarchy employs a wrapper script to manage various instances of underlying models. In this scenario, the wrapper script can be considered both a Worker within a Director-Worker scheme and a Controller for the lower-level modules.

4.1.3 Director on Hold

Director on Hold is the simplest way to implement the Director-Worker scheme. In this pattern, the Director pauses while the Workers are active,

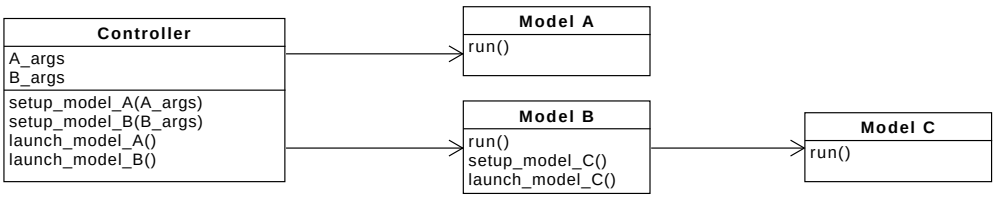


Figure 4.1: UML Class diagram of Model's Controller and Director-Worker combined

resuming its execution only after the tasks of the Workers are completed. Despite its simplicity, this approach may introduce notable overhead, in particular if the creation and destruction of Workers involve significant costs. While constructing and dismantling EBMs might be a relatively lightweight procedure, the same cannot be said for ABMs, where the creation of the agents and the storage of their state often constitute non-negligible activities. Moreover, the Director remains inactive during the Workers' execution, limiting the level of concurrency achievable within the model.

4.1.4 Worker on Demand

Worker on Demand is a design pattern that addresses the problem of the overhead associated with creating and destroying Workers as needed. When these procedures are both resource-intensive and repeated multiple times throughout the execution, a computationally efficient approach is to generate all Workers at the beginning of the execution and to keep them on standby. Then, whenever one or more Workers are required, such sub-models are called to compute some task.

The Worker on Demand pattern introduces a clear separation between the initialization of Workers and the execution of their tasks, bringing two primary advantages. Firstly, complex entities are generated only once, potentially saving significant time, especially when dealing with a large number of entities over the model's lifespan. Secondly, this approach allows for the storage and retrieval of entity states for further examination. In fact, in certain scenarios resuming the state of a Worker is a costly operation, requiring the creation

of temporary files to store information that would otherwise be lost upon the destruction of the Worker's instance. However, a notable drawback is that the Worker pool occupies memory space even when inactive, making this strategy unsuitable for memory-constrained environments.

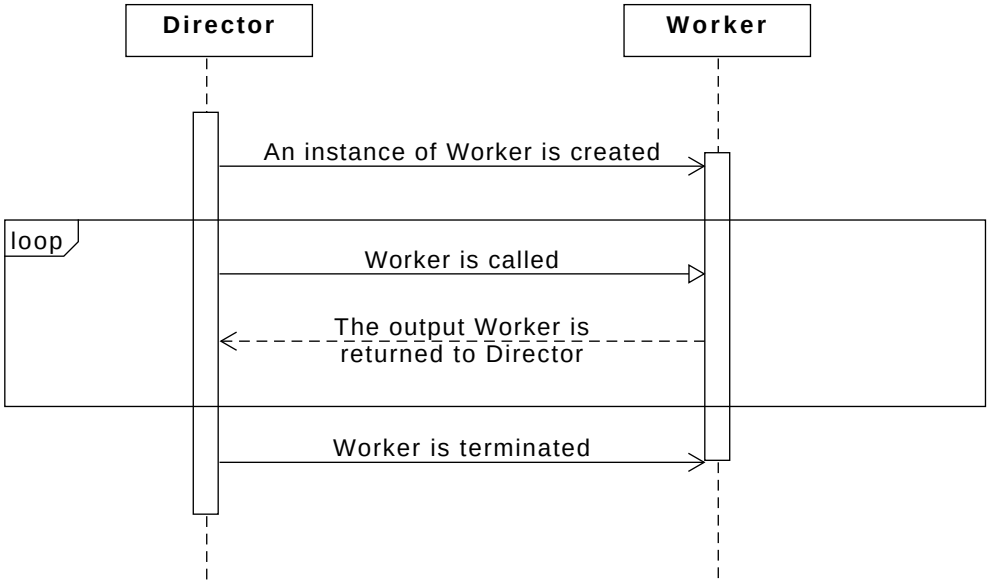


Figure 4.2: UML sequence diagram of Worker on Demand

4.1.5 Concurrent Modularity

The best way to improve the performance of a multilevel simulator is to parallelize its execution, running multiple instances concurrently whenever possible. Unlike PADS, where parallelism refers to the execution of parallel components or multiple instances of the same model launched together, the *Concurrent Modularity* pattern facilitates the simultaneous execution of semantically different models, potentially of varying types. However, concurrent execution requires meticulous time management, as different sub-models may adopt distinct granularities or time concepts, especially when integrating continuous and time-stepped models.

To address the time management issue, several strategies can be employed, such as:

- *Time translation mechanisms*, which translate the local time representation of a sub-model into a global time universally understood by all other components.
- *Checkpointing*, meaning that all sub-models progress in lockstep and synchronize periodically. When a sub-model reaches a checkpoint, its execution halts, resuming only after all other sub-models reach the same checkpoint. To optimize efficiency, it is crucial to determine an appropriate frequency for scheduling checkpoints. If the frequency is too low, it may lead to excessive overhead, while excessively long intervals are more likely to result in inconsistencies.
- *Rollback mechanisms*, which are methods for rolling back in time upon detecting inconsistencies, undoing recent updates, and reverting to a previous (virtual) simulation time where a consistent state was calculated [159].

4.2 Structural Patterns

Structural patterns describe how software elements can be composed into larger structures while promoting flexibility and code maintainability. The patterns described in this section are taken from [155], where they have been initially proposed in the context of software engineering.

4.2.1 Composite

Composite design pattern allows for the hierarchical composition of software elements according to a tree-like structure, enabling the uniform management of both atomic and composite objects. This facilitates the integration of new elements in the system as the interface for both atomic and aggregate components remains the same. The pattern is composed of four elements:

- *Client*, which is the element that interacts with the members in the composite structure.

- *Component*, which is an interface that defines the common methods for both the Leaf and the Composite, allowing them to be treated uniformly.

- *Leaf*, which is an atomic element that does not have children in the hierarchy.

- *Composite*, which is an internal node of a tree structure. It implements the *Component* interface but also maintains a collection of children, which can either be *Leafs* or other *Composites*.

Composite pattern can be applied for building hierarchical ABMs. In this context, macro agents may serve as containers of multiple micro agents. The *Composite* component may thus provide an aggregate coarse-grained representation of some segment of the model; when a higher level of accuracy is required the *Composite* executes the low-level agents that it contains, which in turn might be composite objects and contain some finer-grain agents.

This pattern could find application in [160], where the authors investigated the propagation of black rats via commercial transportation. In this work, the main building block of the simulator is represented by the concept of *World*, defined as a complete and self-sufficient sub-model with its own places, agents, spatial resolution and temporal scale. *Worlds* allow the creation of nested structures, potentially including other *worlds* describing the system at a higher level of resolution. The Composite pattern could be then applied to provide a high-level management of all the *Worlds* in the system.

4.2.2 Bridge

Bridge pattern allows for the separation of a component's interface (abstraction) from its actual implementation, enabling independent modifications

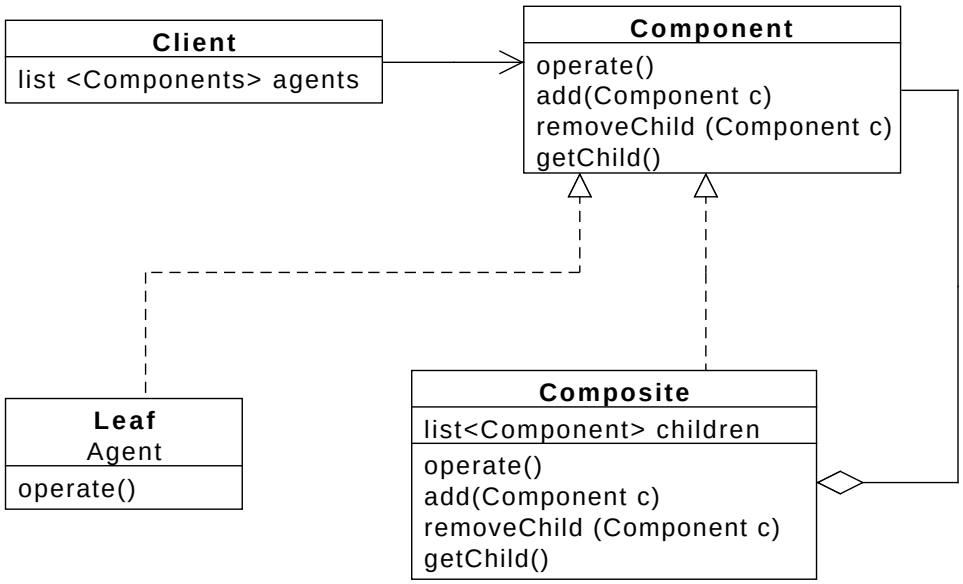


Figure 4.3: UML class diagram representation of Composite design pattern

to both. The Bridge pattern is composed of four main components, as shown in Figure 4.4:

- *Client*, which is the component that interacts with the *Abstraction* without being aware of the implementation details.
- *Abstraction*, which defines the high-level interface that clients will use.
- *Implementor*, which serves as an interface for describing the technical functionalities of the Abstraction.
- *Concrete Implementor*, which defines the concrete implementation of the Implementor.

Decoupling the interface of an object from its implementation is a common practice in object-oriented programming. This approach enables the definition of various implementations of an abstraction that can be used interchangeably, even at runtime, transparently to the client. This enhances code flexibility, as it becomes possible to make changes to the *Concrete Implementor* without

having to modify the interface through which these elements are perceived externally. Additionally, different *Concrete Implementors* can be defined, creating the opportunity to create heterogeneous objects with distinct behavioral rules that are uniformly perceived by the client.

Bridge pattern can find application in multi-agents systems, by treating agents as objects in object-oriented programming, enabling to separate the definition of the agents from the code that dictates the behavior of certain types of simulated entities. In particular, in multiscale modeling the pattern can be employed to treat uniformly atomic and aggregate agents, thus enabling various levels of granularity.

In simulations focusing on emulating human behavior, the Bridge pattern can be employed to handle diverse types of entities, including individual agents (micro-agents), small groups of individual agents (meso-agents), and large groups of individual agents (macro-agents). For instance, in [161], the movement of pedestrians is simulated with a hierarchy of crowds, groups, and individuals that coexist in the same simulated environment. Bridge pattern empowers developers to modify agent behavior without impacting how the *Client* interacts with them. By abstracting away implementation details, the *Abstraction* provides a simplified interface for the *Client*, facilitating the high-level management of agents

4.2.3 Adapter

Adapter is a design pattern that facilitates the collaboration of elements with incompatible interfaces by introducing components capable of translating one interface into another. The pattern is composed of four components:

- *Client*, which is the component that uses the *Adapter* to interact with the *Adaptee*. It might either be a wrapper of the *Models' Controller* orchestration pattern or a *Director* in a *Director-Worker* scheme.

- *Target*, which is the component that the *Client* wants to use.

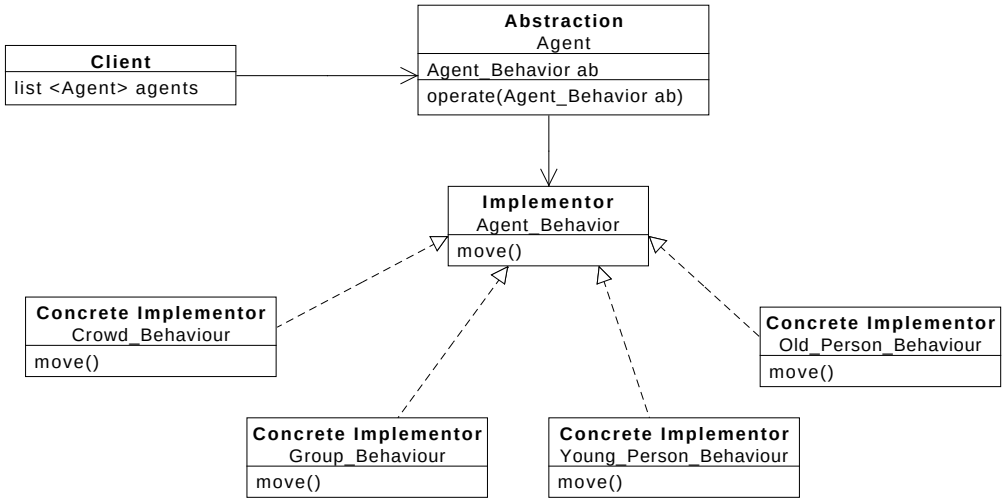


Figure 4.4: UML class diagram representation of Bridge design pattern

- *Adaptee*, which is the component that must be adapted in order to be used by the *Client*.
- *Adapter*, which is the component that acts as an intermediary between the *Client* and the *Adaptee*, translating the interface of the *Adaptee* to the interface expected by the *Client* (i.e., the *Target* interface).

Adapter can be employed in multilevel modeling, treating objects as if they were sub-model of a complex simulation. This pattern can facilitate the interoperability among components that were not originally designed to work together. This commonly occurs when building a multilevel model upon pre-existing components, allowing developers to make only minor changes to the source code of the *Client*. Another scenario occurs when the *Client* is working with a sub-model that represents the system at a specific granularity, and developers aim to integrate new sub-models at different scales.

Let us consider, for instance, a simulation of urban scenarios, where traffic is depicted at the micro scale, detailing the individual movements of vehicles. If developers wish to carry out more comprehensive studies, they may need to include the surroundings of the area under investigation, providing a macro-level representation of traffic characterized by aggregate metrics such

as density, average speed, and vehicle flow. In this case, developers could employ an *Adapter* to translate the interface of the macro model into the interface of the micro model, enabling its utilization by the *Client*.

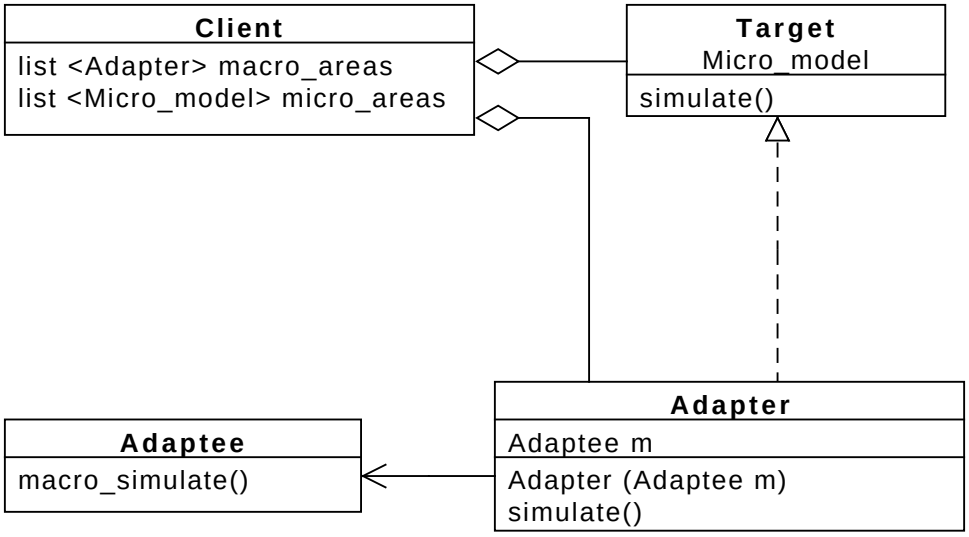


Figure 4.5: UML class diagram representation of Adapter design pattern

4.3 Information Exchange Patterns

The execution of various sub-models involves the need for exchanging information among the various components. Sometimes the flow of data follows a bottom-up direction, with Directors retrieving the information from the underlying Workers, other times the flow is bidirectional. Additionally, the involved components may employ different data representations, leading to the potential conversion of information from discrete to continuous or vice versa. Moreover, another crucial factor to take into account is the execution platform. In distributed environments, communication becomes more complex, necessitating the flow of data between distinct machines and potentially across different execution locations. Information Exchange patterns delineate various possibilities for exchanging data among sub-models and by considering these

mentioned implications, one can identify the most suitable approach among the available options.

4.3.1 Return Value

The *Return Value* pattern represents the most straightforward strategy for data exchange in a Director-Worker scenario. In this approach, the Director invokes the Worker, which then returns a result to the Director. While this strategy is elementary, implementing it becomes challenging when the Director is allowed to execute concurrently with the Workers. In such a scenario, Worker execution operates asynchronously, potentially completing before the Workers themselves terminate. To address this challenge, the *futures* pattern from concurrent programming can be employed [162]. Here, the Worker returns an object representing a *promise* to compute a result, and the Director will block if it attempts to read the result before it has been computed. Also, developers should check the consistency of the results, making sure that the returned values align with the expected outcomes. This can be accomplished through semantic controls or by using *exceptions* to effectively handle errors. When the caller and the called models operate on different machines, the concept of *remote procedure call* comes into play, enabling to treat communications between components in a distributed system as if they were local procedures, abstracting away the underlying network details. To facilitate this, a communication protocol like XML-RPC must be involved to manage the exchange of data. For example, in XML-RPC, XML is used for encoding calls, and HTTP serves as the transport mechanism. The communication process also involves serializing parameters and return values for transmission over the network, encoding data in a format that can be easily reconstructed on the receiving end.

A drawback of this pattern is that information exchange can only occur when the entire method call completes, potentially causing unnecessary delays. Additionally, if it is necessary for the Worker not to terminate its execution after fulfilling the method call from the Director, the Director must perform

two separate method calls - one to activate the Worker and another to execute the simulation module, as illustrated in the Worker on Demand design pattern.

4.3.2 Pipe Through Temporary Files

While the Return Value pattern involves a Director calling a Worker, the Pipe Through Temporary Files strategy, represented in Figure 4.6, offers a more versatile approach for communication in any execution scenario. Data consumers only need to be aware of the location of the file(s) and possibly when new data becomes available. In this solution, data are encompassed into files that are going to be erased just after the exchange of information is over. The organization of data within these files can vary based on the type of information and the requirements of both the caller and the callee. Commonly used formats for data representation in this context include JSON, XML, and YAML, which are human-readable and easy to write. Writing data into temporary files simplifies the process of adding fields within the list of values to be returned. This approach enhances flexibility and ensures a straightforward method for expanding the information exchanged.

Nevertheless, it is crucial to acknowledge certain limitations. File operations, such as creation, reading, and writing introduce a degree of overhead whose impact varies based on the number of operations performed. As the significance of this aspect depends on the scale of operations, it can be safely overlooked in case the impact on the execution time is negligible. Furthermore, supplementary mechanisms are required to alert consumers when new data becomes available. Establishing effective communication about data updates is essential for maintaining synchronization and ensuring that relevant parties are informed promptly. Finally, if the application terminates unexpectedly, the developers should implement some mechanism to delete all temporary files before a new simulation is performed. This precautionary step is essential to prevent the contamination of a new execution with outdated information from previous runs and to avoid the accumulation of meaningless files. A possible approach to this issue is to implement a simple mechanism that removes old

temporary files at the beginning of the execution before the actual simulation phase starts.

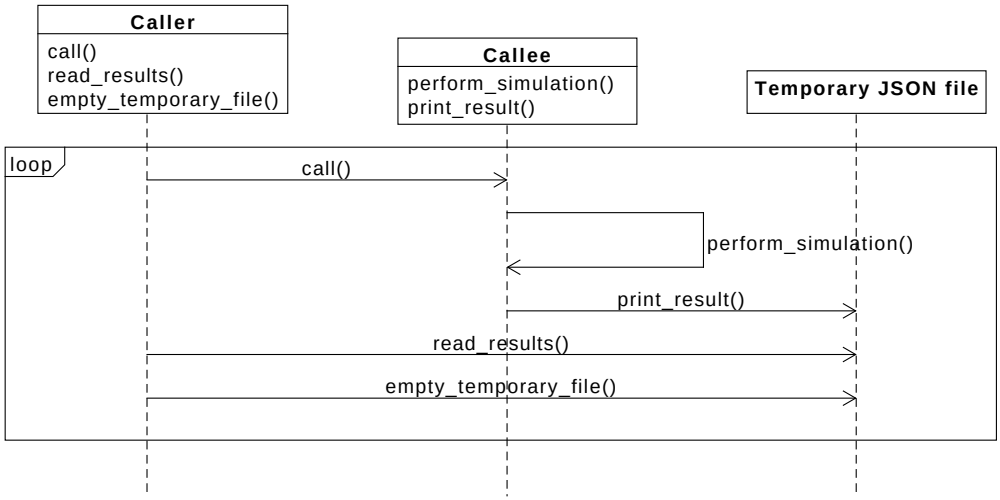


Figure 4.6: UML sequence diagram representation of Pipe Through Temporary Files

4.3.3 Shared Memory

In the *Shared Memory* pattern, data are stored in some memory region accessible to all the sub-models involved. This is the most efficient communication method, as data can be shared between processes without the need for copying or serialization, which can introduce overhead in other communication strategies. However, shared memory is only suitable for communication between processes on the same machine, and shared memory segments need to be properly allocated, deallocated, and managed. Finally, similar to the Pipe Through Temporary Files pattern, some mechanisms are necessary to inform the data consumer when information becomes available, and possible read/write conflicts must be carefully managed to maintain data consistency and integrity.

4.3.4 Rounding Strategies

In a multilevel framework, various components may employ different policies for handling information. As a result, data may need to undergo transformations during the exchange process, especially when transitioning from continuous to discrete values or vice versa. Rounding strategies must be carefully defined to oblige with invariant properties of the model, in order to preserve the global consistency of the representation. Let us consider for example a scenario where a population is split into compartments and a system of ODEs updates periodically the number of agents for each compartment. After the ODE model terminates, one may need to operate again with integer values. Raw discretization can result in unit losses, breaking the invariant of having a stable population. To solve this problem multiple strategies are available.

- *Highest decimals.* Having n compartments and a loss of $1 \leq m < n$ units, the m compartments with the highest decimal number are rounded up. This method, however, is not always applicable, as semantic constraints should be taken into account. For example, in some SEIR model implementations, the number of susceptible people should be strictly decreasing.

- *Designed compartments.* Having a loss of m units, the c compartments (such that $1 \leq m < c$) that are safer or more convenient to increment are identified.

- *Discretization.* If the loss due to the transformation from decimal to integer is not problematic (e.g., when dealing with high values where the loss of a few units is negligible), a raw discretization of the decimal numbers is an acceptable solution.

4.4 Multiscale Patterns

While multiscale models allow us to achieve an optimal trade-off between computational efficiency and accuracy of the representation, the coupling of the various scales also presents some challenges. *Multiscale patterns* are strategies that describe how to switch from one scale to another and how to maintain consistency among the various representations.

4.4.1 Adaptive Resolution

In multiscale modeling, high-resolution sub-models are frequently utilized to describe critical components of the system. The distinction between a more and less important part is often dictated by spatial characteristics, with areas of greater interest detailed at a higher level. However, the concept of criticality can also extend to time, thereby implying the alternation of various models at different levels of resolution.

Adaptive Resolution, depicted in Figure 4.7, is a design pattern that enables the dynamic switch of resolution when specific conditions are fulfilled. These conditions are inherently dependent on the model and they encompass a wide range of cases. Examples include changes in the state variables, the passing of specific thresholds associated with the model's state, temporal constraints, the achievement of specific goals of the simulation, or the availability of computational resources.

An applicative example can be found in [163], where the authors propose an adaptive multiscale framework to model infection propagation. In this work, initially, the simulation starts by employing an agent-based paradigm, to comprehensively capture the initial dynamics of pathogen diffusion. Subsequently, once a specific threshold of infected individuals is reached, the model switches to an equation-based methodology, as there is enough information for the system of equations to work on aggregated measures. This transition facilitates the adoption of a population-averaged approach, optimizing computational efficiency.

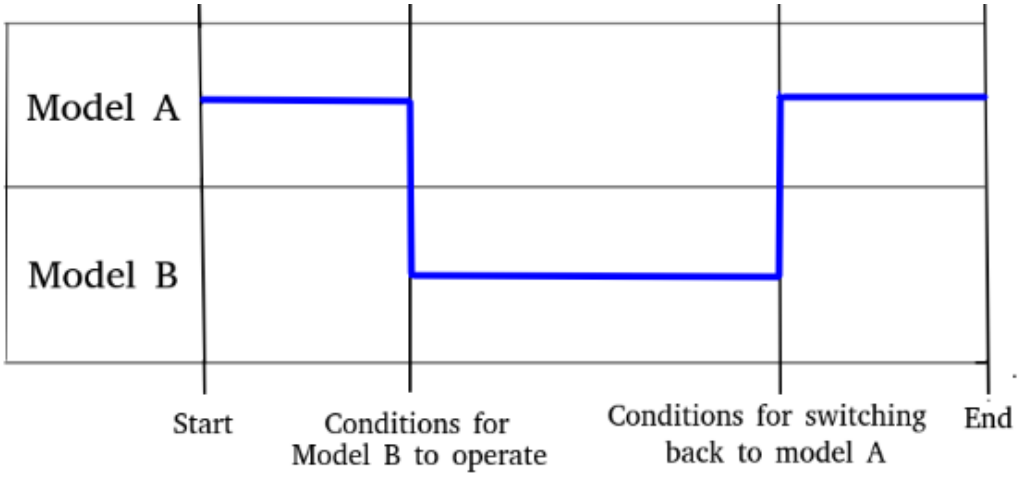


Figure 4.7: Graphical representation of Adaptive Resolution design pattern

4.4.2 Spatial Aggregation-disaggregation

In multiscale frameworks, the notion of scale commonly revolves around spatial resolution, allowing representations at various levels of granularity to coexist within the same virtual environment. A recurring approach involves employing microscopic models like ABMs to individually describe the behavior of entities, alongside macroscopic models like EBMs that handle aggregate representations. The coexistence of different spatial resolutions often requires transitioning from one scale to another, a critical activity that demands careful management. To address this scale transition, the concepts of aggregation and disaggregation become crucial. Aggregation involves collapsing a large number of entities at the micro level to build a single entity at the macro level, while disaggregation is the inverse phenomenon. Different strategies can be employed to establish the transition rules for aggregation and disaggregation. In particular, in [164] the following four design patterns have been proposed. As summarized in Figure 4.8, the choice among these solutions depends on the importance of preserving fine-grained information and how the coupling between the scales is performed.

- *Zoom* pattern, where microscale entities are destroyed when transitioning to coarser-grained representations. This is the simplest way

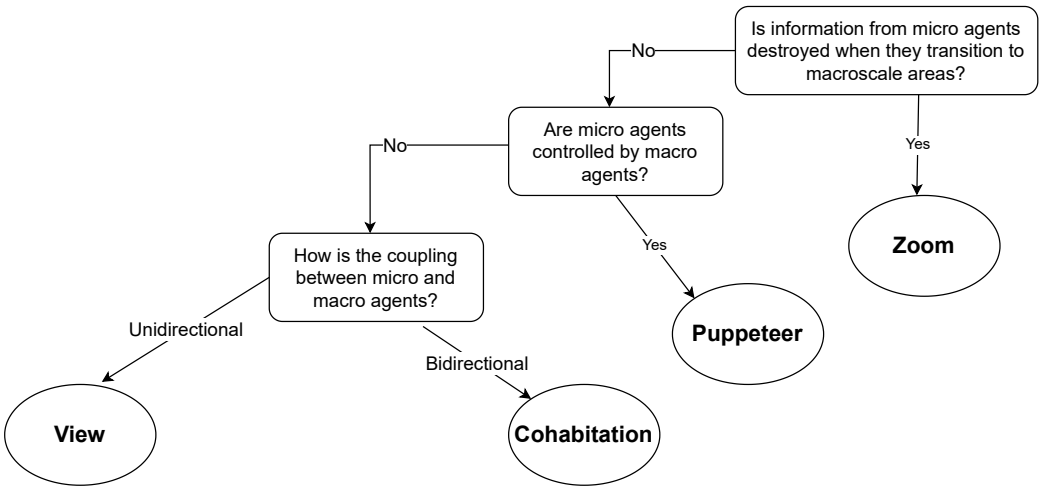


Figure 4.8: Summary scheme of aggregation/disaggregation design patterns

to perform spatial aggregation, but it entails the permanent loss of information.

- *Puppeteer* pattern, where the state of the individual entities is saved when transitioning to the macroscopic scale, avoiding the loss of information. The "frozen" entities will continue to update their state based on their internal dynamics, but they lose the ability to perform actions, which is delegated only to the higher-level agents.

- *View* pattern, where the state of micro entities is computed to reflect the state of the macro entities from which they originate.

- *Cohabitation* pattern, where the link between the source and the target scale is bidirectional so that they influence each other.

Chapter 5

Design Principles for Multilevel M&S

While multilevel modeling emerges in very broad forms in terms of applications, combinations of M&S paradigms employed, and strategies to coordinate the execution of the various components, certain foundational principles that are universal across all types of frameworks can be identified. To formally define these baselines, one approach is to employ a metamodel capable of encompassing all the key procedures for developing a complex model. In the scientific literature, various metamodels have been proposed in the realm of modeling and simulation, although less effort has been specifically devoted to multilevel methodologies. To overcome these limitations, a novel metamodel is presented in this Chapter, with the purpose of providing a comprehensive and generic guideline to assist the development of multilevel simulators.

The remainder of the Chapter is organized as follows. Section 5.1 reviews some metamodels for M&S available in the scientific literature. In Section 5.2 GEMMA is presented. Section 5.3 introduces a Python framework for building GEMMA-compliant models. Finally, Section 5.4 provides an illustrative example to prove the applicability of the proposal.

5.1 Metamodels in the State of Art

In the context of M&S, various metamodels have been proposed by researchers to formally guide model development. Among these, Discrete Event System Specification (DEVS) has emerged as the standard for discrete event models, including versions specifically for multilevel M&S. The goal of DEVS is to enable users to formally articulate how model components transform the internal state in response to events, and generate output based on inputs and the current state.

DEVS atomic models are defined as a tuple [165]:

$$\langle X, Y, S, S_0, ta, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda \rangle$$

where:

- X is the set of all possible inputs;
- Y is the set of all the allowable outputs;
- S is the set of sequential states;
- S_0 is the initial state of the model;
- ta is the time advance function that determines the lifespan of a state;
- δ_{int} is the internal transition function, which defines the rationale for producing local state changes;
- δ_{ext} is the external transition function, which defines how the state evolves based on input events;
- λ is the output function defining which output is produced given a certain state.

To better illustrate the functioning of DEVS, let us consider the modeling scenario of a traffic light. In this use case, the traffic light will alternate between the states of red, yellow, and green until nighttime arrives, after

which it will flash until dawn. When facing a red or yellow light, vehicles will come to a stop, and under other conditions, they will proceed through the intersection. This simple model could be described under DEVS formalism as follows:

- $X = \{NIGHT, DAWN\}$
- $Y = \{MOVE, STOP\}$
- $S = \{GREEN, YELLOW, RED, BLINK\}$
- $S_0 = RED$
- $ta = \begin{cases} GREEN \rightarrow 25sec \\ YELLOW \rightarrow 5sec \\ RED \rightarrow 30sec \end{cases}$
- $\delta_{\text{int}} = \begin{cases} GREEN \implies YELLOW \\ YELLOW \implies RED \\ RED \implies GREEN \end{cases}$
- $\delta_{\text{ext}} = \begin{cases} DAWN \implies RED \\ NIGHT \implies BLINK \end{cases}$
- $\lambda = \begin{cases} GREEN \implies MOVE \\ YELLOW \implies STOP \\ RED \implies STOP \\ BLINK \implies MOVE \end{cases}$

While atomic DEVS can assist the developers in formalizing the behavior of a sub-model, it is not able to describe the interactions among multiple components. To address this limitation, researchers introduced *Coupled DEVS* [166], which is defined as follows:

$$\langle X, Y, D, M_i, EIC, EOC, ID, select \rangle$$

where:

- X is the set of input events;
- Y is the set of output events;
- D is the set of the sub-components that are included in the coupled model;
- $M_i,\ i \in D$ is the list of DEVS components that can be either atomic or coupled;
- EIC is the set of external input couplings, defining mappings between inputs originating from outside the system and input ports of the specific module(s);
- EOC is the set of external output couplings, defining mappings between module outputs and outputs directed outside the system;
- IC is the set of internal couplings, defining mappings between the output of one module and the input of another module;
- *select* is a tiebreaker function that provides a way to resolve temporal ambiguities arising from simultaneous events.

An example of how coupled DEVS should be applied can be found in [167], where the authors simulated a Building Energy Management System by decomposing the system into eight components: Costs, Optimizer, Solver, Regulator, Equipments, Sensor, Hyperconverged infrastructure, and Occupants. The relationships among these sub-models are effectively captured by the meta-model, which facilitates the formalization of interactions among the various components.

Both atomic and coupled DEVS can be further extended to enable parallel execution through Parallel DEVS formalism [168]. In Parallel DEVS an additional element is included in the tuple to handle collisions between simultaneous events, allowing the modelers to define the policies for serialization.

Finally, ML-DEVS is a DEVS extension for multilevel modeling [169] that relies on MICRO-DEVS and MACRO-DEVS models: MICRO-DEVS defines the atomic behavior of a model at the microscale, similarly to atomic DEVS, while MACRO-DEVS describes model interactions, other than the behavior of the models at the macroscale.

In addition to DEVS and its derivatives, various meta-models have been developed for multilevel M&S and agent-based modeling in particular. Some notable examples are:

- *GEAMAS*, an agent-based architecture based on three levels of abstraction, where the microscale agent incorporates the point of view of the actors of the system, the meso level consists of an aggregation of micro agents in a specific context or activity, and the macroscale agents represent the system itself [170].

- *IMR4MLS*, a meta-model centered on the concepts of influence (i.e., agents' decisions based on their current state and the perceived environment) and reaction (i.e., computation of agents' reactions in response to influences). A directed graph is then employed to describe interactions between levels [171].

- *PADAWAN*, a formalism designed to represent systems with multiple environments, each having its space-time scales and rules for the agents that they contain [172].

- *GAMA (GIS Agent-based Modeling Architecture)*, a meta-model for multilevel ABMs, offering three main sets of abstract classes that represent the entities (i.e., individual agents, or aggregated agents), the spatial domain, and temporal aspects within a model [173]. Moreover, GAMA facilitates model specification through GAML (GAma Modeling Language), an agent-oriented language empowering developers to articulate and define the behavior of the system.

Finally, High-Level Architecture (HLA) has emerged as a standard for implementing distributed simulation systems [13]. While not explicitly designed for multilevel modeling, HLA provides a framework that allows multiple simulation systems to interact and exchange information in a coordinated manner through a standardized interface. HLA consists of the following components:

- *Federates*, which are applications that support HLA and are capable of participating in the distributed simulation.
- *Run-Time Infrastructure (RTI)*, a middleware that provides services for the distributed simulation, such as interactions between Federates and synchronization primitives.
- *Federation*, a group of Federates whose interactions are managed by the RTI.
- *Object Models*, descriptions of the essential shareable elements of the simulation.
- *Federate Object Model (FOM)*, a shared specification that defines the structure and the content of the Object Models.
- *Object Model Template (OMT)*, a standard for the creation of an Object Model, defining its data, attributes, and interactions.

While HLA addresses certain challenges in multilevel M&S such as data exchange among sub-models and synchronization, it is not a general-purpose meta-model like DEVS, but rather an API for interoperability of simulators. Therefore, HLA is not suitable for describing general multilevel models.

5.2 GEMMA

The DEVS formalism is well-suited for characterizing the behavior of various types of models, allowing developers to specify both the details of the atomic components and their interactions. However, DEVS lacks a specific

focus on multilevel modeling: the interactions among various elements are described in terms of inputs and outputs exchanged, while other factors such as the hierarchical structuring of system components and the coupling among the meta-models cannot be comprehensively described. To address these concerns, it is necessary to design a novel formalism, explicitly tailored for multilevel modeling. GEneric Multilevel Modeling Abstraction (GEMMA), loosely inspired by DEVS, is an application-independent meta-model that can be employed to describe any type of complex model where multiple components interact. While ML-DEVS refers to scenarios involving multiple scales of detail within the same model, GEMMA is a more versatile solution, as different sub-models can describe distinct and not necessarily related aspects of the same system. The purpose is to assist the developers in the design phase of a multilevel simulator, as the meta-model should help to define a systematic procedure for structuring a multilevel framework and for describing the interaction between sub-models.

GEMMA is defined as follows (as summarized in Figure 5.1):

$$\langle M, T, S, P, C, I, O, L, R \rangle$$

where:

- M is the set of atomic components that are combined in the complex model. The elements of M can be of different types and follow different paradigms, and the specification of their internal behavior is outside the scope of the formalism.

- T is the component that sits at the top of the hierarchical structuring, serving as the interface for the user and providing a starting point for the model execution. It can either be the element of M or a wrapper script in charge of scheduling the execution of the other components.

- S is a tree structure that describes how levels are organized. T is the root of the tree and the other nodes are either elements of M or wrapper scripts that schedule the execution of underlying components. Each

node is responsible for coordinating the execution of its descendants, if any.

- P is the set of model parameters, which can be set by the user to control the execution of the model.
- C is the set of conditions/frequencies that are used by a component a to call another component b. Conditions can either be quantitative (i.e., the call occurs periodically every specified amount of time) or qualitative (i.e., the call occurs after an event, or when a specific value exceeds a certain threshold). The cardinality of C is at least equal to the total number of calls between a and b, but there is no upper bound to the cardinality of C, since there could be multiple conditions for a to invoke b.
- I is the set of inputs to provide when a model is called, a set of inputs for each interaction between models.
- O is the set of outputs of the called models. Similarly to I, there is a set of outputs for each interaction between models.
- L is the set of policies used to maintain consistency, thus preserving the model's invariants. They are particularly important when switching from discrete to continuous representations.
- R Results/Output of the multilevel model: log file or computation of specific metrics that serve as the final outcome of the execution.

5.3 GEMMA-compliant framework

In this section, the structure of the GEMMA metamodel is described in a form that is suitable for implementation in a programming language. In particular, a scheme consisting of a set of abstract classes is presented. This blueprint should serve as a foundational framework for building a multilevel

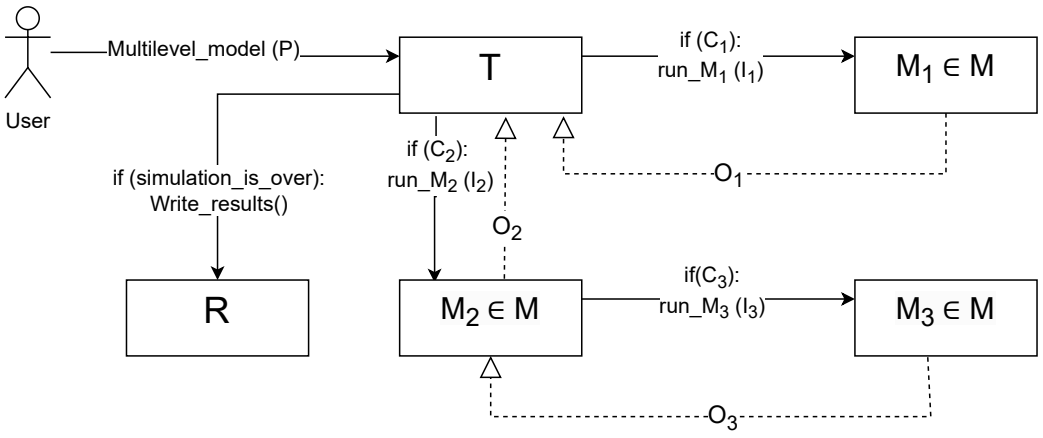


Figure 5.1: Graphical representation of GEMMA formalism

system, ensuring the GEMMA compliance of the complex model. In the proposed scheme, summarized by Figure 5.2, a multilevel model is defined as an instance of the *Composite Pattern* (see Section 4.2.1), as the concrete elements in the hierarchical structure can be classified as either leaves or non-leaves. Leaves implement only Worker functionalities, while non-leaves also encompass Director tasks, therefore supporting the capability to manage the underlying sub-models.

Regardless of the role, each sub-model belonging to the multilevel model is a concrete instance of the abstract class GEMMAComponent. A GEMMA-Component is characterized by a set of parameters of the model, which can be of any kind of type. Furthermore, it must implement the following methods:

- *setup* to execute any model-specific setup procedure.
- *advance* to run the model for a specific amount of simulation time.
- *retrieveResults* to report the metrics of interest computed during the model execution; this method is usually called at the end, but might also be executed periodically to check the evolution of the metrics. When *retrieveResults* is called by the top-level component, then it returns the outcome of the simulation, providing the metrics of the user's interest in a human-understandable format.

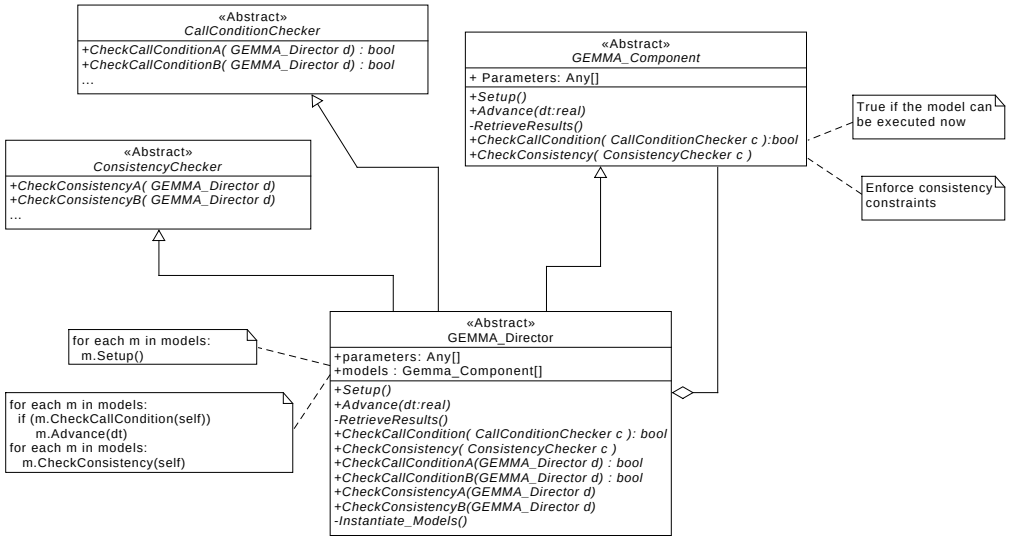


Figure 5.2: UML class diagram of the foundational framework for GEMMA-compliance

- *checkCallConditions* to check the calling conditions of the sub-model, according to the rationale of *Visitor Pattern*.

- *checkConsistency* to check the consistency of the retrieved results retrieved by the sub-model, according to the rationale of *Visitor Pattern*.

On the other hand, a GEMMADirector is a specific type of GEMMAComponent, which implements further functionalities to act as a Director. It is worth noting that a Director can also act as a Worker; in fact, only the top-level component will have exclusive Director functionalities. First, a GEMMADirector contains a set of underlying Workers, which are of type GEMMAComponent. Furthermore, it implements the additional methods:

- *instantiateModels* to create the instances of the Workers that are handled by the director.

- set of methods inherited by CallConditionsChecker to apply director-level policies for the management of the calls to underlying models.

- set of methods inherited by ConsistencyChecker to apply director-level

policies to maintain the global consistency of the system, ensuring that the model's invariants are maintained.

Finally, abstract classes CallConditionsChecker and ConsistencyChecker contain the actual methods to enforce call conditions and consistency constraints. One of the benefits of multilevel models is the possibility of reusing existing models; however, they are unlikely to be compliant with the GEMMA metamodel, so Workers might employ the *Adapter* pattern to make the concrete model to be exposed with the appropriate interface.

5.4 Illustrative Example

To demonstrate how GEMMA should be applied, an illustrative use case is provided. It is noteworthy to emphasize that the model presented below is not designed for precision; instead, its purpose is to showcase the feasibility of the proposed methodology.

Use Case Overview The use case consists of a local pollution model whose simulation space is populated by multiple types of vehicles, each characterized by different pollutant emission rates. The model illustrates how vehicles contribute to environmental pollution and demonstrates that switching to more ecological fuels results in a cleaner environment over time. In the use case, vehicles are either petrol-based, powered by LPG (Liquefied Petroleum Gas), or electric. Petrol-powered cars are the most polluting, while electric vehicles produce no local emissions, but they still contribute to the overall contamination as the environmental cost of electricity production is not negligible.

The multilevel model is composed of three distinct modules: two submodels describe vehicle movements and the process of vehicle transition, while a top-level wrapper script orchestrates the execution of the underlying components. The mobility model is a discrete CA where each cell is associated with a value that represents the concentration of pollutants in that area. At

each time-step, vehicles move toward a random direction, emitting locally a certain amount of pollutants depending on the fuel employed. Contamination spreads to adjacent cells and gradually diminishes over time implying that, in the absence of vehicles in the system, all cells would eventually be free of pollution. On the other hand, the vehicles-change model is a compartmental model that describes through a set of ODE the impact of incentives to replace a vehicle with a less pollutant one. Specifically, let $P(t), L(t), E(t)$ be the number of petrol, LPG, and electric cars, respectively. It is assumed that the total number N of vehicles remains constant and that the number of vehicles in each class at a time obeys the following set of differential equations:

$$\begin{cases} \dfrac{dP}{dt} = (-\beta - \sigma)P \\ \dfrac{dL}{dt} = \beta P - \gamma L \\ \dfrac{dE}{dt} = \sigma P + \gamma L \end{cases} \tag{5.1}$$

subject to initial conditions $P(0) = P_0$, $L(0) = L_0$, and $E(0) = E_0$. β, σ and γ represent respectively the transition rate between different compartments, specifically from P to L (β), from P to E (σ) and from L to E (γ).

In the setup phase of the simulation, the population of the model composed of N petrol-based vehicles is displaced randomly across the grid. When the actual execution starts, the CA handles mobility and pollutant emissions, and the compartmental model is then called at regular intervals, updating the number of each type of vehicle in circulation. The parameters provided to the ODE model (5.1) are the number of vehicles for each type, and a coefficient that has to be applied to β, σ and γ to re-scale their values. Such a coefficient is determined dynamically and is proportional to the pollution level recorded in the environment. The rationale is that the incentives for switching towards more ecological vehicles could be higher when contamination intensifies.

Implementation The mobility model was implemented using NetLogo [174], an integrated development environment for agent-based modeling that pro-

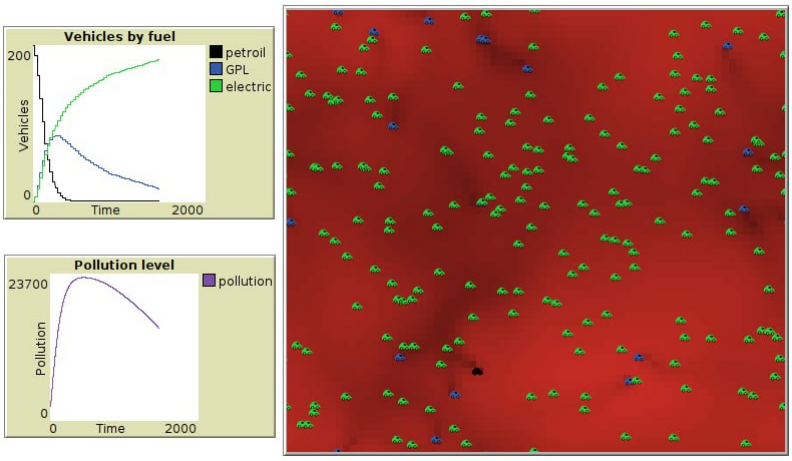


Figure 5.3: NetLogo graphical interface of the pollution model after 1500 time-steps.

vides a multi-agent programming language and graphical interface that allows users to observe the evolution of the system. A screenshot of the running system is shown in Figure 5.3. On the other hand, the compartmental model was developed in Python and its core is the set of ODEs that defines the number of cars for each category. The model is executed periodically after the CA has performed a defined number of steps, in order to update the composition of the population. Finally, a wrapper script is used as a top-level component to schedule the execution of both the sub-models and to manage the exchange of data. This was necessary since the NetLogo language does not natively support interaction with external programs. These components are integrated within the framework discussed above, as shown in Figure 5.4.

According to the GEMMA interaction scheme, the model can be defined as follows:

- M. A compartmental model a describing the temporal evolution of the composition of vehicles and a NetLogo model b that describes mobility and the diffusion of pollutants.

- T. A wrapper script w at the top of the hierarchy that is in charge of orchestrating the execution of the various modeling instances and also serves as the interface for the user.

- S. w is the root of the tree; a and b are the children of w.
- P. Total number of vehicles, set of parameters used by a to describe how vehicle owners upgrade their vehicle (β, γ, σ from (5.1)), call frequency of sub-model a.
- C. Execution starts by running sub-model a; then, sub-model b is invoked periodically.
- I. w initializes b with the total number of agents; model a instantiated with the number of vehicles for each compartment up to date.
- O. The output of a is used to update the number of vehicles for each compartment in b. In turn, the output of b provides the input conditions for model a.
- L. Loss due to the switch from continuous values (the number of vehicles for each compartment is a continuous value) to discrete values (cellular automaton model, each vehicle is an agent).
- R. NetLogo plots representing the diffusion of pollutants through time.

Technical Details Despite the replicability of the proposed framework, inherent features of the programming language must be considered in the development phase. For instance, in Python, the Java-style distinction between abstract classes and interfaces does not exist. By default, Python does not offer abstract classes; instead, it includes a module called ABC (i.e. Abstract Base Classes) and utilizes the "@abstractmethod" decorator to declare a method as abstract, allowing also for multiple inheritance [175]. Following this modus operandi, an abstract class like GEMMAComponent is structured as depicted in Listing 5.1, where methods are declared without implementation, akin to the behavior of an interface.

The abstract classes CallConditionsChecker and ConsistencyChecker delineate the essential operations for overseeing call conditions and ensuring

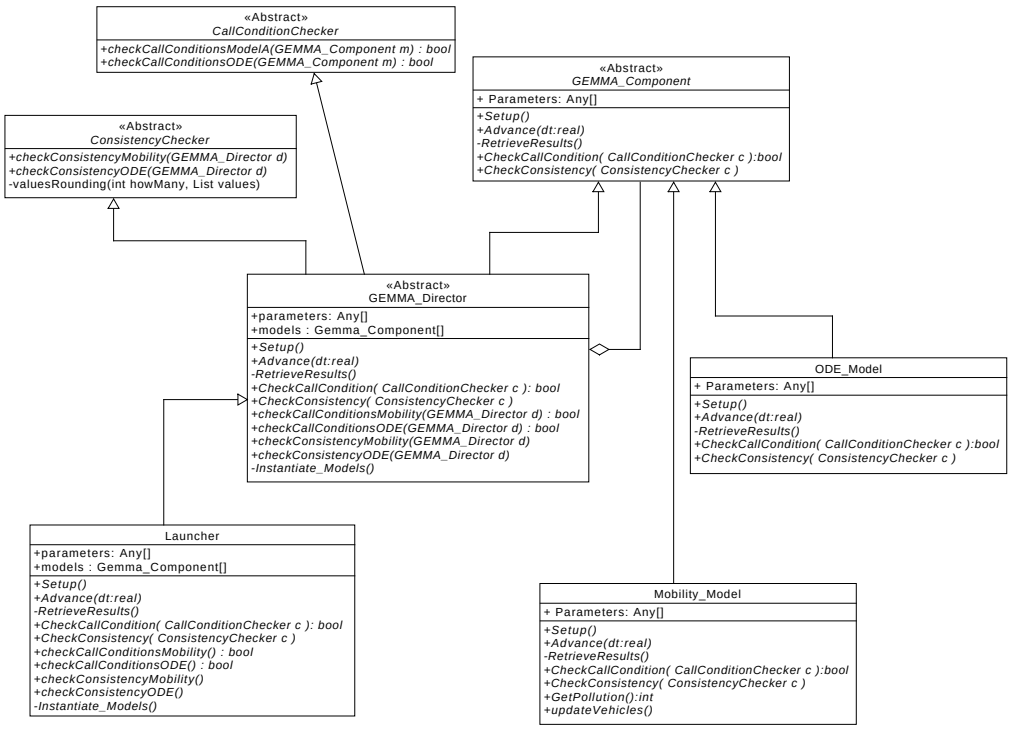


Figure 5.4: Class Diagram of the pollutants model

output consistency, with each Worker in the system having a corresponding method. While nothing has to be done to ensure consistency for the mobility model, the values retrieved by the ODE model are adjusted according to the approach described in Listing 5.2.

The adopted strategy for maintaining consistency is to round up the values of the l elements with the greatest fractional part, where $l < \#compartments$ corresponds to the loss units. This approach guarantees that the population remains constant while obtaining a discrete approximation of the compartments that is as close as possible to the original continuous values.

The framework, despite being based upon a set of Python abstract classes, is easy to integrate with models written with different languages by means of an adapter or through encapsulation. In the considered scenario, the mobility model, written in NetLogo's native language, is encapsulated within a Python class. Specifically, as shown in Listing 5.3, we make use of *pynetlogo*

Listing 5.1: Code snippet of the GEMMAComponent abstract class

```
class GEMMA_Component (ABC):
    def __init__(self):
        self.parameters = {}

    @abstractmethod
    def setup (self, *args):
        #Set up the model
        pass

    @abstractmethod
    def advance (self, *args):
        #Advance with the simulation
        pass
```

Listing 5.2: Python code snippet to manage consistency.

```
def valuesRounding(howMany, valueList):
    returnList = [int(elem) for elem in valueList]
    tempList = [elem - int(elem) for elem in valueList]
    for i in range (howMany):
        max_value = max(tempList)
        max_index = tempList.index(max_value)
        tempList[max_index] = -1
        returnList[max_index] += 1
    return returnList
```

library to facilitate the interactions between Python and Netlogo. Upon instantiation, the mobility model is set up by the launcher, which provides some high-level parameters of the complex model; subsequently, the launcher executes step-by-step the mobility model.

Listing 5.3: Portion of the Mobility model

```
import pyNetLogo

class MobilityModel(GEMMA_Component):

    def __init__(self, modelPath, NetLogoPath, version):
        self.netlogo = pyNetLogo.NetLogoLink(gui=True,
        netlogo_home=NetLogoPath, netlogo_version=version)
        self.netlogo.load_model(modelPath)

    def setup (self, population):
        self.netlogo.command('setup ' + str(population))

    def advance(self):
        self.netlogo.command('go')

    def retrieve_results (self, *args):
        pass
```

Part III

Applications

Chapter 6

Simulation of multi-agents systems

In this chapter, three simulation studies are presented. All of them are characterized by the use of a pure ABM paradigm, as the semantics of the scenarios under examination did not require the inclusion of multiple levels. However, it is worth noting that agent-based models often serve as essential building blocks in multilevel modeling frameworks, as they accurately capture dynamics involving multiple, potentially heterogeneous entities.

The following experiments are conducted using LUNES (Large Unstructured Network Simulator), a time-stepped discrete-event simulator that serves as a key component in the multilevel simulators introduced in Chapter 7. LUNES is built on the ARTÌS/GAIA simulation middleware, which provides communication primitives, time management, and support for parallel and distributed execution. One of LUNES' core strengths is its scalability, enabling simulations with over 10 000 entities on a single machine. Nodes are assigned integer IDs and other state variables, supporting the modeling of multilayer and temporal graphs. LUNES relies on two critical functions: one executes required actions for all nodes at each time-step, while the other is triggered whenever a message is received.

The remainder of the Chapter is organized as follows. In Section 6.1 a

data streaming scenario within a peer-to-peer environment is simulated using temporal networks; Section 6.2 presents the simulation of an opportunistic mobile network exploiting public transportation; finally Section 6.3 analyzes a hybrid P2P-edge computing architecture.

6.1 Peer-to-Peer Environments

A part of the work presented in this chapter has been published in [176] and is reported here for the reader's convenience.

6.1.1 Background

The traditional way to store and retrieve information through the Internet is to rely on a client-server setup where data are stored in remote servers, typically on centralized data centers. However, decentralized alternatives exist. Edge computing, for instance, aims to bring computation closer to end-users by distributing lightweight server replicas across multiple geographic locations. This procedure reduces latency and minimizes single points of failure.

An alternative for content sharing is to use a peer-to-peer approach, where each node in the network actively participates in the system's operations. No central authority is required, although one may be present to oversee participant management and maintain content ownership records. Peer-to-peer systems are generally built on top of an existing physical network, such as the Internet [177], with the overlay network structured as a graph. In this graph, nodes represent peers, and edges connect neighboring nodes that are in direct contact. To accurately model distributed environments with churn (i.e. flow of nodes frequently joining and leaving), temporal graphs are especially useful. These graphs represent interactions among entities over time, differing from traditional graphs by allowing nodes and edges to evolve as time progresses. This adaptability enables temporal graphs to effectively capture the dynamic nature of temporary connections and interactions [178]. In

peer-to-peer systems, participants frequently disconnect within a few minutes, and it is rare for nodes to remain connected for more than a few hours. The high churn rate presents a significant challenge for overlay management, as compared to static networks. To address this, robust policies are required to prevent issues such as node isolation, where a node loses all its neighbors before it can establish new connections, which could impair overall network functionality.

Overlay networks in peer-to-peer (P2P) systems are typically classified into two categories:

- *Tree-based Networks*, where the various participants are organized in a tree structure. The source node, which originates the dissemination, serves as the root of the tree, while neighbors are child nodes. Messages propagate from the source node to the leaves in a top-down manner, optimizing the number of forwards. However, tree construction and management can be costly due to churn, as the tree structure breaks when non-leaf peers exit, requiring a new logical tree for each data source.

- *Mesh-based Networks*, where participants form a randomly connected overlay, maintaining a certain number of incoming and outgoing connections. Upon joining, each peer connects to available peers via a tracker or bootstrap node. Since source and destination nodes usually are not directly connected and they are unaware of the system's overall structure, messages are relayed to multiple other peers until reaching the destination. To this purpose, a gossip algorithm is employed to dictate the policy for dissemination.

In a P2P environment, for scalability reasons, the nodes are directly in touch only with a bunch of peers, and they do not know the location of the other nodes. Thus, in the communication process, the information is relayed multiple times among the participants of the system (i.e. multi-hop), until the final destination is reached. To define the policy for message dissemination

in a P2P environment a gossip protocol is employed. Different types of algorithms can be implemented depending on the semantics of the system. In some networks achieving a very high coverage (i.e. percentage of peers who eventually receive the message) and a low delay (i.e. average time for a peer to receive a message) is fundamental, while other ones may be focused on traffic minimization or retention of anonymity.

Some of the most important gossip algorithms are:

- *Probabilistic Broadcast (PB)*, where each node has a probability p of forwarding the message to all its neighbors (except the forwarder) and a probability $1 - p$ of not forwarding it to any other node.
- *Fixed Probability (FP)*, where each neighbor (except the forwarder) has a $p\%$ chance of receiving the message, determined by an individual roll of the dice.
- *Degree-Dependent Functions (DDP)*, where a message is forwarded based on the number of neighbors of the potential receiver: the fewer connections a neighbor has, the higher the probability it will receive the message.

All these algorithms are characterized by a time-to-live parameter, which defines the maximum number of hops a message can take, thereby preventing infinite loops.

The employment of peer-to-peer applications can bring various benefits, such as avoiding single points of failure, the ability to download from multiple sources simultaneously, and the promotion of self-sustaining systems. However, potential drawbacks like network traffic congestion should also be considered. Dissemination of messages can result in a significantly higher amount of network traffic compared to client-server applications, requiring a balance between delivery speed and network traffic. Additionally, the dynamic nature of peer-to-peer networks, with participants frequently joining and leaving, further complicates message delivery by causing continuous fluctuations in network structure.

6.1.2 Simulation Scenario

Let us consider a hybrid push-pull protocol that combines both mesh-based and tree-based approaches for data dissemination. The protocol utilizes two separate overlay levels: a mesh-based overlay for pulling stream requests and a tree-based overlay for pushing data chunks. This approach enhances system decentralization by eliminating the need for servers or special participants when discovering which peers own certain resources. A network participant can have three possible roles:

- *Streamer*, which is the node that generates the data stream.
- *Downloader*, if the node is interested in receiving the data stream.
- *Non-Downloader*, if the node just contributes to the dissemination of messages.

The protocol operates as follows. When a Downloader node A wants to join a data stream, it disseminates a Request message. Since the IP address of the streamer is unknown, a pull-based gossip algorithm is used to contact the streamer. Once the streamer S receives a request from node A, it adds A to the stream tree. Assuming each node can relay data to at most n other peers (with n being the maximum degree for the nodes) if S has fewer than n children, it adds A as a child. Otherwise, S randomly selects one of its children to place A in the streaming tree. This procedure continues recursively until A becomes a leaf of the tree. After a short delay (needed for tree construction), the streamer begins emitting data. Other nodes can dynamically join the streaming tree at any point. If a non-leaf node in the streaming tree becomes inactive, its children will resend the request message through the overlay to rejoin the streaming tree upon detecting the failure.

The interactions among the nodes are based on the exchange of five types of messages:

- *Request* messages, which are sent by downloaders when joining the data stream. They are disseminated using a gossip algorithm.

- *Item* messages, which contain chunks generated by the streamer and requested by downloaders. They are sent through the streaming tree from parent to children.
- *Tree* messages, which are originated by the streamer upon receiving a request message. They are forwarded down the tree to add the downloader as a leaf.
- *Ping* messages, which are sent periodically to neighbors to check if they are still alive.
- *Pong* messages, which are sent in response to Ping messages.

This strategy aims to minimize both latency and network traffic, which are major concerns in peer-to-peer environments. The only messages disseminated throughout the network are the request messages, which contain minimal data (i.e., information about the node and the stream to join). Instead, the chunks, which constitute the main data load, are forwarded exactly once for each stream participant, employing at most h relays to reach a participant node, where h is the height of the unbalanced tree. Typically, the chunks constitute the majority of the communication overhead in such systems.

6.1.3 Simulation Setup

In the experiments, time is segmented into epochs, each characterized by a single data source, with one active node elected as the streamer at the beginning of each epoch. During each epoch, the streamer emits 470 chunks, one per time-step. The system comprises 10 000 simulated entities, with an average of 80% being active, each having approximately 10 connections in the mesh-based overlay. Among the active nodes, 5% functions as downloaders, and those transitioning from inactive to active during an epoch have an equal chance of assuming the streamer role. Additionally, the streamer waits for 10 time-steps at the epoch's start before emitting chunks to allow downloaders to organize into the streaming tree.

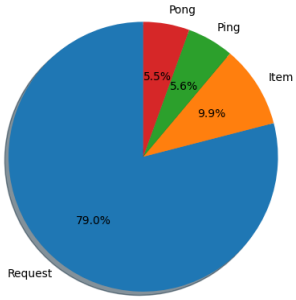
In the experiments, all the chunks are 1MB large, similar to real peer-to-peer applications, where typically chunk size ranges from some hundreds of KBs to 1 MB [179]. Excluding chunks, all the other messages carry very little information, so it is assumed that their size in terms of bytes is equivalent to the minimum Ethernet frame size, which is 72 bytes (also considering the 8 bytes for the preamble) [180], while the 12 bytes for the interframe gap are not necessary in this case, since all the small messages (i.e. the non-item messages) are sent atomically.

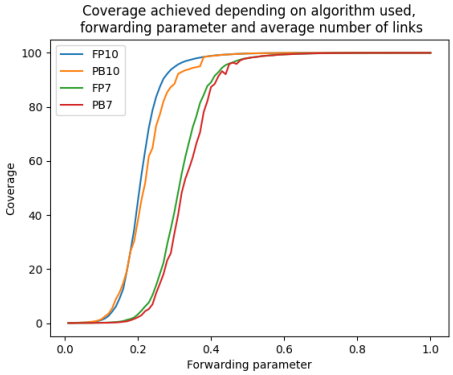
Several parameters influence the simulation outcome, including the dissemination protocol for request messages, network temporariness (activation/deactivation of nodes to maintain stability), and node degree (affecting tree height and average latency). Churn, or nodes leaving the network, significantly impacts tree reconstructions and overall system performance.

6.1.4 Simulation Results

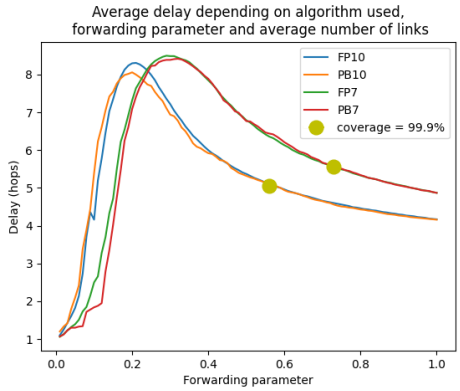
The choice of dissemination protocol can significantly impact network traffic. In the evaluated framework, only request messages are disseminated through the network using a gossip algorithm, while chunks are delivered via the streaming tree. Consequently, the number of forwarded items is proportional to the number of chunks to be sent, while the volume of forwarded request messages depends on both the number of downloaders and the overall active participants in the system. Additionally, an increased churn rate can exacerbate network traffic, as frequent tree reconstructions necessitate that downloaders rejoin the stream, leading to additional overhead.

Figure 6.1 shows the outcomes of various metrics based on the average number of neighbors per node, comparing the Fixed Probability and Probabilistic Broadcast dissemination protocols. When the average number of connections per node and the forwarding parameter of the gossip algorithm are high, coverage and delay improve, but at the cost of increased network traffic.





(a) Less than 10% of the network traffic is actually used for sending the requested items.



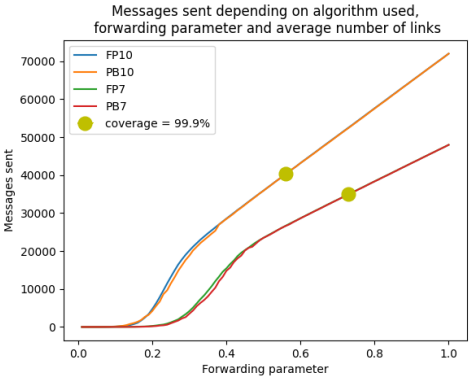
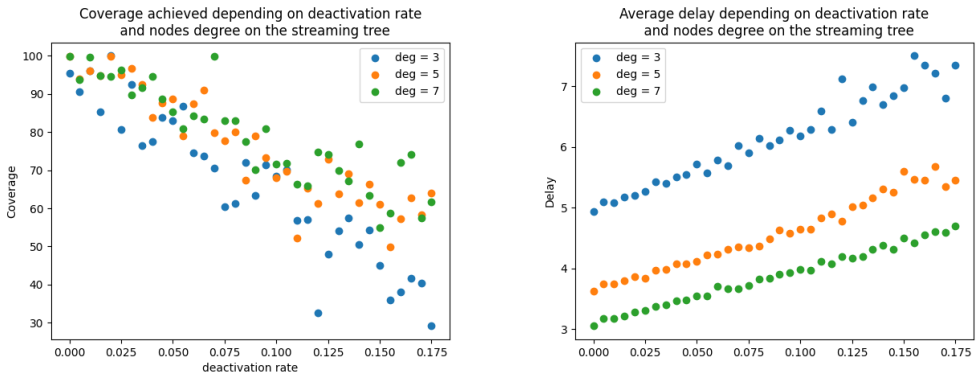


Figure 6.1: The marker indicates the minimum forwarding value to achieve a coverage of 99.9%. FPx = Fixed Probability having x as the average degree. PBx = Probabilistic Broadcast having x as the average degree.

In a scenario with 2000 simulated entities, using Fixed Probability with a 0.6 as a forwarding parameter (the minimum for full coverage), network traffic is evenly split between chunks and overhead. However, with 10 000 nodes, the impact of request messages significantly increases, constituting almost 90% of the network traffic. Introducing a significant deactivation rate (0.1% in these experiments) alters the message weight proportion, as shown in Figure 6.1a. In this configuration, a ping-pong of messages occurs every 20 time-steps. Reducing the ping-pong frequency can lower network traffic but it worsens chunk reception rates. Undetected node failures can

cause significant issues for both the overlay and the streaming tree. As the churn rate increases, detecting failures quickly becomes more critical. Thus, adjusting the ping-pong frequency based on the node deactivation rate is an advisable strategy.

The deactivation rate significantly influences the outcome of the tests, as failures or exits of non-leaf downloaders lead to tree reconstruction, causing other Downloaders to lose the stream for the amount of time needed to detect the failure and rejoin the tree. In both overlays, a lower degree increases the importance of quickly detecting peer disconnections. This is crucial for maintaining an adequate number of links, which allows nodes to effectively propagate request messages through the mesh overlay and efficiently distribute chunks within the streaming tree. Forwarding chunks to a deactivated node wastes bandwidth, and undetected exits can reduce the degree of the streaming tree, compromising propagation efficiency. Downloaders can infer that their parent node has exited the network if they stop receiving messages after a certain period. Conversely, detecting the departure of children or common neighbors in the mesh overlay requires an active strategy, such as implementing a ping-pong messaging system between neighbors at regular intervals to ensure participants are still alive and online.



(a) (b)

Figure 6.2: Average delay and chunk reception rate worsen as the deactivation rate increases.

In the proposed dissemination scheme, the delay in chunk delivery is proportional to the height of the streaming tree, or more accurately, to the average depth of the Downloaders. This is because, on average, more hops are necessary to deliver a chunk. Therefore, the maximum degree of the nodes is a crucial parameter that can significantly influence the outcome. In a real-world scenario, the number of children for each Downloader would depend on its upload bandwidth, and it would be convenient to place nodes with higher bandwidth availability at the top of the tree to minimize its height. However, to reduce the complexity of the simulation scenario, all nodes are assumed to have similar bandwidth availability in the experiments and are characterized by a parameter d indicating the node's degree. While a balanced tree strategy could minimize tree height, the overhead required to maintain the balance might outweigh the benefits.

Figures 6.2a and 6.2b show the interdependence between d and both delay and chunk delivery rate. When the parameter d is high, the average depth of the downloaders in the streaming tree decreases, and more nodes in the streaming tree are leaves. This leads to fewer tree reconstruction events, as the number of requests for re-joining the streaming tree following a node failure is proportional to the number of descendants of the node that failed. On the other hand, the delay is inversely proportional to the average height of the steaming tree, thus d is another relevant factor.

6.2 Data Mules and Smart Territories

A part of the work presented in this chapter has been published in [181] and is reported here for the reader's convenience.

6.2.1 Background

Data Mules (an acronym for Mobile Ubiquitous LAN Extensions) are a technology designed to provide digital communication in areas without direct Internet connectivity. This technology offers a specific type of solution

for opportunistic networking. Data Mules are mobile devices equipped with storage and wireless communication mediums, such as Wi-Fi, Bluetooth, or LoRa, allowing them to exchange data with nearby static sensors or access points they encounter [182]. As they move between remote areas, they create effective data communication links. Given that device movement is crucial for message delivery, Data Mules are suitable for delay-tolerant services.

Data Mules facilitate communication and data transfer in the absence of the Internet, making them vital tools for IoT applications. They are commonly used in services based in smart cities or villages, particularly where significant data flows originate from remote areas [183]. Depending on the context, Mules can be vehicles like buses or cars, or even walking persons.

6.2.2 Simulation Scenario

In the scenario under consideration, it is assumed that a set of Clients have data to communicate, despite a lack of Internet connectivity in the simulation area. The messages need to reach a special node, the Proxy, which is responsible for delivering (online) the message to the final recipient. To contact the Proxy, the nodes can exploit the presence of Data Mules, such as public transportation vehicles. Furthermore, to incentivize collaboration among Clients and incorporate opportunistic networking typical of smart territories, Clients are allowed to relay messages within other nodes in proximity. All communications are performed through Wi-Fi, considering the Wi-Fi link rate at $12Mbps$ and Mules' velocity at $36km/h$, as in [184]. Through simulation, it is possible to analyze the delays (and their specific composition) ranging from the creation of a message by a Client up to its delivery to the Proxy.

6.2.3 Simulation Setup

To conduct the simulations, a square discrete space consisting of 1000 x 1000 cells was used as a testbed. This area represents a single village populated by several Clients and equipped with Couriers and transport

service, as illustrated in Figure 6.3. Each cell in the grid represents a $20m$ x $20m$ area, potentially containing one or more Client nodes. The total area measures $20km$ x $20km$, with a node density of 25 nodes per km^2.

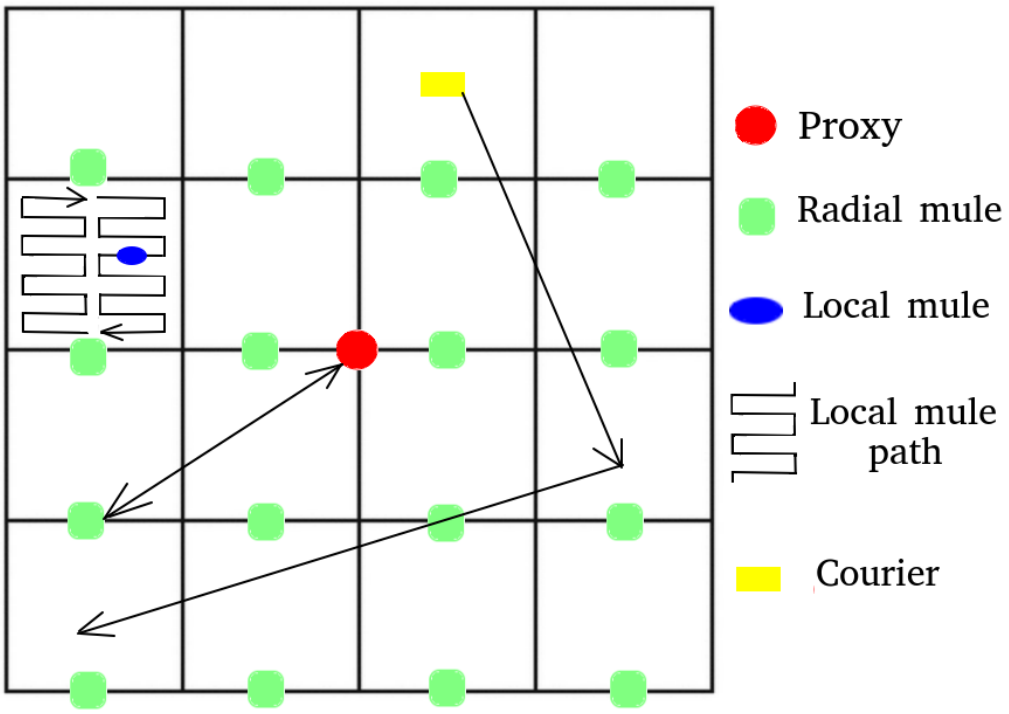


Figure 6.3: Representation of the simulated area. The grid is divided into 16 regions. In addition to the Client nodes scattered throughout the area, there are 16 Local Mules, 16 Radial Mules, and the Proxy located at the center of the grid.

Some of the couriers and buses in the transport service also function as Mules, covering both central and peripheral regions. Consequently, the grid was divided into N^2 square regions.

In the model there are five types of simulated entities:

- *Proxy node*, which is situated at the center of the grid and is the final destination for all the messages.
- *Client nodes*, which have a certain probability of generating new messages at each time-step. In the experiments, 10 000 Clients are placed

within the simulated area, distributed randomly either with a homogeneous or centralized pattern. In the homogeneous distribution, nodes are scattered randomly across the grid. Conversely, in the centralized distribution, the likelihood of a cell hosting nodes decreases with its distance from the center, in order to reproduce a village-like scenario, where usually most individuals reside near the center, with peripheral areas being less densely populated.

- *Local Mules*, which traverses a specific region of the grid in a zigzag pattern, collecting messages from nearby Client nodes as they move. After completing a lap, the Local Mules deliver the messages to the Radial Mule or directly to the Proxy if it is nearby, before starting their route again.

- *Radial Mules*, which gather messages from the designated Local Mule and transport them to the Proxy. After delivering the messages, the Radial Mules return to their initial positions, ready to receive messages from the Local Mule's next lap. Each Local Mule is paired with a corresponding Radial Mule for this task.

- *Couriers*, which move according to the Random Waypoint mobility model, i.e. they are either still or in motion, and when they activate they pick a random point, moving towards it with the same speed as buses. Couriers gather messages to carry them until a reachable mule is found.

Interactions are proximity-based, allowing Clients to deliver messages within a communication range of 200m to Mules or directly to P. Mules move at an average speed of 36 km/h, advancing to adjacent cells in discrete time-steps representing 2 seconds. Additionally, messages can be delivered directly to P or the Radial Mule if sufficiently close for direct communication.

		Avg Delay ± *Std* (seconds)		**Coverage**	
		RC	*NOR*	*RC*	*NOR*
CUR	HOM	2 276 ± *1 202*	3 106 ± *2 324*	98.3%	39.2%
	CENT	1 500 ± *1 197*	2 340 ± *2 311*	96.7%	44.7%
NOC	HOM	2 995 ± *2 557*	3 101 ± *2 320*	61.1%	39.2%
	CENT	2 137 ± *2 415*	2 335 ± *2 298*	71.3%	44.7%

Table 6.1: Average delay & standard deviation (i.e. Std) in seconds. Population is HOM = homogeneous or CEN = centralized. Relay among Clients = RC, or not = NOR. Couriers are present = CUR, or not = NOC.

6.2.4 Simulation Results

Several tests were conducted to measure message delivery delay and coverage (i.e., the percentage of Client nodes able to send messages to a Mule) by varying: (i) the number of couriers, (ii) the feasibility of relaying between clients, and (iii) the population distribution. The experiments included 16 Local Mules, 16 Radial Mules, and, if applicable, 16 Couriers, over 30 000 time-steps, allowing Local Mules to complete their routes several times.

Table 6.1 shows the metrics retrieved from the tests. As expected, a centralized population significantly reduces the average delay because most nodes are near the center, making it easier to directly connect with the Proxy or meet one of the Radial Mules on the way to the center.

The feasibility of relaying between Clients and the presence of Couriers positively impacts coverage, substantially increasing the percentage of nodes reachable by the Mules. Also, these two factors help to reduce the average delay, as shown in the heat maps in Figure 6.4.

Finally, a centralized population generally achieves higher coverage, as central nodes are easier to contact. With a centralized population, areas at the grid's edge lack connectivity despite the feasibility of relaying among Clients. However, this issue is mitigated by employing couriers, which increases coverage from 71.3% to 96.7% for centralized distribution and from 61.1% to

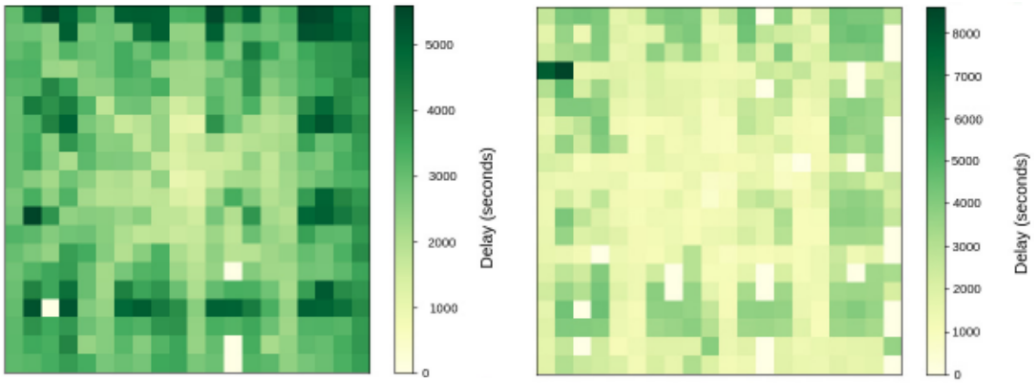


Figure 6.4: Heat map representing the average delay in a scenario with a homogeneous population. The left map does not account for the presence of Couriers or the ability for Clients to relay messages. In contrast, the right map includes both of these factors.

98.3% for homogeneous distribution.

6.3 Edge Computing

A part of the work presented in this chapter has been published in [185] and is reported here for the reader's convenience.

6.3.1 Background

Edge computing is a paradigm that aims to bring computation closer to end-users by deploying lightweight servers, known as edge nodes, geographically close to users. This decentralization of content storage and delivery offers several benefits, such as:

- *Reduced latency*, end-user devices connect to the nearest server, decreasing communication delay and making real-time execution closer to accomplishment.
- *Reducing data center workload*, leading to more sustainable energy consumption.

- *No single point of failure*, the content remains available even in the event of certain data servers being unavailable for malfunction or maintenance.

There have already been studies and proposals to combine P2P and edge computing paradigms, for example in [186] a P2P communication approach among the edge nodes has been proposed. Other works, such as [187] highlighted some similarity in management and structuring between P2P and edge computing, while in [188] spatial modeling was used to investigate computing and communication latencies in an edge computing environment.

Dissemination As introduced in Section 6.1, in a P2P environment nodes are connected only to a few peers and rely on multi-hop communication to relay information across the network. In wireless communication, further constraints have to be considered, as the signal propagates in the air only within a certain range. In particular, in this work the following algorithms are considered:

- *Pure broadcast* - The message is forwarded to all the neighbors, except the forwarder. In this way, we achieve the theoretical minimum time delivery and the maximum coverage, at the cost of a high amount of network traffic.

- *Probabilistic Broadcast* - Given a forwarding parameter p, there is $p\%$ of chance that a node forwards the message to all the neighbors and $(100 - p)\%$ that it does not send it to any other node.

- *Reduced Range* - Since in wireless communication a signal is spread through air, then it is not possible to arbitrarily deliver the content to a limited set of receivers. Therefore, an alternative for traffic minimization is to reduce the power of the signal, thus reaching a lower number of peers. By adjusting a parameter p, the signal is transmitted at only $p\%$ of its normal power range d, thus reaching on average $(d-p)^2/d^2$ of the nodes with respect to the standard configuration.

- *Directed Propagation* - Following the same principle of Reduced Range, the purpose is to reach fewer nodes with the propagation of the signal. In particular, the signal is propagated only in certain directions. In this case, some geographical information about the environment can be exploited.

Mobility Algorithm The typical IoT deployments include applications for both static and mobile devices. Devices emitting signals can either be stationary (e.g., a household appliance) or move to different geographical locations over time (e.g., smartphones, cars, drones). Various approaches can be employed to simulate these movements, considering the human behaviors that drive such changes.

- *Static model* - The end-nodes (i.e. devices) are randomly positioned within the grid and remain stationary for the entire duration of the tests.

- *Random independent movements model* - At each time-step the nodes have a probability p to move into an adjacent cell and probability $1 - p$ to stay still.

- *Random Waypoint model* - In this widely used movement model, a node is either stationary or in motion toward a certain location. Stationary nodes have a probability of activating and choosing a random location on the grid as their destination. Once a destination is selected, the node starts moving towards that point at a given speed [189]. In the proposed model, since the simulation steps represent small time units, nodes only move to an adjacent cell in one time-step.

- *Community-based model*, which represents groups of individuals behaving similarly [190]. When a stationary node n activates by choosing a destination point, nearby stationary nodes will also move towards the same destination cell.

6.3.2 Simulation Scenario

This study investigates a hybrid edge-P2P system where end nodes attempt to connect with an edge node by relaying their requests through other nodes until a destination point (i.e., one of the edge nodes) is reached. In this system, nodes are arranged in a grid, and each participant is associated with a geographical position. The system is represented by a multilayer graph, consisting of two layers that reflect the hybrid configuration of the distributed system: the mobile P2P layer and the edge computing layer. Consequently, there are two types of nodes: (i) peer nodes (end-users) and (ii) edge nodes. Each node can communicate with other nodes within a certain distance, and end-users move along the grid during the simulation. Therefore, neighborhood relationships are dynamic and change over time, depending on the mobility model.

Several factors influence the efficiency of communication:

- Gossip Protocol. The primary concern is the time required to deliver a message, as one of the main goals of edge computing is to reduce communication latency. While pure broadcast guarantees the fastest delivery, other protocols can minimize traffic, especially in crowded environments.

- Mobility Model. Appropriate mobility models can be used to simulate the movements of the actors involved, depending on the specific application being simulated.

- Communication Range. In this wireless communication model, nodes can directly exchange data when within a certain distance. Adjusting this parameter significantly impacts metrics: a short range may cause nodes to struggle to connect with peers, increasing the number of hops and time to reach an edge node. Conversely, a long-range could make the P2P aspects irrelevant, as end-users would quickly connect with edge nodes.

- Node Density. Similar to models with short communication ranges, a sparsely populated graph risks failing to relay requests to their destination. A very crowded environment, however, could lead to excessive network traffic for message relaying. This issue can be mitigated by adopting an appropriate gossip protocol.

- Amount and Position of Edge Nodes. More edge nodes in the network reduce communication latency. Additionally, strategically positioning edge nodes can significantly decrease delivery time.

6.3.3 Simulation Setup

In the default configuration, used for the following tests, 10 000 nodes populate a 1000 x 1000 grid, thus on average there is a node every 100 cells. The communication radius is set to 40, which means that on average a node reaches 50 other nodes within a circular area of $40^2 * \pi$ cells, unless it is positioned on the edges of the grid. Where it is not specified differently, pure broadcast and Random Waypoint are used respectively as the gossip protocol and the mobility algorithm. Furthermore, the time-to-live for messages has been set equal to 20, even though it was noticed that this value is widely abundant. In the model, 9 edge nodes are placed in an "optimized position" at the center of the Cartesian plane, so that the distance between a cell and an edge node is minimized. However, with such a configuration, end nodes at the edge of the Cartesian plane are the most distant from the edge nodes.

6.3.4 Simulation Results

In this subsection, it is shown how different factors can influence the efficiency of the communication

Mobility Model Figure 6.5 shows how different mobility models affect the average number of hops needed to contact an edge node. In a static setup, fewer messages are sent, leading to higher delivery delays because edge nodes

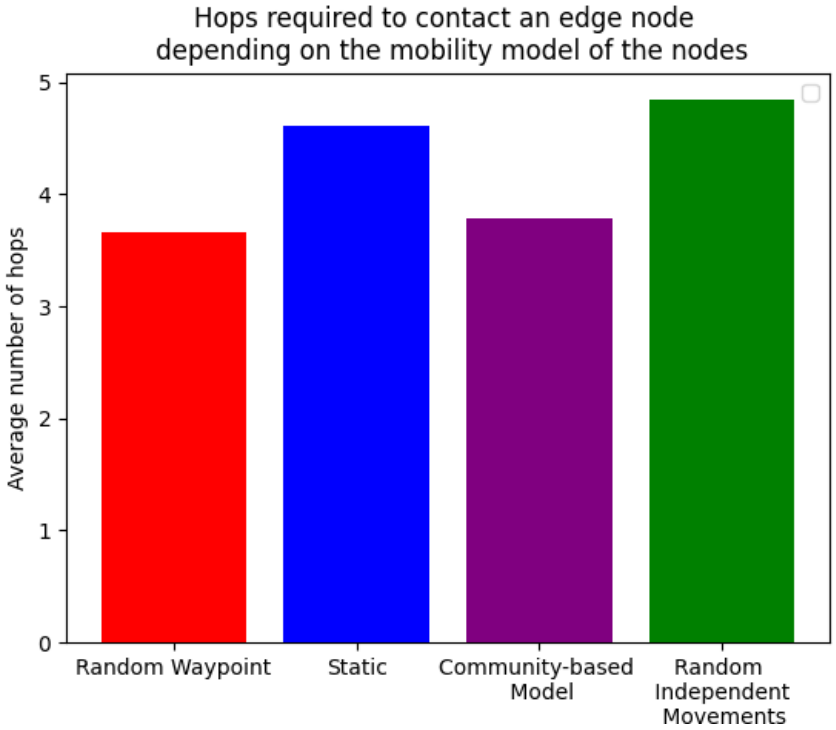


Figure 6.5: Average number of hops necessary to contact an edge node, depending on the mobility model of the nodes.

are centrally placed, making distant nodes send fewer messages. In contrast, the Random Waypoint model has nodes spending more time in the grid center, reducing the number at the edges after an initial period. The static model keeps about 15% of nodes at the edges, while the Random Waypoint model has around 5%. Other algorithms, such as the community-based model and Random Independent Movements, exhibit behaviors akin to the Random Waypoint and static models, respectively.

Grid Density The proposed information dissemination approach assumes a crowded environment where numerous end nodes relay messages to ensure system functionality. In sparsely populated environments, there is a risk of message loss or a significant increase in routing hops. Figure 6.6 illustrates how the successful communication rate (i.e. the percentage of times that a

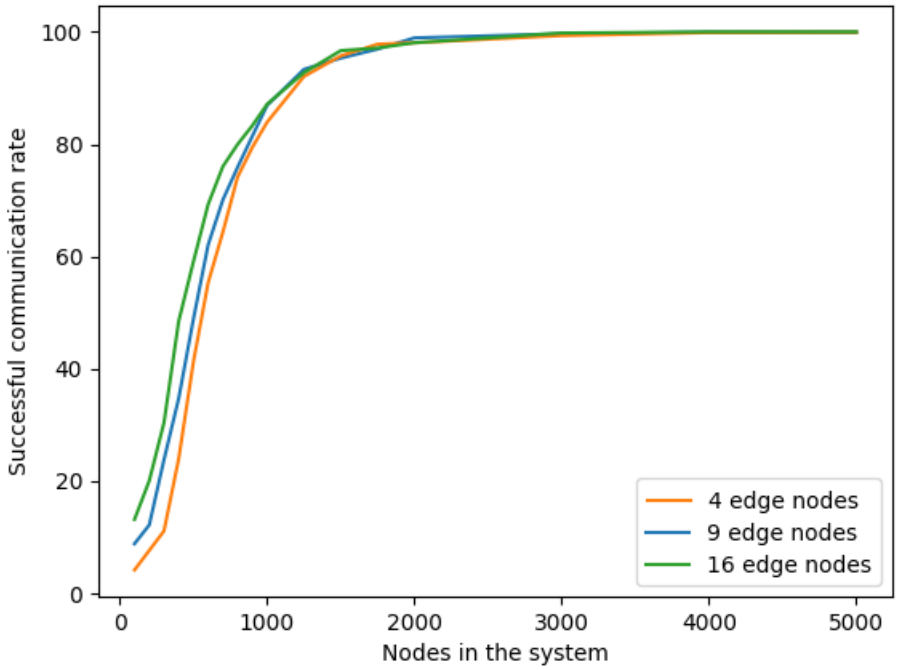


Figure 6.6: Successful communication rate achieved depending on the number of nodes on the grid.

specific node is able to get in touch with a designated node, with respect to all attempts made) changes with a decreasing node population. With the default configuration of 10 000 nodes, 100% coverage is consistently achieved, and similar results are seen with more than 5 000 nodes. Coverage exceeds 99.5% with 3 000 to 5 000 nodes but drops below 99% when fewer than 2 000 nodes are present. Communication reliability notably decreases with 1 500 nodes, where an average of about 7.5 nodes is within the wireless radius.

Edge Nodes The number and arrangement of edge nodes on the Cartesian plane are crucial for minimizing latency. Predictably, the number of hops required to contact an edge node is inversely proportional to the number of edge nodes. In the tests, the default optimized positioning of the edge nodes is compared with a configuration where edge nodes are placed randomly. Figure 6.7 demonstrates that optimized positioning reduces the number

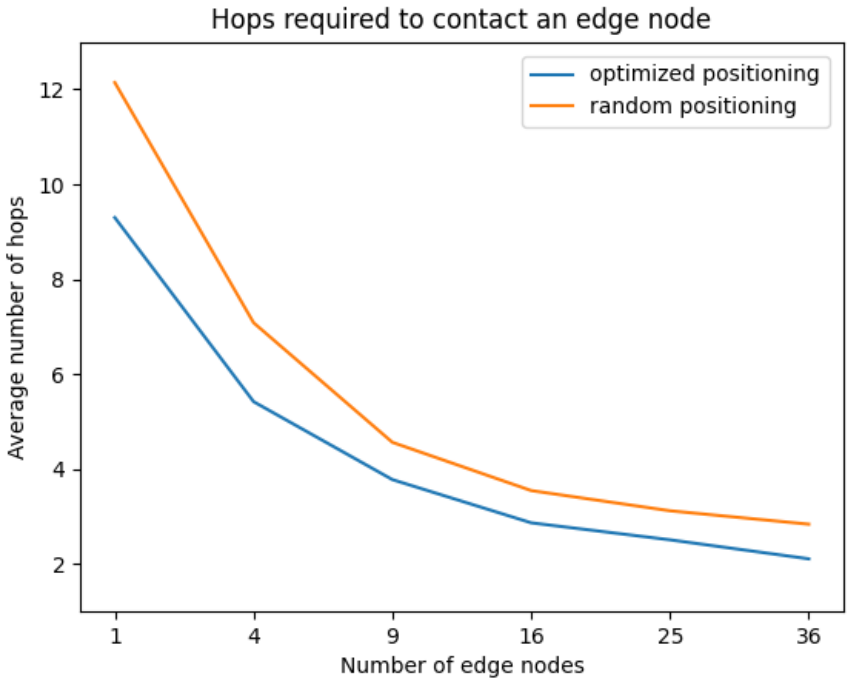


Figure 6.7: Average number of hops necessary to contact an edge node: comparison between an optimized positioning and a random positioning of the edge nodes.

of hops needed to contact an edge node. Despite the reduced delay, the number of messages sent during the experiments remains similar across both configurations. This occurs because the nodes are not aware of the status of the dissemination process, resulting in message propagation continuing even after a destination is reached.

Chapter 7

Simulation of IoT scenarios

The main reason behind the design of this architecture is that while data collection technology is available, large-scale deployment faces practical challenges, primarily linked with the funding of the infrastructure. Crowdsourcing can enhance IoT deployment by leveraging collective community resources, making user motivation crucial for system success.

In this Chapter, a decentralized crowdsensing architecture for collecting environmental data is presented. In the proposed application, vehicles act as data collectors, transferring data to access points, while economic incentives are in the form of blockchain micropayments. The architecture is then evaluated through multilevel M&S. The Chapter is organized as follows. Section 7.1 presents the background concepts behind the proposed architecture and related works. In Section 7.2 the designed architecture is described. Section 7.3 describes the multilevel model and the software integration process. Finally, in Section 7.4 the experiments carried out with the simulator are discussed.

7.1 Background

This section describes the technologies used to build the proposed architecture and reviews relevant works from the scientific literature.

7.1.1 LoRa and LoRaWAN

LoRa is a proprietary wireless communication technology that has gained popularity in IoT scenarios due to its ability to transmit data over long distances while consuming minimal power. LoRa is often used in conjunction with the LoRaWAN, a communication protocol and system architecture that defines the interaction between end devices and gateways in a LoRa network. In a typical LoRaWAN scenario, there are the following actors:

- *Sensor Nodes*, which are the end-devices equipped with sensors to gather the data of interest over the environment. They transmit packets to gateways by using LoRa.
- *Gateways*, which serve as intermediary between IoT devices and the central network. After receiving the LoRa packets from the sensors, the information is then re-transmitted to the Network Server through TCP/IP.
- *Network Server*, which manages network-level functions like packet routing, quality of service, and authentication and authorization for the various involved devices.
- *Join Server*, which handles the device join procedure and the exchange of session keys, establishing a secure connection between end devices and the Network Server.
- *Application Server*, which leverages sensor data to execute business logic, offering services to extract meaningful information for the intended applications, possibly involving external databases or distributed ledgers.

Since LoRaWAN can be used in a wide range of IoT scenarios, multiple types of end devices and activation procedures can be employed. In particular, LoRa sensors can be classified into three categories:

- Class A. Devices sleep most of the time, and after sending uplink packets they open a window for receiving data.

- Class B. In addition to what class A nodes do, devices provide for regularly-scheduled receive windows.

- Class C. Devices can always receive data except when they are in transmit mode. Thus, communicating with the node is easier but the energy consumption is higher.

Regarding the activation procedure, two options are available. Activation By Personalization (ABP) is the solution that ensures lower energy consumption, by preconfiguring the sensor nodes with static security keys before deployment. However, ABP has the following limitations: i) nodes can only work on their predefined network, ii) end devices need to re-negotiate frame counters and session keys, thus leading to the termination of end devices' operational life upon the exhaustion of frame counters, iii) ABP devices cannot negotiate parameters, limiting their flexibility to adapt to changing network conditions. Due to these restrictions, the most preferred strategy is Over-the-Air Activation (OTAA) where end-devices dynamically join the system by exchanging cryptographic keys and parameters with the Network Server over the air during the activation process.

LoRa Simulation Many simulators of LoRa and blockchain technologies exist. Tools like OMNeT++[1] and ns-3[2] provide modules to simulate LoRa, allowing users to customize physical layer attributes like bandwidth and transmission power. However, these tools are quite complex and heavy-weight and are primarily intended to be used in stand-alone models rather than to cooperate with other tools. Fortunately, more lightweight LoRa simulators also exist; for example, LoRaSIM [191] has been used to test various Long Range Wide Area Network (LoRaWAN) configurations with optimized gateway placement, while LoRaWANsim [192] allows users to set the locations of both nodes and gateways.

[1] `https://omnetpp.org/`

[2] `https://www.nsnam.org/`

7.1.2 Blockchain

Distributed Ledger Technology (DLT) is a digital system where data are stored in multiple geographically distributed nodes in the form of transactions. Unlike traditional centralized databases, DLTs rely on Peer-to-peer (P2P) architectures, and a consensus mechanism is employed to ensure that all participants in the system can reach an agreement regarding the status of the ledger. The most popular type of DLT is the blockchain, where transactions are grouped into containers called blocks, which are logically interconnected, as depicted in Figure 7.1.

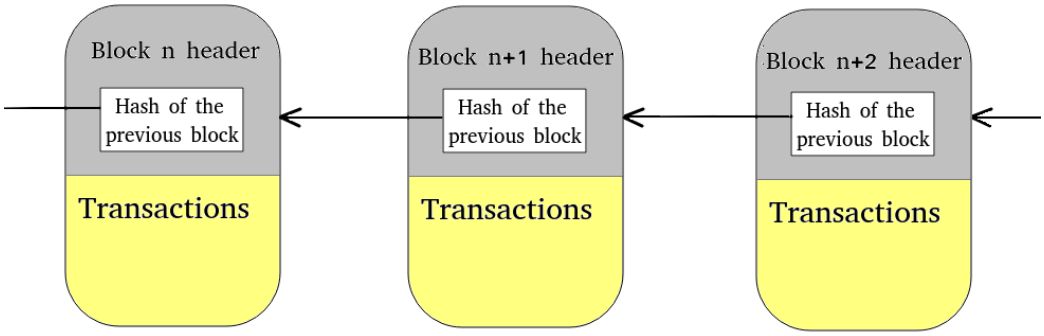


Figure 7.1: A simplified representation of how blocks are chained to form a blockchain.

Cryptography plays a crucial role in various tasks, including ensuring the authenticity of transactions and maintaining a coherent logical chain of blocks within the system. Regardless of the implementation features, in all blockchains the various active participants use a consensus protocol to agree on the state of the distributed ledger, allowing them to determine the validity of the transactions. The most notable examples are Proof of Work (PoW), where participants solve complex computational puzzles to validate transactions and add new blocks to the blockchain, Proof of Stake (PoS) where the chances for the nodes to be the next validators depend on their stake, and Proof of Authority (PoA) where block validators are pre-selected and approved by a central authority or consortium [193].

Common properties and benefits of the blockchain systems are:

- *Immutability.* Once data are recorded on a blockchain, they cannot be altered or deleted.
- *No Bottlenecks or Single Point of Failure* due to the decentralized nature of the blockchains. This makes them more resilient to certain types of attacks and reduces the risk of system failure or unavailability.
- *Non-repudiation.* Private keys are used to sign transactions, which can then be accessible and verified by third parties through the corresponding public key.
- *Transparency and Traceability.* All transactions on a blockchain are visible to all participants in the network, thus creating a high level of accountability. Moreover, each transaction is attached with a timestamp, making it easy for the users to trace the origins of historical data items after analyzing the blockchain data with corresponding timestamps.
- *Cost savings.* By eliminating intermediaries and automating processes, blockchains can reduce the cost of conducting business and increase efficiency.

Despite the common points, blockchains can differ significantly in terms of provided services and implementation features. In particular, three generations of blockchains can be identified. First-generation blockchains like Bitcoin are focused on cryptocurrency exchange, allowing users to trade digital coins by means of transactions. The second generation introduced the concept of smart contracts, which are actual contracts written with code that can automate the execution of complex business operations. Finally, the third generation also aims to address in detail issues such as scalability, interoperability, and sustainability.

Usually, blockchains are *permissionless*, meaning that everyone is allowed to join the system, visualize the history of data, and contribute to the

validation of blocks. No permission has to be granted to the new users, whose identity remains unknown. However, access to the system can be restricted to specific users who have been pre-approved by a single organization or consortium. In these blockchains, referred to as permissioned, the system gives up decentralization, but it can be more suitable for specific business purposes. Permissioned blockchains offer notable scalability benefits by allowing customization of the consensus protocol and block capacity, which can be optimized for efficiency. Furthermore, they provide an additional layer of security because all involved parties are typically considered trusted and expected to adhere to protocols, barring technical failures.

Blockchain Simulation ABM is a fitting modeling approach for replicating blockchain behavior, as each node in the DLT can function as an agent within the simulator. The primary goals of blockchain simulation typically concern scalability, system resilience against cyberattacks or node failures, and performance analysis. Various blockchain simulators exist, such as SimBlock [194] which follows an event-driven approach to model the neighbor nodes selection of the peer-to-peer overlay, BlockSim [195], which has a particular focus on PoW mechanisms and LUNES-blockchain. LUNES-blockchain is based on LUNES (Large Unstructured NEtwork Simulator), a simulator designed to simulate large-scale peer-to-peer networks and to evaluate the efficiency of data dissemination protocols running on top of them.

7.1.3 Blockchain-based LoRaWAN systems

In the scientific literature, many IoT architectures incorporating LoRaWAN and DLTs have been proposed. The use of blockchain within the LoRaWAN framework provides a shared environment where all the stakeholders of the system can execute and receive micropayments, verify the integrity and the authenticity of data, and easily check the history of transactions. For example, data customers might use smart contracts to reward both sensors that collect data and gateways that offer Internet connectivity to the end nodes [196].

The integration of LoRaWAN and the blockchain can be carried out through various approaches, depending on the diverse roles that LoRaWAN devices may assume within the distributed ledger [197]. Gateways can be either *full nodes* or *thin clients* depending on whether they contribute to verifying the integrity of all the received data or only store relevant data fragments. On the other hand, end-devices usually are either *regular sensors* (i.e. low power devices that only broadcast data), *server-trusting clients* (i.e. they rely on APIs to interact with the blockchain, so no storage or computing capability is required), or *thin clients*. In [198] the authors propose to build the blockchain system in the layer of the network servers, citing that gateways normally are resource-constrained and outdoor-deployed IoT devices, thus not suitable to perform the computations required by blockchain activities and to store data. In [199] the authors propose to build the blockchain on the gateway level, claiming that Raspberry platforms have enough power to cope with blockchain functionalities. In [200], on the other hand, the authors propose the employment of a permissioned blockchain network, not directly embedded with one of the LoRaWAN layers. Regardless of the features of the blockchain and the modality of interactions, many existing IoT data marketplace applications rely on the following components [201]:

- A query mechanism that enables clients to filter the data of interest based on selected criteria, such as vendor, temporal and spatial constraints, topology, and other relevant parameters;
- a ranking system for evaluating the quality of data and the reputability of the vendors;
- smart contracts that enable payments from clients to data owners according to a publish-subscribe policy, other than supporting encryption and decryption of data;
- a payment channel where payments can be executed, which supports the execution of smart contracts and possibly guarantees low latencies and low fees.

In [202] a four-layered blockchain platform was designed to manage the trading of weather data. Specifically, the *governance layer* is responsible for deploying and maintaining validators (i.e., those who evaluate the quality of publishers' data) and creating bounties; the *data storage layer* stores the hash of the publishers' data and their score on chain; the *oracle layer* retrieves the weather data, managing the encryption and the validation of the data; and finally the *marketplace layer* manages the payments from the customers to the publishers. In [203] the authors implemented LoRaCHainCare, a LoRa-based blockchain system for healthcare monitoring. In the proposed system LoRa is used to communicate data from the sensors/edge layer to the fog layer (i.e. distributed network of LoRa gateways), which in turn communicates with the cloud layer (i.e. including the LoRa server, the Join server, and the application server) through Internet.

While blockchain is an optimal solution for conducting micropayments and rewarding data producers, other solutions can be used for storing application data. In [204] the authors propose a decentralized marketplace for meter data, where the hash of the data is saved on Ethereum, and IPFS is used to store the entire data. A similar application context is discussed in [205], although with a distinct architectural solution that relies on IOTA as the distributed ledger, again in conjunction with IPFS. Analogously, in [206] the authors describe an architecture where encrypted data are stored with IPFS and a consortium blockchain using Raft as a consensus protocol is used to keep track of metadata. Also, it is not strictly necessary to store transactions into a blockchain, as other types of DLTs like IOTA could be employed. IOTA is based on an acyclic directed graph, where each transaction references and validates previous transactions, thus resulting in higher scalability, faster confirmation times as the network grows, lower costs, and no need for nodes acting specifically as validators. In [207] the authors introduce a smart city data marketplace scenario where transactions are stored on IOTA. The proposed application provides functions to i) find the devices connected to the marketplace, ii) retrieve the data of the specific devices to which the clients

are subscribed, iii) list all the transactions from the clients to the accounts that provide data, iv) receive periodically the data from the sensors whose clients are subscribed.

While LoRa enables communications over a range of several kilometers, the coverage area can be further increased in scenarios involving the mobility of sensors or gateways. In [208], the authors designed and simulated a Smart Livestock Monitoring System where a single mobile gateway moves along the livestock area to cover the whole space to monitor. The experiments proved that the configuration with a mobile gateway is more efficient in terms of economical costs with respect to a setup where multiple static gateways are used. Similarly, in [209] sensors placed in fixed locations collect data about electricity meters, and a gateway moves within the residential area to receive LoRa packets. Mobility also plays an important role in contexts where cellular connectivity is limited or completely missing. In [210], the authors proposed a multi-hop approach where messages are disseminated according to a peer-to-peer strategy in order to enlarge the communication range without any gateway being involved.

Finally, potential security threats must be taken into account. DoS attacks are hard to perform, as LoRa gateways are geographically distributed and only receive physical LoRa signals. However, it is still important that LoRa gateways filter malicious traffic to guarantee that network servers can stay healthy and the consensus process is not disturbed [211]. Furthermore, the distributed nature of the architecture mitigates the problem of the Single Point of Failure, as long as when a gateway is off other gateways are able to cover the signals in that area.

7.2 Architecture of a blockchain-based IoT application

The system discussed in this chapter is based on the LoRaWAN communication protocol for transmitting sensor data and blockchain to facilitate a

marketplace. The use of these technologies aims to enhance energy efficiency and reduce infrastructure costs within smart city frameworks. LoRaWAN has emerged as an efficient, scalable, and secure solution for sensor data exchange in IoT scenarios, supporting various IoT deployments. On the other hand, private blockchains are used to automate frequent micropayments, reduce infrastructure costs, and manage data operations through smart contracts, promoting decentralization while allowing organizations to maintain complete control over the system.

The proposed case study considers a marketplace for environmental data, collected opportunistically by vehicles equipped with LoRa-capable sensors. Data are uploaded via LoRaWAN to a central repository, so that the information is accessible to customers upon payment of some fee, possibly according to the publish-subscribe pattern. Customers may include researchers, meteorologists, public entities in charge of monitoring air quality and pollution, and so forth.

Sensor nodes and LoRaWAN access points are managed by individuals who voluntarily equip their vehicles with sensors or manage access points at home, motivated by an economical compensation that occurs through virtual currency tokens. Similarly, customers pay for accessing the data in the form of the same tokens, with all financial transactions occurring in the blockchain. In particular, in the proposed use case sensors are installed on vehicles, and gateways are placed at fixed locations. While static sensors may provide continuous location-specific information, mobile IoT devices enable data gathering on a larger area, possibly including remote places where stable network connectivity would be impractical or too expensive.

For what concerns the distributed ledger, both permissionless and permissioned blockchains can be employed in IoT applications. However, it is worth noting that permissioned blockchains are particularly suited for this context, as they provide significant scalability benefits by enabling the customization of the consensus protocol and block capacity, which can be set to get an optimal configuration. Additionally, they offer an extra layer of security since

all involved parties may be considered trusted, and therefore are assumed to behave according to the protocols unless technical failures occur. The integration of LoRaWAN components in the blockchain can vary based on the design choices of the application. Gateways can be either *full nodes* or *thin clients*, depending on whether they contribute to verifying the integrity of all the received data or merely store data fragments of interest. On the other hand, end-devices could either be *regular sensors* (i.e., low power devices that only broadcast data), *server-trusting clients* (i.e., they rely on APIs to interact with the blockchain, so no storage or computing capability is required), or *thin clients*.

For the sake of simplicity, it is assumed that all actors are trusted and have followed the necessary LoRaWAN join procedure. Mobile sensors store environmental data locally and forward it to the application server when a gateway is encountered. Gateways then send the data to the network server, which authenticates the source and removes duplicate messages. The network server forwards the filtered information to the application server, which stores the data into a database and credits the gateway, the LoRaWAN provider, and the sensor with tokens, representing virtual currency. The system allows multiple independent service providers (i.e., data brokers) to coexist. A permissioned blockchain might therefore be maintained cooperatively by the service providers, under the assumption that all participants are identifiable and trustworthy. An appropriate balance between the inflow of virtual currency from customers and the outflow towards LoRaWAN administrators and sensor owners is required for the service providers to break even and generate profit for themselves. All monetary transactions are managed by the blockchain, which automatizes the micropayments at a lower cost compared to the traditional channels. Rewards are intended for blockchain nodes linked to physical devices, allowing a single actor to own multiple gateways or sensors. This allows sensor nodes, which are resource-constrained and have sporadic network connectivity, to be excluded from any interaction with the blockchain. Of course, all involved entities (customers, service providers, sensor owners)

can check their balance anytime using personal devices.

To summarize, four categories of nodes are defined in the proposed architectures:

- *Full nodes*, which are the nodes operated by providers that gather transactions and insert them into blockchain blocks. Each provider has at least one full node, enabling a shared marketplace that distributes infrastructure costs and offers a common platform for purchasing sensor data.

- *Sensors*, which generate data and are rewarded by service providers through micropayments. Sensor owners can access the blockchain via personal devices to monitor transactions and withdraw virtual currency so that no blockchain functionality must be embedded in the sensors, conserving their energy by avoiding complex computations.

- *Gateways*, which act as proxy allowing sensors to upload data to the application servers. Gateways are placed at fixed locations where good network connectivity is assumed to be available. Similarly to sensors, no blockchain functionality is required on gateways, since their owners can access the blockchain by other means to check their balance.

- *Customers*, which are the final users of the data. They are subscribed to one or more flows of data upon payment of some fees in virtual currency. The fee might be calculated based on the amount of data accessed, the number of queries, or any other metric that is deemed appropriate.

7.3 Multilevel Model

The system previously discussed was evaluated through simulation, posing a challenge due to the complexity of involving a permissioned blockchain, mobile IoT entities, LoRaWAN technology, and micropayments. To address the involvement of factors of various kinds, the simulator was developed using

the principles of multilevel modeling, in order to speed up the development process and simplify verification and validation by leveraging existing sub-models. The choice of the simulators that are going to be integrated into a multilevel architecture should consider both technical aspects, such as accuracy in reproducing features of interest, and integration feasibility within a complex environment. In fact, modifications to existing models are often necessary to adapt them to the specific scenarios of interest, and to enable interaction with other components.

The building blocks of the proposed model are:

- A mobility simulator for modeling the movements of vehicles. SUMO, an agent-based time-stepped tool for microscopic traffic simulation [212], was selected for this purpose. SUMO provides a platform for modeling complex traffic scenarios, including road networks, vehicles, pedestrians, traffic signals, and various control strategies. Furthermore, it allows for the selection of geographical areas and traffic configurations through provided scripts.

- A LoRa simulator, capable of measuring metrics such as transmit energy consumption, packet delivery ratio, and network throughput. The choice fell on simlorasf [213], a lightweight simulator written in Python that can be more easily integrated into a multilevel framework with respect to other network simulators like OMNeT++ or ns-3.

- A blockchain simulator to reproduce the storage of data and the exchange of tokens within the marketplace. The simulator employed was LUNES-blockchain [214], a time-stepped agent-based simulator capable of reproducing the behavior of all actors involved in the blockchain activities with high accuracy and efficiency. LUNES-blockchain can be easily customized to study different types of scenarios and blockchain policies.

SUMO allows developers to import geographic and road data from OpenStreetMap and configure vehicle flow within the geographic area of interest.

Two parameters contribute to the setup: *Through Traffic Factors* controls the likelihood for vehicles to enter or exit the simulation area. A higher value involves a greater influx and outflow of vehicles at the network borders, displaying increased movement of cars entering and exiting the simulation area. *Count* instead determines the number of vehicles per Km of lane that are generated every hour. The execution of SUMO can be triggered through the TraCI Python library[3] which facilitates the interoperability of SUMO with external software.

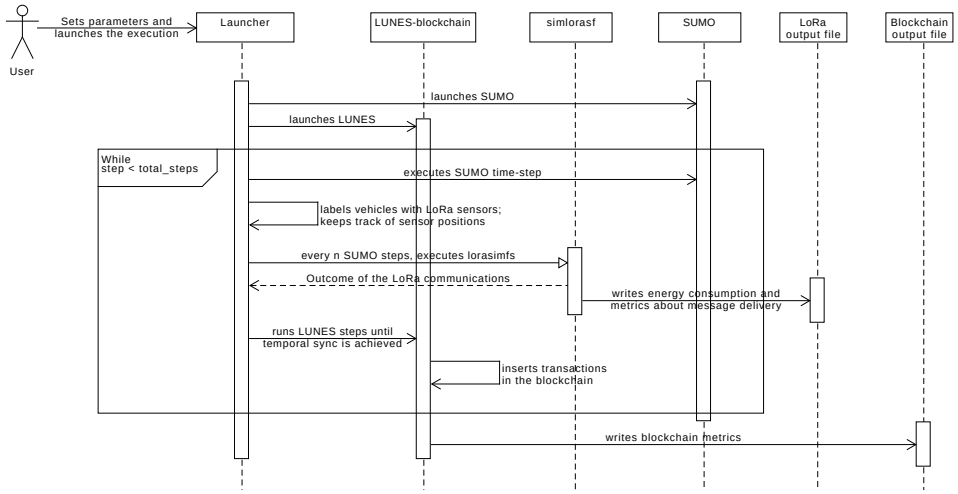


Figure 7.2: Sequence diagram of the multilevel simulator.

The integration of different sub-models requires the definition of orchestration and data exchange patterns, that specify how the various components interact and how they exchange information (for more details see Chapter 4. The interactions are shown in the sequence diagram in Figure 7.2. The *Model's Controller* orchestration pattern is employed, where an ad-hoc module (the Controller, in this case, a launcher script) sits at the top of the multilevel model hierarchy, acting as both the user interface and the Controller for the execution of the sub-models. SUMO and LUNES-blockchain are initiated at the start and remain active throughout the simulation to maintain state

[3] `https://pypi.org/project/traci/`

information (e.g., vehicle positions, velocities, directions, blockchain content) across the entire execution. The launcher progresses through a predefined number of steps: at each step, SUMO advances to the next timestep to update the positions of the vehicles. Next, simlorasf is triggered to manage data transfers for vehicles that have entered the communication range of a gateway. Additionally, simlorasf calculates the energy consumption of the mobile sensors and records this data for post-simulation analysis. Once all communications are finalized, the launcher calls LUNES-blockchain to allocate rewards to nodes and gateways, storing the results in the distributed ledger.

7.4 Simulation Results

The city of Bologna was chosen as the geographical location for the experiments, while the gateways were placed in the positions shown in Figure 7.3. Gateways 1, and 2 are actual LoRaWAN Gateways that are installed in the city, Gateways 3, 4, and 5 are placed in strategic positions (Piazza Maggiore, Dall'Ara Stadium, and Bologna-Mazzini train station), and Gateway 6 is placed in a peripheral location. The arrangement of the sensors has been chosen to cover a significant portion of the map and particularly focusing on high-traffic areas. Most of the vehicles in fact are active in the center, while the green-shaded parts of the map represent hilly zones with the surrounding streets having lower vehicle density.

Following an initial warm-up phase required to populate the area with a suitable number of vehicles, the simulation proceeded for 1000 steps, where each SUMO step represents 1 second of wall-clock time. It is assumed that every 10 seconds the sensors emit messages, which can be received in the range of 2100 meters [215]. LoRa-equipped vehicles lack awareness of gateway presence within their transmission range, making it hard to ensure that all transmitted data has been received. While occasional data losses might be acceptable in some applications, it might not be acceptable in others. There are various solutions to mitigate this issue. One approach involves

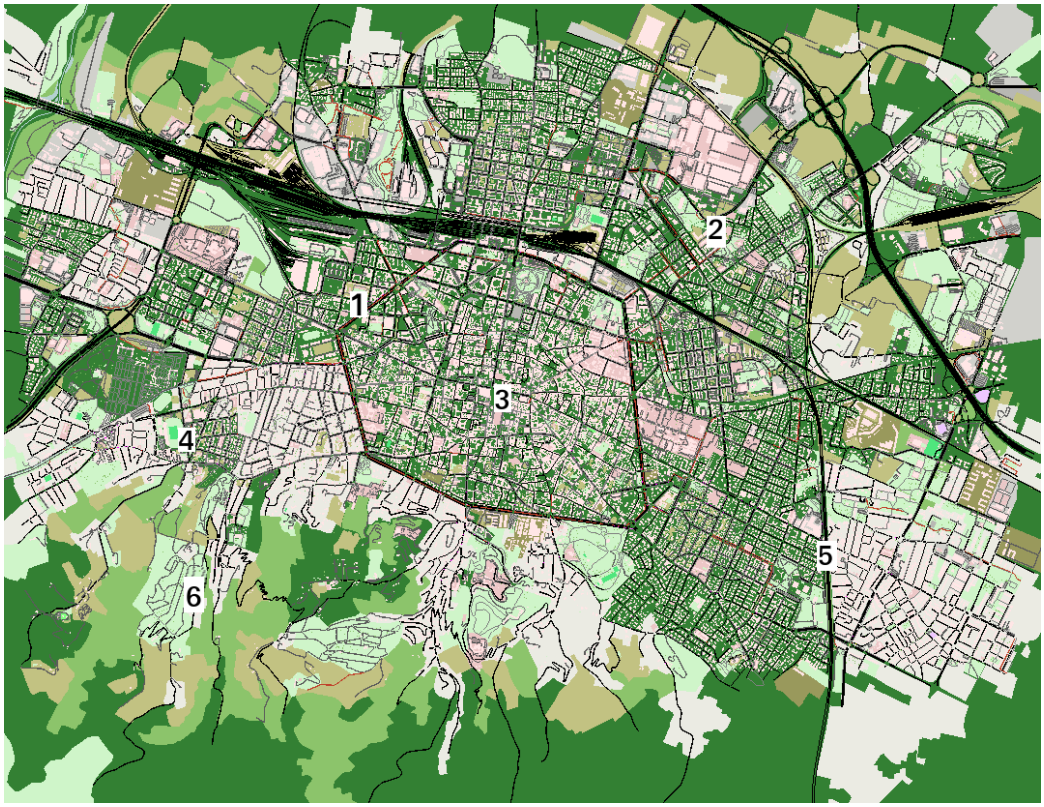


Figure 7.3: Placement of LoRa Gateways in the simulation scenario.

storing and transmitting the last k data points, where k is the transmission window. Higher values of k increase the likelihood that all data points will eventually reach the application server at least once, at the cost of a larger message size and greater effort for identifying and removing duplicates on the receiving side. Other approaches based on explicit acknowledgements using the downlink channel might be designed, but they require sensors to wait for message receipts and the handling of retransmissions. In terms of SUMO traffic generation parameters, the testbed has a *Through Traffic Factors* value (i.e. likelihood for the vehicles of choosing a boundary edge) of 5 and a *Count* value (i.e., quantity of vehicles produced every hour per kilometers of lane) of 100; it is also assumed that 10% of all vehicles are equipped with sensors.

Regarding financial compensation, it is assumed that sensors, gateways, and providers are rewarded with 1 token for every transaction validated in

the blockchain. This fixed reward has been chosen to simplify experiments, whereas in a real scenario different services may have varying prices, and the reward could be distributed among the involved actors in different ways. In the experiments, the amount of tokens earned by gateways is directly proportional to the number of forwarded messages. If multiple gateways receive the same message, LoRaWAN deduplication policies apply, and the token is assigned to the gateway with the strongest signal, which in the simulation is always assumed to be the one closest to the sensor.

Figure 7.4 shows the reward obtained by the gateways after 1000 seconds of simulated time. The gateways receiving a higher reward are either those positioned in the busiest areas (Gateways 1 and 3) or those individually covering a wide territory (Gateway 5). On the other hand, Gateway 6, which is located in a peripheral and low-traffic area of the city, is by far the one that retransmits the fewest messages.

The fraction of cars equipped with sensors has a big influence on metrics such as transmit energy consumption and the packet delivery ratio, as shown in Figure 7.5. As this parameter increases, the energy consumption grows proportionally, while the packet delivery ratio decreases due to a higher amount of interference.

Managing a large volume of transactions implies the need for blockchain solutions capable of handling high data flow. Permissioned blockchains offer the advantage of customizing the distributed ledger's configuration to effectively accommodate the data rate required by the IoT application. The capacity to support transactions within a specific timeframe is determined by two key attributes: the maximum number of transactions per block and the block production frequency, together defining the maximum transaction throughput. Figure 7.6 shows the percentage of transactions successfully inserted into a block, depending on the number of vehicles equipped with LoRa sensors and the throughput, measured as the maximum number of transactions that can be inserted in one minute. Obviously, in a real-world system, it is desirable that no transaction gets lost. Therefore, it is necessary

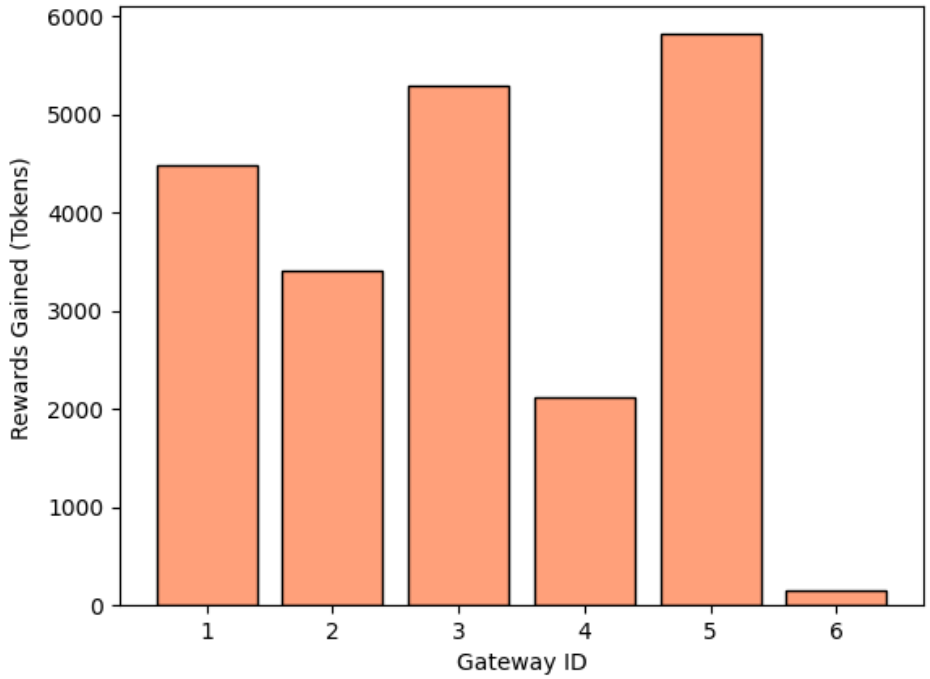


Figure 7.4: Rewards obtained by each gateway.

to customize the blockchain attributes in a way that a 100% insertion rate is ensured.

In the experiments, consensus is achieved through PoA where provider nodes, serving as trusted entities, are responsible for block production. The proposed architecture is not heavily reliant on a specific consensus mechanism. While PoA is a well-established choice for permissioned blockchain [216], there is no specific standard about how the PoA is implemented. Therefore, a PoS-based implementation was chosen due to its compatibility with the simulator. However, any suitable consensus mechanism could be integrated.

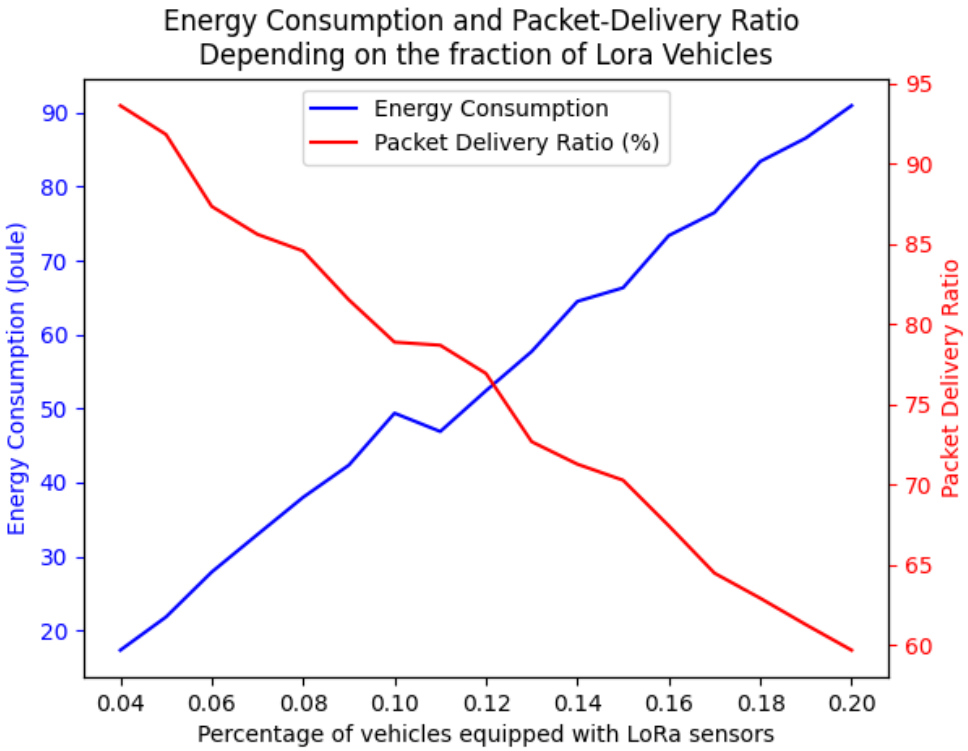


Figure 7.5: Increasing the number of sensors entails greater transmit energy consumption and lower packet delivery ratio.

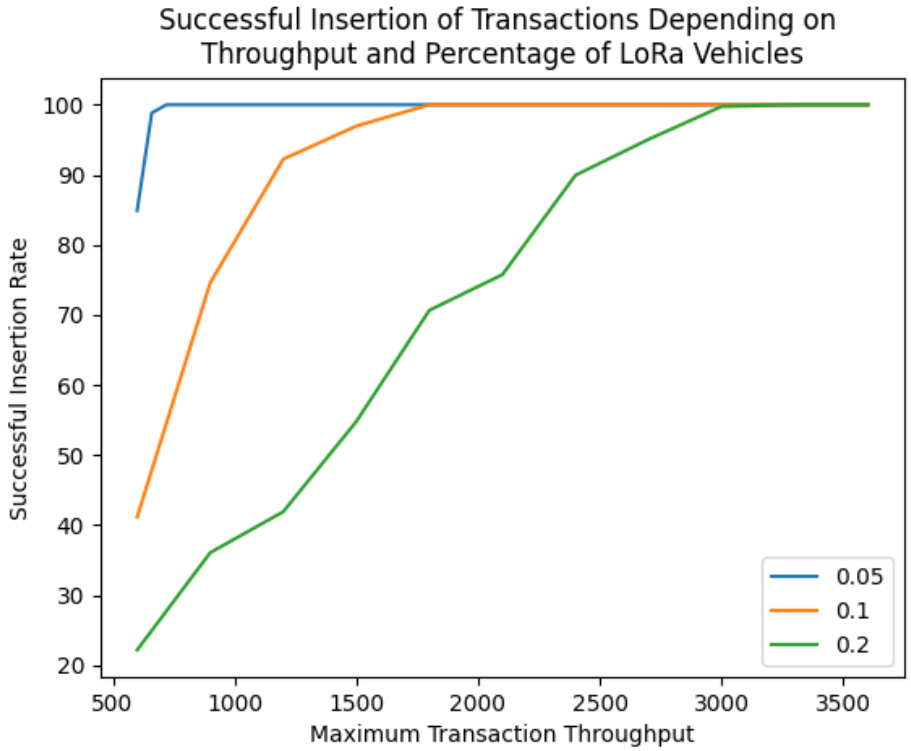


Figure 7.6: Transactions successfully validated in the blockchain. The three curves represent the percentage of vehicles equipped with LoRa sensors, respectively 5%, 10%, and 20% of all vehicles.

Chapter 8

Conclusions

In this dissertation, several aspects connected with multilevel modeling have been investigated. Through the analysis of the state of the art, it turned out that multilevel modeling is widely used in scientific research, and also finds application in fields concerning human mobility and behavior. In particular, in the various areas that have been considered, it was possible to identify certain methodological patterns, meaning that frequently certain types of studies are carried out with similar methodologies, as pointed out in Table 8.1.

Application Area	**Methodology**
Local epidemic diffusion	Within-host model + Between-host model
Global epidemic diffusion	Large-scale mobility model + Local between-host models
Traffic	Macroscopic model + Microscopic model
Crowd behavior	Macroscopic model + Microscopic model
Crowd evacuation	Smoke/fire model + Microscopic model
Population Dynamics	Multilevel logistic regression model
Information diffusion	Multilayer graph

Table 8.1: Most frequently used methodologies within each research area.

Research on epidemic spreading frequently employs a combination of within-host and between-host models. These models naturally describe the diffusion of a pathogen inside the human body (within-host) and its spread across individuals (between-host). On a larger scale, global epidemic diffusion

uses mobility models to account for human movement. Traffic modeling and crowd mobility studies utilize microscopic models to accurately depict individual behavior, combined with macroscopic models to describe flows of vehicles or crowds. Emergency scenarios, such as crowd evacuation due to floods or fires, integrate microscopic models with water or fire propagation models, as individual behavior is influenced by perceived danger levels in their surroundings. Population dynamics studies typically rely on multilevel continuous models, and finally, multilayer graphs are commonly used for studying information diffusion.

While it has emerged that various types of methodologies can be employed and combined, common design principles for multilevel modeling can still be identified, regardless of the semantics. Therefore, this dissertation introduces GEMMA, a metamodel designed to facilitate the creation of multilevel models. GEMMA accurately specifies the interplay among sub-models, potentially based on different paradigms, including call conditions, information exchange, and consistency rules.

Furthermore, four categories of design patterns were proposed to give technical solutions to recurrent modeling issues. Specifically, Orchestration Patterns are strategies for organizing the various building blocks, Structural Patterns are design solutions for developing software systems with a hierarchical structure, Multiscale Patterns are procedures for representing a system with multiple scales of detail, and lastly, Information Exchange patterns describe how data can be exchanged among the sub-models.

The final part of the thesis presents applications of ABM and multilevel modeling. ABM was proved to be well-suited for detailed representations of scenarios involving several heterogeneous entities, while multilevel modeling was employed to study an IoT application, as the presence of multiple factors of interest makes this methodology particularly appropriate for such a context. Specifically, the proposed IoT architecture integrates the gathering and trading of sensor data to create a decentralized marketplace, facilitating the acquisition of sensor data over the territory of interest. While LoRaWAN is an established

standard for IoT applications, enabling efficient retrieval of data, blockchain technology could facilitate the establishment of a marketplace, empowering customers to subscribe to various data streams, possibly generated and managed by different entities. From a modeling perspective, a blockchain model, a LoRa model, and a vehicle mobility model were integrated using the discussed methodology. This approach enabled the use of existing specialized models to represent various aspects of a complex scenario, without the need of developing a new simulator from scratch. The experiments indicate that even a small number of strategically positioned gateways can achieve extensive coverage in large urban areas.

Bibliography

[1] D. Weyns and T. Holvoet, "A formal model for situated multi-agent systems," *Fundamenta Informaticae*, vol. 63, no. 2-3, pp. 125–158, 2004.

[2] L. Serena, G. D'Angelo, and S. Ferretti, "Security analysis of distributed ledgers and blockchains through agent-based simulation," *Simulation Modelling Practice and Theory*, vol. 114, p. 102413, 2022.

[3] S. Wolfram, "Cellular automata as models of complexity," *Nature*, vol. 311, no. 5985, pp. 419–424, 1984.

[4] R. G. Coyle, "System dynamics modelling: a practical approach," *Journal of the Operational Research Society*, vol. 48, no. 5, pp. 544–544, 1997.

[5] D. H. Meadows, J. Randers, and D. L. Meadows, *Limits to Growth–The 30 year update*. Chelsea Green Publishing, 2004.

[6] S. Ghosh, "On the concept of dynamic multi-level simulation," in *Proceedings of the 19th annual symposium on Simulation*, pp. 201–205, 1986.

[7] A. Handel and P. Rohani, "Crossing the scale from within-host infection dynamics to between-host transmission fitness: a discussion of current assumptions and knowledge," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 370, no. 1675, p. 20140302, 2015.

[8] S. Karabasov, D. Nerukh, A. Hoekstra, B. Chopard, and P. V. Coveney, "Multiscale modelling: approaches and challenges," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 372, no. 2021, p. 20130390, 2014.

[9] P. D. Dans, J. Walther, H. Gómez, and M. Orozco, "Multiscale simulation of dna," *Current opinion in structural biology*, vol. 37, pp. 29–45, 2016.

[10] D. Bishara, Y. Xie, W. K. Liu, and S. Li, "A state-of-the-art review on machine learning-based multiscale modeling, simulation, homogenization and design of materials," *Archives of computational methods in engineering*, vol. 30, no. 1, pp. 191–222, 2023.

[11] Z. Hammoud and F. Kramer, "Multilayer networks: aspects, implementations, and application in biomedicine," *Big Data Analytics*, vol. 5, no. 1, pp. 1–18, 2020.

[12] A. C. Kinsley, G. Rossi, M. J. Silk, and K. VanderWaal, "Multilayer and multiplex networks: an introduction to their use in veterinary epidemiology," *Frontiers in veterinary science*, vol. 7, p. 596, 2020.

[13] J. S. Dahmann, "High level architecture for simulation," in *Proceedings First International Workshop on Distributed Interactive Simulation and Real Time Applications*, pp. 9–14, IEEE, 1997.

[14] L. Serena, M. Marzolla, G. D'Angelo, and S. Ferretti, "A review of multilevel modeling and simulation for human mobility and behavior," *Simulation Modelling Practice and Theory*, p. 102780, 2023.

[15] V. K. Nguyen, R. Mikolajczyk, and E. A. Hernandez-Vargas, "High-resolution epidemic simulation using within-host infection and contact data," *BMC public health*, vol. 18, no. 1, pp. 1–11, 2018.

[16] E. A. Hernandez-Vargas, A. Y. Alanis, and J. Tetteh, "A new view of multiscale stochastic impulsive systems for modeling and control of epidemics," *Annual Reviews in Control*, vol. 48, pp. 242–249, 2019.

[17] S. Lukens, J. DePasse, R. Rosenfeld, E. Ghedin, E. Mochan, S. T. Brown, J. Grefenstette, D. S. Burke, D. Swigon, and G. Clermont, "A large-scale immuno-epidemiological simulation of influenza a epidemics," *BMC public health*, vol. 14, no. 1, pp. 1–15, 2014.

[18] B. Musundi, J. Müller, and Z. Feng, "A multi-scale model for cholera outbreaks," *Mathematics*, vol. 10, no. 17, p. 3114, 2022.

[19] X. Cen, Z. Feng, and Y. Zhao, "Emerging disease dynamics in a model coupling within-host and between-host systems," *Journal of theoretical biology*, vol. 361, pp. 141–151, 2014.

[20] Z. Feng, J. Velasco-Hernandez, and B. Tapia-Santos, "A mathematical model for coupling within-host and between-host dynamics in an environmentally-driven infectious disease," *Mathematical biosciences*, vol. 241, no. 1, pp. 49–55, 2013.

[21] E. Numfor, S. Bhattacharya, S. Lenhart, and M. Martcheva, "Optimal control in coupled within-host and between-host models," *Mathematical Modelling of Natural Phenomena*, vol. 9, no. 4, pp. 171–203, 2014.

[22] N. Bellomo, D. Burini, and N. Outada, "Multiscale models of covid-19 with mutations and variants," *Networks and Heterogeneous Media*, vol. 17, no. 3, p. 293, 2022.

[23] X. Wang, S. Wang, J. Wang, and L. Rong, "A multiscale model of covid-19 dynamics," *Bulletin of Mathematical Biology*, vol. 84, no. 9, pp. 1–41, 2022.

[24] B. Durand, M. A. Dubois, P. Sabatier, D. Calavas, C. Ducrot, and A. Van de Wielle, "Multiscale modelling of scrapie epidemiology: Ii.

geographical level: hierarchical transfer of the herd model to the regional disease spread," *Ecological Modelling*, vol. 179, no. 4, pp. 515–531, 2004.

[25] M. A. Gilchrist and A. Sasaki, "Modeling host–parasite coevolution: a nested approach based on mechanistic models," *Journal of Theoretical Biology*, vol. 218, no. 3, pp. 289–308, 2002.

[26] M. Tracy, M. Cerdá, and K. M. Keyes, "Agent-based modeling in public health: current applications and future directions," *Annual review of public health*, vol. 39, p. 77, 2018.

[27] I. Cooper, A. Mondal, and C. G. Antonopoulos, "A sir model assumption for the spread of covid-19 in different communities," *Chaos, Solitons & Fractals*, vol. 139, p. 110057, 2020.

[28] M. H. A. Biswas, L. T. Paiva, and M. De Pinho, "A seir model for control of infectious diseases with constraints," *Mathematical Biosciences & Engineering*, vol. 11, no. 4, p. 761, 2014.

[29] H. Shi, Z. Duan, and G. Chen, "An sis model with infective medium on complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 8-9, pp. 2133–2144, 2008.

[30] A. Welling, A. Patel, P. Kulkarni, and V. G. Vaidya, "Multilevel integrated model with a novel systems approach (mimansa) for simulating the spread of covid-19," *medRxiv*, 2020.

[31] N. Chen, D. Rey, and L. Gardner, "Multiscale network model for evaluating global outbreak control strategies," *Transportation Research Record*, vol. 2626, no. 1, pp. 42–50, 2017.

[32] T. Li, "Simulating the spread of epidemics in china on multi-layer transportation networks: Beyond covid-19 in wuhan," *EPL (Europhysics Letters)*, vol. 130, no. 4, p. 48002, 2020.

[33] B. Lieberthal and A. M. Gardner, "Connectivity, reproduction number, and mobility interact to determine communities' epidemiological superspreader potential in a metapopulation network," *PLOS Computational Biology*, vol. 17, no. 3, p. e1008674, 2021.

[34] L. Kou, X. Wang, Y. Li, X. Guo, and H. Zhang, "A multi-scale agent-based model of infectious disease transmission to assess the impact of vaccination and non-pharmaceutical interventions: The covid-19 case," *Journal of Safety Science and Resilience*, vol. 2, no. 4, pp. 199–207, 2021.

[35] L. G. A. Zuzek, C. Buono, and L. A. Braunstein, "Epidemic spreading and immunization strategy in multiplex networks," in *Journal of Physics: Conference Series*, vol. 640, p. 012007, IOP Publishing, 2015.

[36] L. G. A. Zuzek, H. E. Stanley, and L. A. Braunstein, "Epidemic model with isolation in multilayer networks," *Scientific reports*, vol. 5, pp. 1–7, 2015.

[37] D. Wu, Y. Liu, M. Tang, X.-K. Xu, and S. Guan, "Impact of hopping characteristics of inter-layer commuters on epidemic spreading in multilayer networks," *Chaos, Solitons & Fractals*, vol. 159, p. 112100, 2022.

[38] A. Vajdi, D. Juher, J. Saldaña, and C. Scoglio, "A multilayer temporal network model for std spreading accounting for permanent and casual partners," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[39] S. Chang, E. Pierson, P. W. Koh, J. Gerardin, B. Redbird, D. Grusky, and J. Leskovec, "Mobility network models of covid-19 explain inequities and inform reopening," *Nature*, vol. 589, no. 7840, pp. 82–87, 2021.

[40] W. Boscheri, G. Dimarco, and L. Pareschi, "Modeling and simulating the spatial spread of an epidemic through multiscale kinetic transport

equations," *Mathematical Models and Methods in Applied Sciences*, pp. 1–39, 2021.

[41] S. Namilae, P. Derjany, A. Mubayi, M. Scotch, and A. Srinivasan, "Multiscale model for pedestrian and infection dynamics during air travel," *Physical review E*, vol. 95, no. 5, p. 052320, 2017.

[42] P. Derjany, S. Namilae, D. Liu, and A. Srinivasan, "Multiscale model for the optimal design of pedestrian queues to mitigate infectious disease spread," *PloS one*, vol. 15, no. 7, p. e0235891, 2020.

[43] A. Bouchnita and A. Jebrane, "A hybrid multi-scale model of covid-19 transmission dynamics to assess the potential of non-pharmaceutical interventions," *Chaos, Solitons & Fractals*, vol. 138, p. 109941, 2020.

[44] D. J. Watts, R. Muhamad, D. C. Medina, and P. S. Dodds, "Multiscale, resurgent epidemics in a hierarchical metapopulation model," *Proceedings of the National Academy of Sciences*, vol. 102, no. 32, pp. 11157–11162, 2005.

[45] L. Wang, Y. Zhang, Z. Wang, and X. Li, "The impact of human location-specific contact pattern on the sir epidemic transmission between populations," *International Journal of Bifurcation and Chaos*, vol. 23, no. 05, p. 1350095, 2013.

[46] D. Balcan, V. Colizza, B. Gonçalves, H. Hu, J. J. Ramasco, and A. Vespignani, "Multiscale mobility networks and the spatial spreading ofr infectious diseases," *Proceedings of the National Academy of Sciences*, vol. 106, no. 51, pp. 21484–21489, 2009.

[47] B. Bonté, R. Duboz, G. Quesnel, and J. P. Müller, "Recursive simulation and experimental frame for multiscale simulation," in *Proc. 2009 Summer Computer Simulation Conference*, Jan. 2009.

[48] S. Funk, E. Gilad, C. Watkins, and V. A. Jansen, "The spread of awareness and its impact on epidemic outbreaks," *Proceedings of the National Academy of Sciences*, vol. 106, no. 16, pp. 6872–6877, 2009.

[49] K. Peng, Z. Lu, V. Lin, M. R. Lindstrom, C. Parkinson, C. Wang, A. L. Bertozzi, and M. A. Porter, "A multilayer network model of the coevolution of the spread of a disease and competing opinions," *Mathematical Models and Methods in Applied Sciences*, vol. 31, no. 12, pp. 2455–2494, 2021.

[50] B. She, J. Liu, S. Sundaram, and P. E. Paré, "On a networked sis epidemic model with cooperative and antagonistic opinion dynamics," *IEEE Transactions on Control of Network Systems*, 2022.

[51] J. Casas, J. Perarnau, and A. Torday, "The need to combine different traffic modelling levels for effectively tackling large-scale projects adding a hybrid meso/micro approach," *Procedia-Social and Behavioral Sciences*, vol. 20, pp. 251–262, 2011.

[52] J. J. Olstam and A. Tapani, *Comparison of Car-following models*, vol. 960. Swedish National Road and Transport Research Institute Linköping, 2004.

[53] N. E. Yunus, S. F. A. Razak, S. Yogarayan, and M. F. A. Abdullah, "Lane changing models: A short review," in *2021 IEEE 12th Control and System Graduate Research Colloquium (ICSGRC)*, pp. 110–115, IEEE, 2021.

[54] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[55] D. Makowiec and W. Miklaszewski, "Nagel-schreckenberg model of traffic–study of diversity of car rules," in *International Conference on Computational Science*, pp. 256–263, Springer, 2006.

[56] G. O. Kagho, M. Balac, and K. W. Axhausen, "Agent-based models in transport planning: Current state, issues, and expectations," *Procedia Computer Science*, vol. 170, pp. 726–732, 2020.

[57] J. Nguyen, S. T. Powers, N. Urquhart, T. Farrenkopf, and M. Guckert, "An overview of agent-based traffic simulators," *Transportation research interdisciplinary perspectives*, vol. 12, p. 100486, 2021.

[58] A. L. Bazzan and F. Klügl, "A review on agent-based technology for traffic and transportation," *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 375–403, 2014.

[59] P. Jing, H. Hu, F. Zhan, Y. Chen, and Y. Shi, "Agent-based simulation of autonomous vehicles: A systematic literature review," *IEEE Access*, vol. 8, pp. 79089–79103, 2020.

[60] M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, and K. Nagel, "Matsim-t: Architecture and simulation times," in *Multi-agent systems for traffic and transportation engineering*, pp. 57–78, IGI Global, 2009.

[61] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *2018 21st international conference on intelligent transportation systems (ITSC)*, pp. 2575–2582, IEEE, 2018.

[62] G. Ben-Dor, E. Ben-Elia, and I. Benenson, "Population downscaling in multi-agent transportation simulations: A review and case study," *Simulation Modelling Practice and Theory*, vol. 108, p. 102233, 2021.

[63] V. L. Knoop, D. De Jong, and S. P. Hoogendoom, "Influence of road layout on network fundamental diagram," *Transportation Research Record*, vol. 2421, no. 1, pp. 22–30, 2014.

[64] D. Vikram, P. Chakroborty, and S. Mittal, "Exploring the behavior of lwr continuum models of traffic flow in presence of shock waves," *Procedia-Social and Behavioral Sciences*, vol. 104, pp. 412–421, 2013.

[65] C. Caligaris, S. Sacone, and S. Siri, "On the payne-whitham differential model: stability constraints in one-class and two-class cases," *Applied Mathematical Sciences*, vol. 4, no. 76, pp. 3795–3821, 2010.

[66] H. Yu and M. Krstic, "Traffic congestion control for aw–rascle–zhang model," *Automatica*, vol. 100, pp. 38–51, 2019.

[67] A. Poschinger, R. Kates, and H. Keller, "Coupling of concurrent macroscopic and microscopic traffic flow models using hybrid stochastic and deterministic disaggregation," in *Transportation and Traffic Theory in the 21st Century. Proceedings of the 15th International Symposium on Transportation and Traffic Theory*, pp. 583–605, Emerald Group Publishing Limited, 2002.

[68] N. Bouha, G. Morvan, H. Abouaissa, and Y. Kubera, "A first step towards dynamic hybrid traffic modeling," in *Proceedings 29th European Conference on Modelling and Simulation*, pp. 64–70, 2015.

[69] J. Sewall, D. Wilkie, and M. C. Lin, "Interactive hybrid simulation of large-scale traffic," in *Proceedings of the 2011 SIGGRAPH Asia Conference*, pp. 1–12, 2011.

[70] G. Jakovljevic and D. Basch, "Implementing multiscale traffic simulators using agents," in *26th International Conference on Information Technology Interfaces, 2004.*, pp. 519–524, IEEE, 2004.

[71] M. P. Raadsen, M. C. Bliemer, and M. G. Bell, "Aggregation, disaggregation and decomposition methods in traffic assignment: historical perspectives and new trends," *Transportation research part B: methodological*, vol. 139, pp. 199–223, 2020.

[72] M. S. El Hmam, H. Abouaissa, D. Jolly, and A. Benasser, "Towards an hybrid simulation approach of transportation systems," *IFAC Proceedings Volumes*, vol. 37, no. 19, pp. 75–80, 2004.

[73] M. said EL HMAM, H. ABOUAISSA, D. JOLLY, and A. BENASSER, "Macro-micro simulation of traffic flow," *IFAC Proceedings Volumes*, vol. 39, no. 3, pp. 351–356, 2006. 12th IFAC Symposium on Information Control Problems in Manufacturing.

[74] S. Mammar, S. Mammar, and J.-P. Lebacque, "Highway traffic hybrid macro-micro simulation model," *IFAC Proceedings Volumes*, vol. 39, no. 12, pp. 627–632, 2006.

[75] A. Banos, N. Corson, C. Lang, N. Marilleau, and P. Taillandier, "Multiscale modeling: application to traffic flow," in *Agent-based Spatial Simulation with NetLogo, Volume 2*, pp. 37–62, Elsevier, 2017.

[76] E. Bourrel and J.-B. Lesort, "Mixing microscopic and macroscopic representations of traffic flow: Hybrid model based on lighthill–whitham–richards theory," *Transportation Research Record*, vol. 1852, no. 1, pp. 193–200, 2003.

[77] M. Joueiai, L. Leclercq, H. Van Lint, and S. P. Hoogendoorn, "Multiscale traffic flow model based on the mesoscopic lighthill–whitham and richards models," *Transportation Research Record*, vol. 2491, no. 1, pp. 98–106, 2015.

[78] M. Joueiai, H. Van Lint, and S. P. Hoogendoom, "Multiscale traffic flow modeling in mixed networks," *Transportation Research Record*, vol. 2421, no. 1, pp. 142–150, 2014.

[79] X. Boulet, M. Zargayouna, G. Scemama, and F. Leurent, "Service-oriented architecture for multiscale traffic simulations," in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–8, IEEE, 2019.

[80] M. Garavello, P. Goatin, T. Liard, and B. Piccoli, "A multiscale model for traffic regulation via autonomous vehicles," *Journal of Differential Equations*, vol. 269, no. 7, pp. 6088–6124, 2020.

[81] M. L. Delle Monache and P. Goatin, "Scalar conservation laws with moving constraints arising in traffic flow modeling: an existence result," *Journal of Differential equations*, vol. 257, no. 11, pp. 4015–4029, 2014.

[82] W. Burghout and J. Wahlstedt, "Hybrid traffic simulation with adaptive signal control," *Transportation Research Record*, vol. 1999, no. 1, pp. 191–197, 2007.

[83] W. Burghout, H. N. Koutsopoulos, and I. Andreasson, "Hybrid mesoscopic–microscopic traffic simulation," *Transportation Research Record*, vol. 1934, no. 1, pp. 218–225, 2005.

[84] R. Jayakrishnan, J.-S. Oh, and A.-E.-K. Sahraoui, "Calibration and path dynamics issues in microscopic simulation for advanced traffic management and information systems," *Transportation Research Record*, vol. 1771, no. 1, pp. 9–17, 2001.

[85] S. Bosmans, T. Bogaerts, W. Casteels, S. Mercelis, J. Denil, and P. Hellinckx, "Adaptivity in multi-level traffic simulation using experimental frames," *Simulation Modelling Practice and Theory*, vol. 114, p. 102395, 2022.

[86] P. Kumar, R. Merzouki, B. Conrard, V. Coelen, and B. O. Bouamama, "Multilevel modeling of the traffic dynamic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1066–1082, 2014.

[87] D. Ni, "Multiscale modeling of traffic flow," *Mathematica Aeterna*, vol. 1, no. 1, pp. 27–54, 2011.

[88] R. Alqurashi and T. Altman, "Hierarchical agent-based modeling for improved traffic routing," *Applied Sciences*, vol. 9, no. 20, 2019.

[89] J. Serras, "Extending transims technology to an integrated multilevel representation," in *CUPUM '05: Computers in Urban Planning and Urban Management*, 2005.

[90] K. Ijaz, S. Sohail, and S. Hashish, "A survey of latest approaches for crowd simulation and modeling using hybrid techniques," in *17th UKSIMAMSS international conference on modelling and simulation*, pp. 111–116, 2015.

[91] N. Bellomo and A. Bellouquid, "On multiscale models of pedestrian crowds from mesoscopic to macroscopic," *Communications in Mathematical Sciences*, vol. 13, no. 7, pp. 1649–1664, 2015.

[92] F. Martinez-Gil, M. Lozano, I. García-Fernández, and F. Fernández, "Modeling, evaluation, and scale on artificial pedestrians: a literature review," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–35, 2017.

[93] Z. Zhang and L. Jia, "Optimal guidance strategy for crowd evacuation with multiple exits: A hybrid multiscale modeling approach," *Applied Mathematical Modelling*, vol. 90, pp. 488–504, 2021.

[94] R. Alqurashi and T. Altman, "Multi-level multi-stage agent-based decision support system for simulation of crowd dynamics," in *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, pp. 82–92, IEEE, 2018.

[95] N. T. N. Anh, Z. J. Daniel, N. H. Du, A. Drogoul, and V. D. An, "A hybrid macro-micro pedestrians evacuation model to speed up simulation in road networks," in *International Conference on Autonomous Agents and Multiagent Systems*, pp. 371–383, Springer, 2011.

[96] P. C. Tissera, A. M. Printista, and E. Luque, "A hybrid simulation model to test behaviour designs in an emergency evacuation," *Procedia Computer Science*, vol. 9, pp. 266–275, 2012.

[97] O. Richardson, A. Jalba, and A. Muntean, "Effects of environment knowledge in evacuation scenarios involving fire and smoke: a multiscale modelling and simulation approach," *Fire technology*, vol. 55, no. 2, pp. 415–436, 2019.

[98] A. Tsvirkun, A. Rezchikov, L. Y. Filimonyuk, A. Samartsev, V. Ivashchenko, A. Bogomolov, and V. Kushnikov, "System of integrated simulation of spread of hazardous factors of fire and evacuation of people from indoors," *Automation and Remote Control*, vol. 83, no. 5, pp. 692–705, 2022.

[99] J. Barreiro-Gomez, S. E. Choutri, and H. Tembine, "Risk-awareness in multi-level building evacuation with smoke: Burj khalifa case study," *Automatica*, vol. 129, p. 109625, 2021.

[100] V. Karbovskii, D. Voloshin, A. Karsakov, A. Bezgodov, and A. Zagarskikh, "Multiscale agent-based simulation in large city areas: emergency evacuation use case," *Procedia Computer Science*, vol. 51, pp. 2367–2376, 2015.

[101] A. Mordvintsev, V. Krzhizhanovskaya, M. Lees, and P. Sloot, "Simulation of city evacuation coupled to flood dynamics," in *Pedestrian and Evacuation Dynamics 2012*, pp. 485–499, Springer, 2014.

[102] K. Chapuis, T. A. Elwaqoudi, A. Brugière, E. Daudé, A. Drogoul, B. Gaudou, D. Nguyen-Ngoc, H. Q. Nghi, and J.-D. Zucker, "An agent-based co-modeling approach to simulate the evacuation of a population in the context of a realistic flooding event: A case study in hanoi (vietnam)," in *Modelling, Simulation and Applications of Complex Systems: CoSMoS 2019, Penang, Malaysia, April 8-11, 2019*, pp. 79–108, Springer, 2021.

[103] W. van Toll, C. Braga, B. Solenthaler, and J. Pettré, "Extreme-density crowd simulation: Combining agents with smoothed particle hydrody-

namics," in *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pp. 1–10, 2020.

[104] N. Chooramun, P. Lawrence, and E. Galea, "Implementing a hybrid space discretisation within an agent based evacuation model," in *Pedestrian and Evacuation Dynamics* (R. D. Peacock, E. D. Kuligowski, and J. D. Averill, eds.), (Boston, MA), pp. 449–458, Springer US, 2011.

[105] R. Ekyalimpa, M. Werner, S. Hague, S. AbouRizk, and N. Porter, "A combined discrete-continuous simulation model for analyzing train-pedestrian interactions," in *2016 Winter Simulation Conference (WSC)*, pp. 1583–1594, IEEE, 2016.

[106] D. A. Quistberg, E. J. Howard, B. E. Ebel, A. V. Moudon, B. E. Saelens, P. M. Hurvitz, J. E. Curtin, and F. P. Rivara, "Multilevel models for evaluating the risk of pedestrian–motor vehicle collisions at intersections and mid-blocks," *Accident Analysis & Prevention*, vol. 84, pp. 99–111, 2015.

[107] G. Lämmel, M. Chraibi, A. K. Wagoum, and B. Steffen, "Hybrid multi-and inter-modal transport simulation: a case study on large-scale evacuation planning," in *Transportation Research Board 95th Annual Meeting*, Jülich Supercomputing Center, 2016.

[108] D. H. Biedermann, C. Torchiani, P. M. Kielar, D. Willems, O. Handel, S. Ruzika, and A. Borrmann, "A hybrid and multiscale approach to model and simulate mobility in the context of public events," *Transportation Research Procedia*, vol. 19, pp. 350–363, 2016.

[109] L. Crociani, G. Lämmel, J. Park, and G. Vizzari, "Cellular automaton based simulation of large pedestrian facilities—a case study on the staten island ferry terminals," in *96th Transportation Research Board annual meeting*, Jan. 2017.

[110] A. Ford, S. Barr, R. Dawson, J. Virgo, M. Batty, and J. Hall, "A multi-scale urban integrated assessment framework for climate change studies: A flooding application," *Computers, Environment and Urban Systems*, vol. 75, pp. 229–243, 2019.

[111] T. K. Lim, M. Ignatius, M. Miguel, N. H. Wong, and H.-M. H. Juang, "Multi-scale urban system modeling for sustainable planning and design," *Energy and Buildings*, vol. 157, pp. 78–91, 2017.

[112] M. Adnan, F. C. Pereira, C. M. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. Zegras, and M. Ben-Akiva, "Simmobility: A multi-scale integrated agent-based simulation platform," in *95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record*, 2016.

[113] Y. Yu, J. He, W. Tang, and C. Li, "Modeling urban collaborative growth dynamics using a multiscale simulation model for the wuhan urban agglomeration area, china," *ISPRS International Journal of Geo-Information*, vol. 7, no. 5, p. 176, 2018.

[114] T. Xu, J. Gao, G. Coco, and S. Wang, "Urban expansion in auckland, new zealand: a gis simulation via an intelligent self-adapting multiscale agent-based model," *International Journal of Geographical Information Science*, vol. 34, no. 11, pp. 2136–2159, 2020.

[115] J. Cheng and I. Masser, "Modelling urban growth patterns: a multiscale perspective," *Environment and Planning A*, vol. 35, no. 4, pp. 679–704, 2003.

[116] A. Veldkamp and L. Fresco, "Clue-cr: an integrated multi-scale model to simulate land use change scenarios in costa rica," *Ecological modelling*, vol. 91, no. 1-3, pp. 231–248, 1996.

[117] L. Yang, K. H. van Dam, B. Anvari, and L. Zhang, "Multi-level agent-based simulation for supporting transit-oriented development in beijing,"

in *International Workshop on Agent-Based Modelling of Urban Systems (ABMUS)*, p. 17, 2021.

[118] A. Drogoul, N. Q. Huynh, and Q. C. Truong, "Coupling environmental, social and economic models to understand land-use change dynamics in the mekong delta," *Frontiers in environmental science*, vol. 4, p. 19, 2016.

[119] J. Santiago, B. Sanchez, C. Quaassdorff, D. de la Paz, A. Martilli, F. Martín, R. Borge, E. Rivas, F. Gómez-Moreno, E. Díaz, *et al.*, "Performance evaluation of a multiscale modelling system applied to particulate matter dispersion in a real traffic hot spot in Madrid (Spain)," *Atmospheric pollution research*, vol. 11, no. 1, pp. 141–155, 2020.

[120] E. Moreira, S. Costa, A. P. Aguiar, G. Câmara, and T. Carneiro, "Dynamical coupling of multiscale land change models," *Landscape Ecology*, vol. 24, no. 9, pp. 1183–1194, 2009.

[121] K. P. Overmars and P. H. Verburg, "Multilevel modelling of land use from field to village level in the philippines," *Agricultural Systems*, vol. 89, no. 2-3, pp. 435–456, 2006.

[122] G. D'Angelo, S. Ferretti, and V. Ghini, "Multi-level simulation of internet of things on smart territories," *Simulation Modelling Practice and Theory*, vol. 73, pp. 3–21, 2017.

[123] G. D'Angelo, S. Ferretti, and V. Ghini, "Distributed hybrid simulation of the internet of things and smart territories," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 9, p. e4370, 2018.

[124] Z. Gu, F. Safarighouzhdi, M. Saberi, and T. H. Rashidi, "A macro-micro approach to modeling parking," *Transportation Research Part B: Methodological*, vol. 147, pp. 220–244, 2021.

[125] M. Magnani and L. Rossi, "The ml-model for multi-layer social networks," in *2011 International conference on advances in social networks analysis and mining*, pp. 5–12, IEEE, 2011.

[126] Y. Zhuang and O. Yağan, "Information propagation in clustered multilayer networks," *IEEE Transactions on Network Science and Engineering*, vol. 3, no. 4, pp. 211–224, 2016.

[127] Y. Murase, J. Török, H.-H. Jo, K. Kaski, and J. Kertész, "Multilayer weighted social network model," *Physical Review E*, vol. 90, no. 5, p. 052810, 2014.

[128] C. Ju, C. Wang, Y. Jiang, F. Bao, H. Zhou, and C. Xu, "Exploring a multi-layer coupled network propagation model based on information diffusion and bounded trust," *International Journal of Public Health*, p. 120, 2022.

[129] C. Ju, Y. Jiang, F. Bao, B. Zou, and C. Xu, "Online rumor diffusion model based on variation and silence phenomenon in the context of covid-19," *Frontiers in Public Health*, vol. 9, 2021.

[130] X. Zhang, S. Onufrak, J. B. Holt, and J. B. Croft, "A multilevel approach to estimating small area childhood obesity prevalence at the census block-group level," *Prev Chronic Dis*, vol. 10, no. 8, p. E68, 2013.

[131] P. Congdon, "A multilevel model for comorbid outcomes: obesity and diabetes in the us," *International Journal of Environmental Research and Public Health*, vol. 7, no. 2, pp. 333–352, 2010.

[132] V. M. Oguoma, A. E. Anyasodor, A. O. Adeleye, O. A. Eneanya, and E. C. Mbanefo, "Multilevel modelling of the risk of malaria among children aged under five years in Nigeria," *Transactions of The Royal Society of Tropical Medicine and Hygiene*, vol. 115, no. 5, pp. 482–494, 2021.

[133] G. Werneck, C. Costa, A. Walker, J. David, M. Wand, and J. Maguire, "Multilevel modelling of the incidence of visceral leishmaniasis in teresina, brazil," *Epidemiology & Infection*, vol. 135, no. 2, pp. 195–201, 2007.

[134] P. Banandur, U. Mahajan, R. S. Potty, S. Isac, T. Duchesne, B. Abdous, B. M. Ramesh, S. Moses, and M. Alary, "Population-level impact of avahan in karnataka state, south india using multilevel statistical modelling techniques," *JAIDS Journal of Acquired Immune Deficiency Syndromes*, vol. 62, no. 2, pp. 239–245, 2013.

[135] A. E. Iyanda and T. Osayomi, "Is there a relationship between economic indicators and road fatalities in Texas? a multiscale geographically weighted regression analysis," *GeoJournal*, vol. 86, no. 6, pp. 2787–2807, 2021.

[136] K. Ball, D. Crawford, and G. Mishra, "Socio-economic inequalities in women's fruit and vegetable intakes: a multilevel study of individual, social and environmental mediators," *Public health nutrition*, vol. 9, no. 5, pp. 623–630, 2006.

[137] A. K. Baumle and D. L. Poston Jr, "The economic cost of homosexuality: Multilevel analyses," *Social Forces*, vol. 89, no. 3, pp. 1005–1031, 2011.

[138] J. Gibbons, "Are gentrifying neighborhoods more stressful? a multilevel analysis of self-rated stress," *SSM-population health*, vol. 7, p. 100358, 2019.

[139] H. Gibbs Knotts and M. Haspel, "The impact of gentrification on voter turnout," *Social science quarterly*, vol. 87, no. 1, pp. 110–121, 2006.

[140] D. Courgeau and B. Baccaini, "Multilevel analysis in the social sciences," *Population: An English Selection*, pp. 39–71, 1998.

[141] M. Tian, Z. Tian, and W. Sun, "The impacts of city-specific factors on social integration of chinese migrant workers: A study using multilevel modeling," *Journal of Urban Affairs*, vol. 41, no. 3, pp. 324–337, 2019.

[142] J. Gil-Quijano, "Mechanisms of automated formation and evolution of social-groups: A multi-agent system to model the intra-urban mobilities of bogotá city," in *Social Simulation: Technologies, Advances and New Discoveries*, pp. 151–168, IGI Global, 2008.

[143] G. Gilioli and S. Pasquali, "Use of individual-based models for population parameters estimation," *Ecological modelling*, vol. 200, no. 1-2, pp. 109–118, 2007.

[144] M. Möhring, "Social science multilevel simulation with mimose," in *Social Science Microsimulation*, pp. 123–137, Springer, 1996.

[145] L. Billard, "On lotka–volterra predator prey models," *Journal of Applied Probability*, vol. 14, no. 2, pp. 375–381, 1977.

[146] C. O. Retzlaff, M. Ziefle, and A. Calero Valdez, "The history of agent-based modeling in the social sciences," in *International Conference on Human-Computer Interaction*, pp. 304–319, Springer, 2021.

[147] A. Hjorth, B. Head, C. Brady, and U. Wilensky, "Levelspace: A netlogo extension for multi-level agent-based modeling," *Journal of Artificial Societies and Social Simulation*, vol. 23, no. 1, 2020.

[148] U. Wilensky, "Netlogo." Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999. `http://ccl.northwestern.edu/netlogo/`.

[149] Y. H. Lee, M. K. Cho, S. J. Kim, and Y. B. Kim, "Supply chain simulation with discrete–continuous combined modeling," *Computers & Industrial Engineering*, vol. 43, no. 1-2, pp. 375–392, 2002.

[150] M. M. Rahman, R. Nguyen, and L. Lu, "Multi-level impacts of climate change and supply disruption events on a potato supply chain: An agent-based modeling approach," *Agricultural Systems*, vol. 201, p. 103469, 2022.

[151] A. Alho, B. Bhavathrathan, M. Stinson, R. Gopalakrishnan, D.-T. Le, and M. Ben-Akiva, "A multi-scale agent-based modelling framework for urban freight distribution," *Transportation Research Procedia*, vol. 27, pp. 188–196, 2017.

[152] J. Duggan, "A simulator for continuous agent-based modelling," in *25th International Conference of the Systems Dynamics Society Boston*, Citeseer, 2007.

[153] Q. Zhou, "Multi-layer affective computing model based on emotional psychology," *Electronic Commerce Research*, vol. 18, no. 1, pp. 109–124, 2018.

[154] L. Serena, M. Marzolla, G. D'Angelo, and S. Ferretti, "Design patterns for multilevel modeling and simulation," in *2023 IEEE/ACM 27th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pp. 48–55, IEEE, 2023.

[155] E. Gamma, R. Johnson, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.

[156] J. Zheng and K. E. Harper, "Concurrency design patterns, software quality attributes and their tactics," in *Proceedings of the 3rd International Workshop on Multicore Software Engineering*, pp. 40–47, 2010.

[157] A. Sharma, M. Kumar, and S. Agarwal, "A complete survey on software architectural styles and patterns," *Procedia Computer Science*, vol. 70, pp. 16–28, 2015.

[158] M. Xiong, S. Tang, and D. Zhao, "A hybrid model for simulating crowd evacuation," *New Generation Computing*, vol. 31, no. 3, pp. 211–235, 2013.

[159] D. R. Jefferson, "Virtual time," *ACM Trans. Program. Lang. Syst.*, vol. 7, p. 404–425, jul 1985.

[160] P. A. Mboup, K. Konaté, and J. Le Fur, "A multi-world agent-based model working at several spatial and temporal scales for simulating complex geographic systems," *Procedia Computer Science*, vol. 108, pp. 968–977, 2017.

[161] S. R. Musse and D. Thalmann, "Hierarchical model for real time simulation of virtual human crowds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 2, pp. 152–164, 2001.

[162] B. Liskov and L. Shrira, "Promises: Linguistic support for efficient asynchronous procedure calls in distributed systems," *SIGPLAN Not.*, vol. 23, p. 260–267, jun 1988.

[163] G. V. Bobashev, D. M. Goedecke, F. Yu, and J. M. Epstein, "A hybrid epidemic model: combining the advantages of agent-based and equation-based approaches," in *2007 winter simulation conference*, pp. 1532–1537, IEEE, 2007.

[164] P. Mathieu, G. Morvan, and S. Picault, "Multi-level agent-based simulations: Four design patterns," *Simulation Modelling Practice and Theory*, vol. 83, pp. 51–64, 2018.

[165] Y. Labiche and G. Wainer, "Towards the verification and validation of devs models," in *in Proceedings of 1st Open International Conference on Modeling & Simulation*, pp. 295–305, Citeseer, 2005.

[166] G. A. Wainer, *Discrete-event modeling and simulation: a practitioner's approach*. CRC press, 2017.

[167] A. Maatoug, G. Belalem, and K. Mostefaoui, "Modeling and simulation of energy management system for smart city with the formalism devs: Towards reducing the energy consumption," *International Journal of Computer Applications*, vol. 90, no. 18, 2014.

[168] A. C. H. Chow and B. P. Zeigler, "Parallel devs: A parallel, hierarchical, modular modeling formalism," in *Proceedings of Winter Simulation Conference*, pp. 716–722, IEEE, 1994.

[169] A. Steiniger, F. Krüger, and A. M. Uhrmacher, "Modeling agents and their environment in multi-level-devs," in *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pp. 1–12, IEEE, 2012.

[170] P. Marcenac and S. Giroux, "Geamas: A generic architecture for agent-oriented simulations of complex processes," *Applied Intelligence*, vol. 8, no. 3, pp. 247–267, 1998.

[171] G. Morvan and D. Jolly, "Multi-level agent-based modeling with the influence reaction principle," *arXiv preprint arXiv:1204.0634*, 2012.

[172] S. Picault, P. Mathieu, and Y. Kubera, "Padawan, un modèle multi-échelles pour la simulation orientée interactions," in *Systèmes Multi-agents, Défis Sociétaux - JFSMA 10 - Dix-huitièmes journées francophones sur les systèmes multi-agents, Mahdia, Tunisia, October 18-20, 2010* (M. Occello and L. Rejeb, eds.), pp. 195–204, Cepadues Editions, 2010.

[173] A. Grignard, P. Taillandier, B. Gaudou, D. A. Vo, N. Q. Huynh, and A. Drogoul, "Gama 1.6: Advancing the art of complex agent-based modeling and simulation," in *International conference on principles and practice of multi-agent systems*, pp. 117–131, Springer, 2013.

[174] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *International conference on complex systems*, vol. 21, pp. 16–21, Boston, MA, 2004.

[175] S. Khoirom, M. Sonia, B. Laikhuram, J. Laishram, and T. D. Singh, "Comparative analysis of python and java for beginners," *Int. Res. J. Eng. Technol*, vol. 7, no. 8, pp. 4384–4407, 2020.

[176] L. Serena, M. Zichichi, G. D'Angelo, and S. Ferretti, "On the modeling of p2p systems as temporal networks: a case study with data streaming," in *2022 Annual Modeling and Simulation Conference (ANNSIM)*, pp. 66–77, IEEE, 2022.

[177] A. Malatras, "State-of-the-art survey on p2p overlay networks in pervasive computing environments," *Journal of Network and Computer Applications*, vol. 55, pp. 1–23, 2015.

[178] L. Serena, M. Zichichi, G. D'Angelo, and S. Ferretti, "Simulation of dissemination strategies on temporal networks," in *2021 Annual Modeling and Simulation Conference (ANNSIM)*, pp. 1–12, IEEE, 2021.

[179] Y. Song, H. Ni, and X. Zhu, "Analytical modeling of optimal chunk size for efficient transmission in information-centric networking," *Int. J. Innov. Comput. Inf. Control*, vol. 16, pp. 1511–1525, 2020.

[180] D. Thiele and R. Ernst, "Formal worst-case performance analysis of time-sensitive ethernet with frame preemption," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–9, IEEE, 2016.

[181] M. Zichichi, L. Serena, S. Ferretti, and G. D'Angelo, "Dlt-based data mules for smart territories," in *2022 International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–7, IEEE, 2022.

[182] G. Anastasi, M. Conti, and M. Di Francesco, "Data collection in sensor networks with data mules: An integrated simulation analysis," in *2008 IEEE Symposium on Computers and Communications*, pp. 1096–1102, IEEE, 2008.

[183] M. M. Coutinho, A. Efrat, T. Johnson, A. Richa, and M. Liu, "Healthcare supported by data mule networks in remote communities of the amazon region," *International scholarly research notices*, vol. 2014, 2014.

[184] A. Balasundram, T. Samarasinghe, and D. Dias, "Performance analysis of wi-fi direct for vehicular ad-hoc networks," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–6, IEEE, 2016.

[185] L. Serena, M. Zichichi, G. D'Angelo, and S. Ferretti, "Simulation of hybrid edge computing architectures," in *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pp. 1–8, IEEE, 2021.

[186] V. Karagiannis, A. Venito, R. Coelho, M. Borkowski, and G. Fohler, "Edge computing with peer to peer interactions: Use cases and impact," in *Proceedings of the Workshop on Fog Computing and the IoT*, pp. 46–50, 2019.

[187] G. Yadgar, O. Kolosov, M. F. Aktas, and E. Soljanin, "Modeling the edge: Peer-to-peer reincarnated," in *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[188] S.-W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, 2018.

[189] E. Hyytiä and J. Virtamo, "Random waypoint mobility model in cellular networks," *Wireless Networks*, vol. 13, no. 2, pp. 177–188, 2007.

[190] N. Vastardis and K. Yang, "An enhanced community-based mobility model for distributed mobile social networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 1, pp. 65–75, 2014.

[191] T. Voigt, M. Bor, U. Roedig, and J. Alonso, "Mitigating inter-network interference in lora networks." arXiv preprint arXiv:1611.00688, 2016.

[192] S. Javed and D. Zorbas, "LoRaWAN Downlink Policies for Improved Fairness," in *IEEE Conference on Standards for Communications and Networking (CSCN '22)*, pp. 1–6, IEEE, Nov. 2022.

[193] B. Lashkari and P. Musilek, "A comprehensive review of blockchain consensus mechanisms," *IEEE Access*, vol. 9, pp. 43620–43652, 2021.

[194] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "Simblock: A blockchain network simulator," in *Proc. of the 2nd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, CryBlock'19, IEEE, 2019.

[195] M. Alharby and A. van Moorsel, "Blocksim: A simulation framework for blockchain systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 46, pp. 135–138, Jan. 2019.

[196] E. C. Harillo and F. Freitag, "Loracoin: Towards a blockchain-based platform for managing lora devices," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–2, IEEE, 2022.

[197] K. R. Ozyilmaz and A. Yurdakul, "Designing a blockchain-based iot with ethereum, swarm, and lora: The software solution to create high availability with minimal security risks," *IEEE Consumer Electronics Magazine*, vol. 8, no. 2, pp. 28–34, 2019.

[198] J. Lin, Z. Shen, C. Miao, and S. Liu, "Using blockchain to build trusted lorawan sharing server," *International Journal of Crowd Science*, vol. 1, no. 3, pp. 270–280, 2017.

[199] L. Felli and R. Giuliano, "Access control in woodland through blockchain and lorawan," in *2021 AEIT International Conference on Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, pp. 1–5, IEEE, 2021.

[200] S. M. Danish, M. Lestas, W. Asif, H. K. Qureshi, and M. Rajarajan, "A lightweight blockchain based two factor authentication mechanism for lorawan join procedure," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, IEEE, 2019.

[201] K. R. Özyilmaz, M. Doğan, and A. Yurdakul, "Idmob: Iot data marketplace on blockchain," in *2018 crypto valley conference on blockchain technology (CVCBT)*, pp. 11–19, IEEE, 2018.

[202] M. Grebovic, T. Popovic, and R. S. Grebovic, "Blockchain technology for weather data management," in *2023 22nd International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1–6, IEEE, 2023.

[203] B. Dammak, M. Turki, S. Cheikhrouhou, M. Baklouti, R. Mars, and A. Dhahbi, "Lorachaincare: An iot architecture integrating blockchain and lora network for personal health care data monitoring," *Sensors*, vol. 22, no. 4, p. 1497, 2022.

[204] I. Vlachos and N. Harziargyriou, "Design and implementation of a decentralized amr system using blockchains, smart contracts, and lorawan," in *proc. 25th International Conference on Electricity Distribution*, AIM, 2019.

[205] L. Gigli, F. Montori, M. Zichichi, L. Bedogni, S. Ferretti, and M. Di Felice, "On the decentralization of mobile crowdsensing in distributed ledgers: an architectural vision," in *Proc. of the IEEE Consumer Communications & Networking Conference (CCNC 2024)*, (Las Vegas, USA), IEEE ComSoc, January 2024.

[206] M. Shahjalal, M. M. Islam, M. M. Alam, and Y. M. Jang, "Implementation of a secure lorawan system for industrial internet of things integrated with ipfs and blockchain," *IEEE Systems Journal*, 2022.

[207] S. Musso, G. Perboli, M. Rosano, and A. Manfredi, "A decentralized marketplace for m2m economy for smart cities," in *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pp. 27–30, IEEE, 2019.

[208] M. G. Ikhsan, M. Y. A. Saputro, D. A. Arji, R. Harwahyu, and R. F. Sari, "Mobile lora gateway for smart livestock monitoring system," in *2018*

IEEE International Conference on Internet of Things and Intelligence System (IOTAIS), pp. 46–51, IEEE, 2018.

[209] S. Sugianto, A. Al Anhar, R. Harwahyu, and R. F. Sari, "Simulation of mobile lora gateway for smart electricity meter," in *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pp. 292–297, IEEE, 2018.

[210] L. Sciullo, A. Trotta, and M. Di Felice, "Design and performance evaluation of a lora-based mobile emergency management system (locate)," *Ad Hoc Networks*, vol. 96, p. 101993, 2020.

[211] L. Hou, K. Zheng, Z. Liu, X. Xu, and T. Wu, "Design and prototype implementation of a blockchain-enabled lora system with edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, 2020.

[212] D. Krajzewicz, "Traffic simulation with sumo–simulation of urban mobility," *Fundamentals of traffic simulation*, pp. 269–293, 2010.

[213] T. Yatagan and S. Oktug, "Smart spreading factor assignment for lorawans," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–7, IEEE, 2019.

[214] L. Serena, G. D'Angelo, and S. Ferretti, "Security analysis of distributed ledgers and blockchains through agent-based simulation," *Simulation Modelling Practice and Theory*, vol. 114, p. 102413, 2022.

[215] M. R. Villarim, J. V. H. de Luna, D. de Farias Medeiros, R. I. S. Pereira, C. P. de Souza, O. Baiocchi, and F. C. da Cunha Martins, "An evaluation of lora communication range in urban and forest areas: A case study in brazil and portugal," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2019.

[216] N. Z. Tomić, "A review of consensus protocols in permissioned blockchains," *Journal of Computer Science Research*, vol. 3, no. 2, pp. 19–26, 2021.