UNIVERSITÁ DI ROMA
"TOR VERGATA"

FACOLTÁ DI INGEGNERIA

DIPARTIMENTO DI INFORMATICA, SISTEMI e PRODUZIONE

P H D T H E S I S

to obtain the title of

PhD of Science

Specialty : Computer Science

Agent Based Modeling and Simulation for critical and interdependent systems

Defended by

Emanuele Galli

Course: XXII
A.Y. 2009/2010

Thesis Advisor
Prof. Salvatore Tucci

Coordinator
Prof. Daniel P. Bovet

To Michela and my Family

Acknowledgments

First, I would like to sincerely thank my advisors Salvatore Tucci and Emiliano Casalicchio.

Salvatore, from his personal big experience, gave me always excellent suggestions to my research and also to other topics external to my own area.

Emiliano has been my personal inspiration and helped me soon to find the correct direction in my research field. We have worked together in many topics and we have obtained together really good results. Today I consider Emiliano a friend and I hope to keep working and having other extraordinary experiences with him in the future.

I would also thank my university friends with whom we have shared happiness, gaiety, satisfactions, but also, sometimes, disappointment and sadness moments. In particular I would remind my friends Emiliano Betti, Francesca Fallucchi, Cristina Giannone, Fabio Dellutri, Gianluca Grilli, Vittorio Ottaviani, Danilo Croce and Diego De Cao.

A special thank goes to Donato Griesi, who I will never forget for the rest of my life and keeps living in my heart and in my thoughts .

I spent an unforgettable year in Los Alamos National Laboratory, working with Stephan Eidenbenz, who has been my mentor and co-author. Here I met really brilliant young people from all the world and I would like to express my heartfelt thanks to Ha Duc, Chiara Pozzi, Milan Bradonjić, Guanhua Yan, Shiva Kasiviswanathan and Andrea Asztalos.

Finally, I would like to thank my Love Michela, my brother Massimiliano, my parents Emilia and Antonio, Michela's parents Serenella and Giulio, for their continuous love and support.

Agent Based Modeling and Simulation for critical and interdependent systems

Abstract: The research community has been just recently attracted by the study of critical infrastructure. All related topics can be grouped in the so called "critical infrastructure protection" (CIP). Lewis[1] defines the study of CIP as "the study of challenges to be met and solutions to be found". He also divides the challenges of CIP in seven possible categories, which are:

1. Vastness: related to the vastness of problem, which renders impractical to protect all infrastructures;

2. Command: associated to the problem to define who takes the last decision;

3. Information Sharing: the absence of a clear way to share and distribute information among different infrastructures made data completely incompatible;

4. Knowledge: every infrastructure has its own domain and technology, so it is very hard to have a whole knowledge of a so vast complex system;

5. Interdependencies: every infrastructure depends on many other ones directly or indirectly. Dependencies are caused by human organizational structures as well as physical linkage between components;

6. Inadequate Tools: there is not yet a general approach or tool to study critical infrastructure;

[1]Presented in Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation, di Ted G. Lewis

7. Asymmetric Conflict: small attack can produce big damages.

Such scenario has also attracted us specifically in the study of human and physical interdependencies, their own valuation and quantification. In particular, the goal of this thesis has been to provide a framework for the simulation and analysis of physical, geographical, informational and temporal interdependencies using the agent based modeling and simulation approach with the theory and architecture of distributed simulation in order to allow the reuse of already implemented simulators as well as to increase performances and to scale the problem. We have also used the micro-simulation as an alternative approach to the study of critical infrastructure. For such aim, we have implemented a simulator of road-traffic using the parallel and discrete events approach in order to simulate daily traffic in big cities and to evaluate how other infrastructures and individuals depend on transportation system. The major contributions are as in the following:

- We introduce a new model to simulate and analyze critical infrastructures and their interdependencies using the agent based modeling and simulation. An agent is an entity which has a specific behavior that can be influenced by the environment, the memory and experience of the agent, can interact with the environment and other agents (heterogeneous and homogenous) to reach the same goal, has a specific geographical position. The agent based modeling and simulation has been used to simulate and to define physical and geographical interdependencies.

- We have used the parallel and distributed simulation to reuse already implemented and well-tested specific sector simulators as well as to distribute the load and increase both performances and scalability. Such characteristics allow to simulate a big scenario composed of thousands and thousands of components and

multiple infrastructures at the same time. Moreover we have used a standard as the High Level Architecture so that the framework can be easily extended with new sector simulators. Information interdependency as well as physical one is simulated directly by the sector simulators.

- We have used the standard representation to geo-reference objects in order to create realistic scenario and reuse real stakeholders data.

- We have also considered the workload generated by the people on the infrastructure networks during their regular activities while is still a big challenge to provide the workload during catastrophic events.

- We have developed a parallel and scalable micro-simulator for transportation network which uses the discrete-event queue model which uses the workload generated by the daily activities simulator.

- We have introduced some new metrics to measure direct and indirect interdependencies using collected data from sector simulators. Such metrics are really helpful for managers who have to take important decisions to prevent catastrophic events and to reduce the risk of threats.

Keywords: Critical Infrastructure, Agent Based Modeling and Simulation, parallel and distributed simulation, HLA-RTI

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivation

The current industrial and technology development of a country relies on availability and correct operation of multiple infrastructures. If we take the Oxford dictionary, an infrastructure is defined as "*the basic physical and organizational structures and facilities (e.g., buildings, roads, and power supplies) needed for the operation of a society or enterprise*".

Even though most of dictionaries have a similar definition, the political institutions have often changed and modified their own definition in the last twenty years. This is principally due to the fact that we have seen an escalation of terroristic attacks especially in

regard to destructive effects. Most of us still remember tragic moments of 2001 in New York, 2004 in Madrid and 2005 in London. During such attacks most of infrastructures have been seriously affected either directly or indirectly. So, if at the beginning engineers as well as politicians were concerned about the quality and adequacy of an infrastructure, today they are more interesting in their protection. Protection today does not mean only to prevent to be attacked, but, especially, to ensure a continuos flow of services for every single infrastructure and to mitigate the risk that if an infrastructure cannot offer the standard level of service, all other infrastructure are minimally affected and can keep operating.

On August 14, 2003 failures of some software system was one of contribution to the blackout in Northeast and Midwestern USA and in Ontario, Canada [66]. Even though some essential services remained in operation, cell phones system, water system and rail system were heavily affected. Though everyone can generally imagine that a blackout could affect other infrastructures, it is much harder to imagine the opposite. On January 25, 2003 the SQL Slummer warm [90] started to spread around the world infecting more than 75.000 hosts in the first 10 minutes and caused big slowing downs of Internet. Successively it was able to penetrate and disable a safety monitoring system of Davis-Besse nuclear power plant. Such a case puts in evidence the importance of cyber-security today. In 2007 one of the biggest Distributed Denial of Service (DDoS) attack in Estonia [4], one of the most computerized country in the world, obligated the government to shut down several of its own key systems. Since the current Estonia's government and economy is based on high tech technologies (citizens can ballot at home using Internet), banks, ministries, newspapers and broadcasters were put out of service.

Such threats have obligated countries to draft some shared documents and to create a plan to protect all infrastructures. The USA patriot act of 2001[12] and, successively in 2004, the USA 's Congres-

sional Research Service drafted a report [61] to define and identify current critical infrastructures. In like manner during the 2006 the Commission of European Communities distributed a document [16] for defining an overall strategy among all countries of the community for the protection of Critical Infrastructures. Even though the cited reports as well as other ones of different nations do not identify same infrastructures, there is an important fact shared by all countries: if twenty years ago infrastructures were considered only public works of a nation, today they contemplate both public and private work of different sectors.

Even the research community has been recently attracted by the study of critical infrastructure. All related topics can be grouped in the so called "critical infrastructure protection" (CIP). In [87] Lewis defines the study of CIP as "the study of challenges to be met and solutions to be found". He also divides the challenges of CIP in seven possible categories which are:

1. Vastness: related to the vastness of problem which renders impractical to protect all infrastructures;

2. Command: associated to the problem to define who takes the last decision;

3. Information Sharing: the absence of a clear way to share and distribute information among different infrastructures made data completely incompatible;

4. Knowledge: every infrastructure has its own domain and technology, so it is very hard to have a whole knowledge of a so vast complex system;

5. Interdependencies: every infrastructure depends on many other ones directly or indirectly. Dependencies are caused by human

organizational structures as well as physical linkage between components;

6. Inadequate Tools: there is not yet a general approach or tool to study critical infrastructure;

7. Asymmetric Conflict: small attack can produce big damages.

Such scenario has also attracted us specifically in the study of human and physical interdependencies, their own valuation and quantification. Moreover in this thesis we want to provide a tool as general as possible which can be used in most of scenarios.

Before presenting contributions of this thesis, we introduce some of already proposed models and solutions that we can find in literature.

1.2 Critical infrastructures models and approaches

The goal of critical infrastructure model is to study their interdependency so that we can prevent both direct and cascade effects. Taking into account of the proposed categorization of Rinaldi (2001-2004) [74] dependencies can be classified in terms of: physical, cyber, geographic, or logical dependency. Physical interdependencies is related to the production of materials or services used by an other one. The risk of failure from normal operating conditions in one infrastructure will be a function of risk in another infrastructure. Cyber interdependencies occur when state of an infrastructure depends on information transmitted through the information infrastructure. This is the type of complex system whereby control of a networked system is dependent upon the transmission of information. Geospatial interdependencies involve the physical proximity of one infrastructure

to another. An event such as a catastrophic event in an urban area could create correlated disruptions with other infrastructures, such as communication and electric services to a community. Logical ones are all other kind of interdependencies. They could be, i.e., economic or political.

A revision of Dudenhoeffer, Permann and Boring (2004) called logical interdependencies as policy ones, that occur when there is a binding of infrastructure components due to policy or high level decisions.

Finally the same Dudenhoeffer and Permann with Manic in 2006, aside the already presented interdependencies, introduced a new one that is the societal interdependency. Differently from the policy interdependency, the societal interdependency refers principally to interdependencies or influences than an event on an infrastructure component may impact on societal factors such as public opinion, public confidence, fear and cultural issues.

Interdependencies analysis depends also on temporal and economy scales. In case of temporal scale the dynamics of infrastructures varies from milliseconds (e.g. power plants and network communications) to hours (e.g. water systems). In case of economy scale it measures how much the cost of an infrastructure depends on an other ones and how the economy could be affected.

Follows a brief description of some related works that uses analytic and/or simulated approach for the analysis of interdependencies.

1.2.1 Analytic approach

Analytic approaches give us a higher overview of the problem and generally is not possible to evaluate with such methodologies all kinds of interdependencies. So we should use more than one only analytic approach to cover all problems. In literature we can find several analytic approaches that are from completely different domain like

network, electric, economy, etc...

For example, the topological approaches adopt models of interdependencies between nodes at level of topology and their structural properties. They use the so called perimetring procedure to individuate and identify the set of network of an infrastructure A (auxiliary network) necessary to the infrastructure B (behaving network). Topological approaches necessity stakeholders data to construct the two networks. The topological approaches measure the probability that a node of B fails when a node of B is removed. The average of such metrics should give us the robustness of a network.

Other approaches like the holistic ones, rather than to decompose the infrastructure into its components in order to analyze their interactions, analyze each infrastructure assumed as a single entity and try to emphasize how the behavior of a single infrastructure is able to affect the behavior of other infrastructures.

These approaches are generally more abstract, simplified and qualitative than the previous ones. They can be set-up more easily even if they appears not useful from an operative point of view. Moreover, while the topological approaches generally refer to the coupling between only two infrastructures, this type of approach is naturally designed to consider in the same time more infrastructures.

The holistic approaches appear more able to analyze interdependencies in terms of services exchange rather than the topological formulation (that are, on the contrary, more effective to analyze physical interdependencies). Hence they are able to provide ordinary indicators more useful for strategic analysis rather than for operational plans.

1.2.2 Simulation approach

The simulation approach allows to model, in deeper details, critical infrastructures coupling and their behavior.

Theoretically, a simulation model allows to do the measurement of every observable value and then every modeled interdependency. For example, if we have geo-referenced infrastructures, it is possible to model geographical inter-dependencies in case of natural disasters and, hopefully, to quantify them with an appropriate metric.

Simulation approach can be subdivided in base of time models (discrete, continuos), if they use time-stepped or discrete-events approach, in base of representation models like entities or agents, in base of level of detail, and if they are serial, parallel or distributed programs. During the selection of a simulator, we should always consider which characteristic are important for us: performance, accuracy, usability, distributability, etc..

1.3 Contribution

The goal of this thesis is to provide a framework for the simulation and analysis of physical, geographical, informational and temporal interdependencies using the agent based modeling and simulation approach with the theory and architecture of distributed simulation to allow the reuse of already implemented simulators as well as to increase performances and to scale the problem. We have also used the micro-simulation approach to simulate road-traffic using a parallel and discrete events approach in order to simulate daily traffic in big cities and to evaluate how other infrastructures as well as individuals depend on transportation system. The major contributions are as in the following:

- We introduce a new model to simulate and analyze critical infrastructures and their interdependencies using the agent based modeling and simulation. An agent is an entity which has a specific behavior that can be influenced by the environment, the memory and experience of the agent, can interact with the en-

vironment and other agents (heterogenous and homogenous) to reach the same goal, has a specific geographical position. The agent based modeling and simulation has been used to simulate and to define physical and geographical interdependencies.

- We have used the parallel and distributed simulation to reuse already implemented and well tested specific sector simulators as well as to distribute the load as well as to increase both performances and scalability. Such characteristics allow to simulate a big scenario composed of thousands and thousands of components and multiple infrastructures at the same time. Moreover we have used a standard as the High Level Architecture so that the framework can be easily extended with new sector simulators. Information interdependency as well as physical one is simulated directly by the sector simulators.

- We have used the standard representation to geo-reference objects in order to create realistic scenario and reuse real stakeholders data.

- We have also considered the workload generated by the people on the infrastructure networks during their regular activities while is still a big challenge to provide the workload during catastrophic events.

- We have developed a parallel and scalable micro-simulator for transportation network which uses the discrete-event queue model which uses the workload generated by the daily activities simulator.

- We have introduced some metrics to measure direct and indirect interdependencies using collected data from sector simulators. Such metrics are really helpful for managers who have to

take important decisions to prevent catastrophic events and to reduce the risk of threats.

1.4 Outline

The thesis is organized as in the following. The chapter 2 presents deeply some of the most famous related works. We present both analytic and simulated approach for the analysis of interdependencies.

The chapter 3 is dedicated to definition and formalization of an architecture that can embrace so many different domains. To overcome this difficulty we have used agent based modeling and simulation approach together with parallel and distributed simulation technique. The whole system is simulated as a composition of multiple systems that can communicate, exchange information and send events each other. Every system is an agent while a specific sector simulator simulates the agent behavior. Agents and specific sector simulators are synchronized thanks to distributed simulation approach that permits to put together heterogeneous simulators as well as to distribute and to balance the load. WIth such an architecture, domain experts have to be concentrated principally only on their domains defining correctly what their system provide and what they need from other ones.

The chapter 4 presents an implementation of the proposed framework, which federates Repast, an agent-based simulation engine and OMNeT++ an IT systems and communication networks modeling and simulation environment. We have used both the High Level Architecture and an own architecture to federate the two simulators.

In chapters 5 and 6 we present the micro-simulation technique applied to the individual daily activities generation as well as to the transportation network. The micro-simulation is an other technique deeply based on parallel simulation in order to maximize the level

of details to the detriment of necessary effort to develop such kind of system and hypothetical performance issues. ActivitySim is the simulator of daily activities. In ActivitySim, activities are generated using a synthetic-population, meta-heuristic algorithms that take into consideration the current position, the last performed activity, priorities of activities and some constraints (e.g. duration and maximum, minimum time). Daily activities have been used to generate the workload for the transportation network simulator FastTrans. The FastTrans has been used to evaluate some optimization algorithm for the shortest path as well as some partitioning schemas for load balancing, including geographic partitioning (that assigns simulation entities that are geographically close by to the same processor) and scattering (that assigns geographically close entities to different processors).

In the seventh chapter we present metrics and statistics to evaluate direct and indirect interdependencies. We propose a taxonomy of Interdependencies Quantification Metrics (IQM) which uses informations we have obtained from our simulations to measure some performance indexes that allow to risk analyst and domain experts to evaluate the goodness of mechanism and/or strategies designed in order to increase critical infrastructure protection and resilience.

In the last chapter we present conclusions and challenges for future works.

Chapter 2

Models for critical interdependent infrastructure analysis

Contents

If we take into consideration the definition of DoD Modeling and Simulation Glossary, a model is defined as "A physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process". An implementation of a model is called a simulation.

In this chapter we want to provide a review of related works on modeling and simulation of critical infrastructures and, in particular, of interdependencies models and tools . The study of critical infrastructures interdependency can be described by a general framework as illustrated in figure 2.3. When we want to model interdependencies and successively to simulate and analyze simulation-results, we need significant amount of data, a geographical representation and correlation between geographical position and infrastructures components, some visualization components (or GUI tools) that allow users to interact with our models and, finally, we need some utility to apply some experimental scenario.

Infrastructure Models The modeling of a single infrastructure like electricity or telecoms networks are mature field. One of the biggest issue is to find a way to get correct information from expert domains to gain an understanding of infrastructure interdependencies and risks.

Interdependency Modeling The modeling of interdependencies can be done at different level of abstraction (from high level abstraction to detailed models) as well as at different perceived model (physical, logical, cybernetic and so on). For each abstraction layer can be used a wide range of models based on several theories that

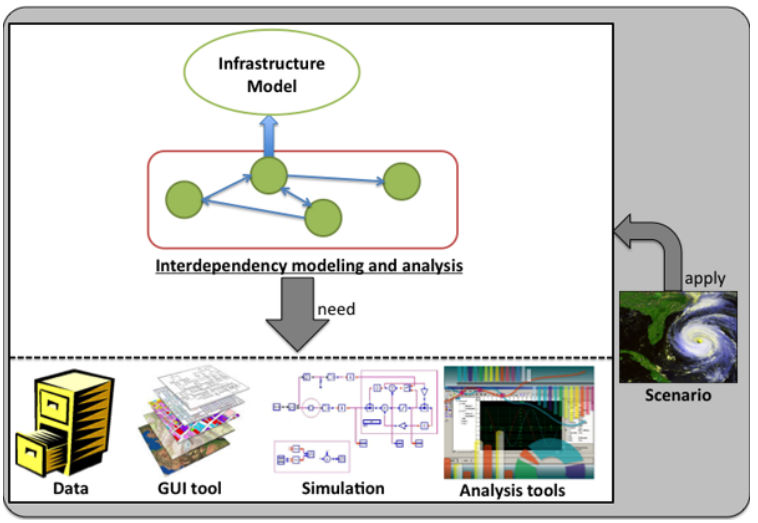


Figure 2.1: Critical Infrastructures modeling

can be range from qualitative models, stochastic activity networks to high fidelity simulation. At the same time every approach can be deployed at a several level of details, varying from detailed topology to service topology and cascading effects.

Simulators We have seen above that a simulation is actually an execution of a model over a period of time. It could be interactive or not. In the first case the simulation time has to coincide with the real time. In case the simulator takes and returns identical data to real data we can speak more of emulation. Generally simulators can be classified in base of what is simulated and what is virtual:

- Live simulation (where real people use simulated equipment in the real world)
- Virtual simulation (where real people use simulated equipment in a simulated world, or virtual environment) and

- Constructive simulation (where simulated people use simulated equipment in a simulated environment).

Since a simulator is a software, when we build a simulator we have to consider what kind of software paradigm we have to use. If we need rapidly a simulator that does not need high performance capability, we could implement it serially. In case we need high performance (consider simulation of millions of atoms) we should implement it using parallel programming. In such case, time synchronization of simulated events is crucial to avoid the loose of realism. Finally, in case we want to use multiple heterogenous simulators, we have to use distributed simulation. In such case, aside from considering events synchronization, we need to use a standard language and representation of data to publish and subscribe for sharing data among federates.

Data This refers to the data is collected from stakeholders and is used for the simulation of our models. Data can be static (i.e. geographical position) or dynamic (i.e. direction of wind during a day). Data verification is an important challenge since incorrect data could successively mislead the analysis. Apart from validating the data, when we have a big amount of data, good data mining approaches are necessary to get only the necessary informations.

Visualization, GUI tools GUI tools are necessary both during the modeling and data analysis. An example of visualization is the Geographical Information Systems (GIS) to represent both point, lines and more complex figures on a map. Information are often layered, specially when we have to work with so many abstraction layers (i.e. physical and logical layer) as well as with so different domains (i.e. electrical and telecommunication network).

Scenario After that we have chosen the model to use, we have implemented it with a simulator and we have been able to collect all data that we need, we have to allow stakeholders/users to apply some specific scenario that they want to simulate. For example, if we have build a simulator to simulate infrastructures inside our city, we could allow users to experiment some failure in a particular neighborhood to evaluate and analyze the effect of such scenario in the surrounding zones or in all the area under study.

2.1 Analytic Models

With the increasing interest from the research communities of the critical infrastructures , there has been an increment of proposed models for infrastructures. Currently there is not a shared and precise classification of all proposed analytic models and, may be worst, there is not a comparison of such models that allows us to classify which model is the best choice for a particular case study or which model we should use to simulate a certain type of scenario. So the choice of the best model remains an open question [19]. Due to the complexity of critical infrastructure model, every model is based on different simplifications of the mathematical model or the input parameters set. Such choices influence both the level of details that are simulated and scenarios that can be simulated. Moreover the model during the implementation could have further simplification to avoid some performance as well as numerical issues and such information should be included in a compendium [63].

Starting from the work [21] we give an overview of common infrastructure models that differentiate each others for the abstraction level and model boundaries, for the underlying theories and for the model applicability. Finally we will present some of the most successful project that we can find in literature presenting their maturity and

validation level.

2.1.1 Qualitative and semi-qualitive models

The qualitative and semi-qualitative models is based on continuos time stochastic processes as well as on Stochastic-Activity-Network (SAN) theory. Activity networks are used in cognitive science and cognitive engineering to show the time dependencies and flow of control when multiple parallel resources are required [65]. Moreover they are used to simulate and study common mode failures (CMF). CMF occurs when events are not statistically independent. A classic example of CMF is the calculation of probability that if a hard-disk inside, for example, a RAID-1 fails then also the second will fail. Obviously if we consider the two events as independent, the probability of two failure at the same time will be the multiplication of two probabilities. But such probability would increase if, for example, the two HDs were built by the same company, during the same period of production (serial numbers are almost the same) and so on[41].

In the case of critical infrastructure SAN are used to estimate the availability and reliability of network components when there is a failure in a certain component. Qualitative models can provide useful support to identify and analyze dependency at high level and to provide stochastic measures of the durations of events as well as the likelihood of the occurrence. Qualitative models can be implemented by means of already implemented tools like MODAF, ASCE or Mobius tools.

2.1.2 Input-output models

Input-output models (IOM), also know as Leontief's model, has been widely used in economy to predict the effect of changes in one industry on others and by consumers, government, and foreign suppliers on

the economy. With IOM a system-wide solution can be determined for the cascading effects caused by a single perturbation. If for example the operability of one producer decreases by a certain amount, this model can calculate how the operability of all interconnected producers is affected (including an amplification of the inoperability of the originally affected producer)[72]. In [49] the output is the inoperability that can be triggered by one or multiple failures due to their inherent complexity or to external perturbations (e.g., natural hazards, accidents, or acts of terrorism). In this approach the time is divided in several frames and in each frame is calculated a conceptual situation of equilibrium after an attack or other catastrophic events that affect the infrastructures. With such approach a qualitative analysis of short long term effect can be conducted.

In [49] the IOM model is applied to examine and predict the effects of changes in one infrastructure system on other ones. The limit of IOM is not able to describe the details of complex system evolution and to predict all basic behaviors of interdependent infrastructures [54]. To overcome such limit, generic cascading model [81] has been proposed which is a continuos model that considers also the recovery capacity of system components and time-delayed interaction among the nodes. In [69] the model was used to asses also the efficiency of different recovery strategies.

GCM seems to be more appropriate for studying spreading behavior in large networks.

2.1.3 Indicative system dynamics models

System dynamics model was created by Professor Jay Forrester during 1950s at the MIT. It is a top-down approach and is generally used to study the behavior of a complex system over time. It uses feedback loops and time delays to study the whole complex system. All dynamics in a system are assumed to arise from the interaction of two

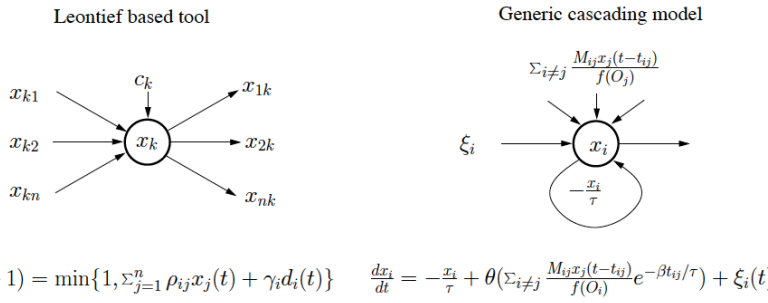


Figure 2.2: The Leontief and general cascading models [22]

types of feedback loops, positive and negative ones[86]. These loops are represented in loop diagrams. A diagram includes stocks (the accumulation of resources in a system), flows (the rates of change that alter those resources) and information (about the value influences based on changes in the regarded stocks). Changes in stocks and flows are described with differential equations, so it can also be simulated.

System dynamic has been widely used in a wide range of areas, for example population, economic and ecological systems. In the case of critical infrastructures it can be used to explore the dynamic behavior of the complex system and the cascade propagations of failures. It does not need a big amount of data and in [81] has been used to study network vulnerabilities to failures.

2.1.4 Hybrid System Modeling

Hybrid system is a dynamic system that exhibits both continuos and discrete behavior. So the system can be described by differential equations as well as difference equations or a discrete events machine. Hybrid systems have been intensively studied in the past few years both for their rigorous mathematical foundations and for engineering

designs [55]. That means a continuously changing variable describing the change of systems behavior over time can trigger a state machine transition, which means an occurrence of a discrete event [38]. On the other hand a state can change as a result of some discrete events which are directly linked to the continuous system behavior [29].

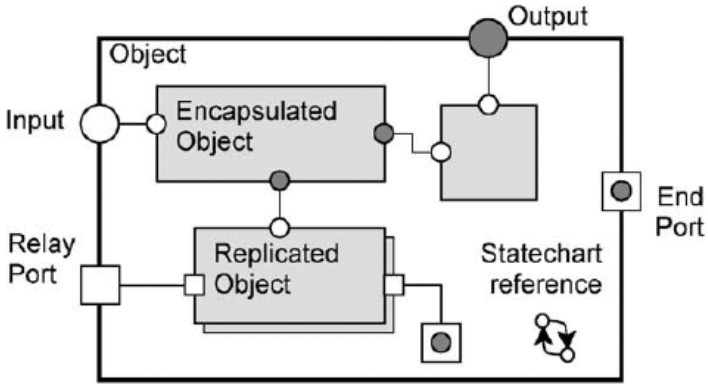


Figure 2.3: Example of Hybrid System Model [29]

Hybrid System Modeling applies mainly to the huge group of continuous systems controlled by discrete events, such as systems coordinating processes (e.g. air and ground transportation), infrastructures, robots, etc.

2.1.5 Hierarchical Holographic Modeling

The term hierarchical refers to an understanding of risks due to different levels in a hierarchy (i.e. risks at "system-of-systems" level, individual system level, sub-system level and component level). The term holographic modeling refers to a multi-view image of a system with regard to identifying vulnerabilities. Central to the mathematical and systems basis of hierarchical holographic modeling (HHM)

is the overlapping among various holographic models with respect to the objective functions, constraints, decision variables, and input-output relationships of the basic system. Through HHM [43] multiple models can be developed and coordinated to capture the essence of many dimensions, visions, and perspectives of infrastructure systems. One example is the study conducted for the President's Commission on Critical Infrastructure Protection on the U.S. water supply system. In applying the HHM philosophy the risk to a water supply infrastructure is decomposed into 16 major categories. The categories represent the risks to a water supply system from the multifaceted dimensions of each major category including the likelihoods, root causes, consequences, and direct and indirect impacts.

The HHM approach to reduce infrastructure vulnerability addresses its holistic nature in terms of its hierarchical institutional, organizational, managerial, and functional decision making structure in conjunction with factors that shape that hierarchical structure. These include the hydrologic, technologic, and legal aspects as well as time horizons, user demands on the infrastructure, and socioeconomic conditions. Unfortunately no real-time simulation is possible. HMM has been applied to study risks for agencies in the U.S. such as - besides the PCCIP - the FBI, the NASA, the Virginia Department of Transportation (VDOT), and the National Ground Intelligence Center [44].

2.1.6 Topological and complex network models

The topological analysis of properties of a network can reveal important and interesting information about the system structure [17], evolutionary dynamics, topological vulnerabilities[84], and the level of functionality demanded of its own components (for instance, topological centrality measures allow us to determine which network elements are likely undergo intense usage because of heir "location" in

the network).

Anyhow has been demonstrated that simple topological description cannot capture all of system's properties when there is some dynamic process acting on the network. Therefore at the topological network has been included basic features of the dynamical process. In [56] topological and dynamic models have been used to simulate the failure rates of electric power systems or in [81] to study the flow of failures in a complex network and the produced delay.

2.2 Simulation Models

Simulation, or more specifically, computer simulation is often used as an adjunct to, or substitution for, modeling systems for which simple closed form analytic solutions are not possible. There are many different types of computer simulation, but all of them have the common goal to generate a sample of representative scenarios for a model in which a complete enumeration of all possible states would be prohibitive or impossible.

Several software packages exist for running computer-based simulation modeling (e.g. Monte Carlo simulation, stochastic modeling, multi-method modeling) that makes the modeling almost effortless.

We present two modern technique of simulation that are generally used in the study of complex systems as well as system-of-systems: the Agent Based Modeling & Simulation and the federated approach.

2.2.1 Agent Based model

Agent-based models (ABM) consist of dynamically interacting, rule–based agents. An agent-based model can exhibit complex behavior patterns and provide valuable information about the dynamics of the real-world system simulated. An agent is a software object implemented on a computer network. Agents have access to certain

information and they are able to "communicate" with each other. Additionally, agents' design can include an ability to learn about the environment and formulate unique sets of decision rules[29]. Agent based models are often used to observe aggregate activity for a population of agents. ABM can also be seen as a modeling framework rather than a methodology, because it is based on further underlying techniques like Monte-Carlo, FTA, etc. The major advantage of the ABM approach for modeling and simulation critical infrastructure interdependencies is the possibility to emulate an emergent behavior. The overall system behavior results from the interactions of the multiple single agents and is not specified on the system level. Detailed data is only needed on the agents' level (bottom-up principle), not on the system level. Detailed data is only needed on the agents' level (bottom-up principle), not on the system level. More details about ABM will be given in the next chapter when we present our model partially based on such methodology. Anyhow many projects that use ABM can be found in literature.

2.2.2 Multi-domain or federated simulation model

Every specific sector simulator is appropriate to model a particular domain by means for example a specific mathematical model like Input-output model. Anyhow, to model critical infrastructure we need to be able to simulate different domains at the same time without loosing important information during the exchange of data among heterogenous simulators. The use of heterogenous simulators is pretty common in the military field to model, for example, air and terrain forces at the same time. Moreover, in such filed there is interest in real-time simulation to train soldiers as well as commanders in deploying troops. The current standard for distributed simulation is the High-Level Architecture v.1.3 and its successor IEEE.1516 which provides rules for federated simulation. The communication among

federates is managed by a kind of broker called Runtime Infrastructure (RTI). The RTI, aside managing data and communication, has to manage the synchronization since every federate can have a different representation of the real time. Today there are many tools that implement such an architecture and help to manage and build our own federation, that is where all federates reside. High Level Architecture has many advantages especially for the modeling and simulation of dynamic behavior of "system of systems". It is useful for quantitative simulation even though is very time and resource consuming.

2.2.3 Petri nets model

Stochastic Petri Nets (in short SPN) are a time enhanced variant of place-and-transition nets which are mathematical models of non-deterministic and discrete distributed systems. A Petri Net model is a bipartite directed graph. It consists of places and transitions. Places may contain any number of tokens. When a transition switches ("fires"), it consumes the tokens from its input places, performs some processing task, and places a specified number of tokens into each of its output places.

Generalized Stochastic Petri Nets (GSPNs) are an extension of SPNs which allow timeless as well as timed (exponential) transitions. Petri nets are a well known technique to implicitly define large automatons needed to model distributed systems. Petri nets have an advantage in that the size of the net, i.e. the number of places and transitions, grows but in linearity with the number of components.

SPN can be applied to model common mode failures and cascading effects in complex systems, e.g. [50], as well as to analyze the impact of communication on power grids [78]. GSPNs are also suitable for formalizing and simulating dynamic aspects describing the semantics and activities of e.g. workflow systems and distributed and

Table 2.1: Legend for Petri Net Model in[42]

Transtions	Places
1. Electric Power is Disrupted	1. Electric Power ON
2. Lubricants in Reserves are Consumed	2. Electric Power OFF
3. Power Disruption Affects Natural Gas Pro-duction	Natural Gas Production Stops
4. Natural Gas in Reserves is consumed	4. Consumed Natural Gas
5. Power Disruption affects Oil Lubricants Production	5. Oil Lubricant Productions Stop
.....	.....

concurrent computing systems.

2.3 Critical Infrastructures Projects

Currently in literature we can find many projects-tool to support the modeling of critical infrastructure. Most of them are supported by government agencies as well as private companies and are mostly implemented in universities, centre of research or as the collaboration of both. In particular in US there are three department involved in critical infrastructure:

- Department of Homeland Security (DHS): for the protection of citizens from terroristic attacks and natural disasters
- Department of Defense (DOD): for the coordination and supervising of all agencies of the government related directly to

military forces and the national security

- Department of Energy (DOE): responsible for the energy and the nuclear safety. It is also responsible for the nuclear weapons and the nuclear waste disposal.

Some centre of research like the Aragone National Laboratories (ANL), Idaho National Laboratories (INL) and the Los Alamos National Laboratories (LANL) are heavily involved in such projects. In Europe there is not yet a well centralized department for the protection of critical infrastructure and generally every nation has its own departments. Recently, the Information Society of the European Commission has provided several documents and is supporting the Joint Research Centre (JRC) both to develop models for supporting critical infrastructure protection and to manage multiple different centre of research and universities spread around Europe.

2.3.1 Projects survey

The modeling of critical infrastructure is a very challenging task since it is composed of multiple interconnected infrastructures that exchange service as well as products each other. Considering the survey presented in [67] we want to show the approaches adopted besides the status and goals of the project-tools. In [67] were considered six categories to characterize every tool which are:

Infrastructures: in the survey were considered 12 different sectors:

1. Agriculture and food
2. Banking and finance
3. Defense industrial base
4. Emergency services
5. Energy

6. Government
7. Industry/manufacturing
8. Information and telecommunication
9. Postal and shipping
10. Public health and safety
11. Transportation
12. Water

Modeling and simulation technique that we have seen above

Integrated vs. coupled models: use of a single framework that integrates everything in one simulator or the coupling of multiple sector simulators

Hardware/software requirements: the portability and exportability of programs and data.

Maturity level: the survey defines 4 maturity level:

1. Research: the model is only conceptual
2. Development: the model is in developing
3. Mature Analytic: the code is stable and has reached a good level, and some experiments are executed to conduct some analysis.
4. Mature Commercial: the tool is a commercially licensed product.

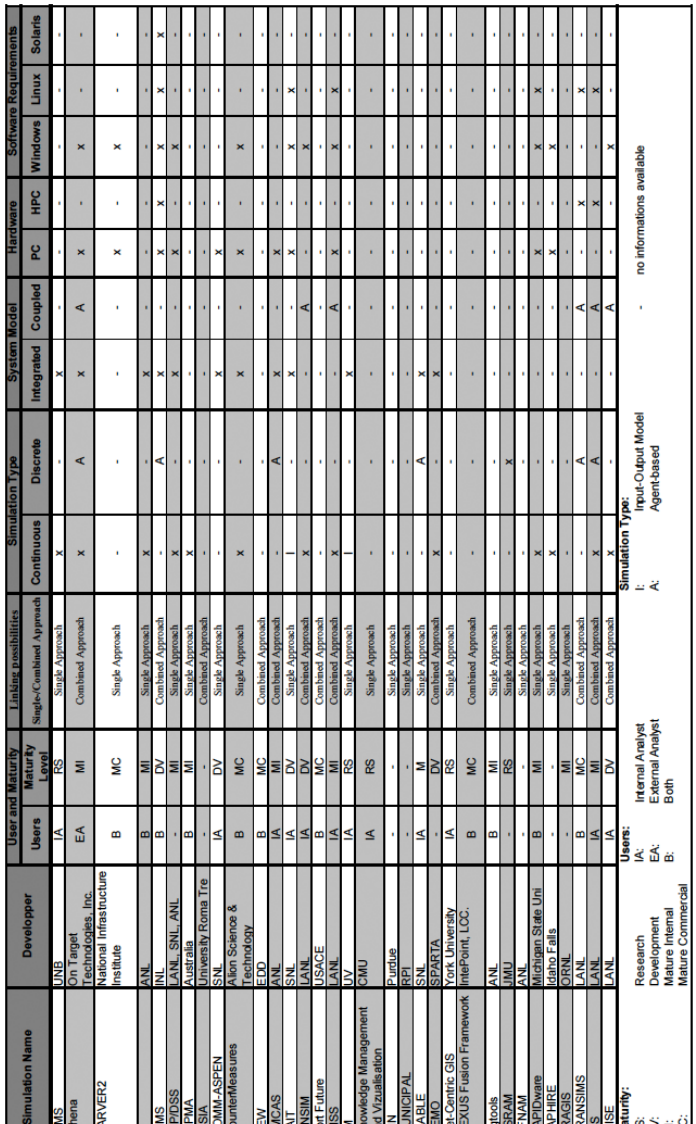
In particular in the survey were presented 33 tools that are reported in figure 2.4 and 2.5. Most efforts have been made in the sectors electricity, information and telecommunication technology, and transportation, but further infrastructure sectors are also considered

(12 sectors in total). Some of the approaches consider more than one infrastructure sector but this does not mean that they can provide combined modeling and simulation. A more detailed description of every project can be found in [67].

Using the same methodologies, we want to considers other three important projects that have reached a mature analytic level. Such projects are the IRRIS, DIESIS and I2Sim project. The first two are European projects while the last one has been implemented in the British Columbia University in Canada.

	Simulation Name	Developper	Area Model by Infrastructure Sector																# of CIS
			Electric Power	Natural Gas	Drinking Water	Sewage Water	Storm Water	Human Activity	Financial Networks	SCADA	Telecom	Computer Networks	Oil Pipeline	Rail System	Higway System	Waterway System	Police/ Regulation Constraints	
1	AIMS	UNB														X		1
2	Ahena	On Target Technologies, Inc.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	15
3	CARVER2	National Infrastructure																0
4	CI	ANL	X	X						X	X							4
5	CIMS	INL	X					X		X								3
6	CIP/DSS	LANL, SNL, ANL	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	15
7	CIPMA	Australia	X	X					X		X							4
8	CISIA	University Roma Tre	X							X								2
9	COMM-ASPEN	SNL							X		X							2
10	CounterMeasures	Alion Science & Technology												X	X			2
11	DEW	EDD	X							X								2
12	EMCAS	ANL	X						X							X		3
13	FAIT	SNL	X	X												X		3
14	FINSIM	LANL							X		X							2
15	Fort Future	USACE	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	15
16	IESS	LANL	X	X					X									2
17	IIM	UV	X						X		X							3
18	Knowledge Management and Vizualisation	CMU	X											X	X	X		4
19	MIN	Purdue						X							X			2
20	MUNICIPAL	RPI	X								X			X				3
21	N-ABLE	SNL	X						X					X				3
22	NEMO	SPARTA	X	X	X									X			X	5
23	Net-Centric GIS	York University												X	X		X	3
24	NEXUS Fusion	IntePoint LCC	X					X			X			X	X	X		5
25	Ngtools	ANL	X	X														2
26	NSRAM	JMU	X								X						X	3
27	PFNAM	ANL		X									X					2
28	RAPIDware	Michigan State Uni	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	15
29	SAPHIRE	Idaho Falls	X															1
30	TRAGIS	ORNL												X	X	X	X	4
31	TRANSIMS	LANL						X										1
32	UIS	LANL			X	X	X	X			X					X		7
33	WISE	LANL			X	X	X											3
		# of Activities	21	11	7	6	6	9	10	8	13	4	5	12	11	11	7	

Figure 2.4: Tools survey for modeling of critical infrastructure (developers and areas) [38]



	Simulation Name	Developper	User and Maturity		Linking possibilities	Simulation Type		System Model		Hardware		Software Requirements		
			Users	Maturity Level	Single-/Combined Approach	Continuous	Discrete	Integrated	Coupled	PC	HPC	Windows	Linux	Solaris
1	AIMS	UNB	IA	RS	Single Approach	x	-	x	-	-	-	-	-	-
2	Athena	On Target Technologies, Inc.	EA	MI	Combined Approach	x	A	x	A	x	-	x	-	-
3	CARVER2	National Infrastructure Institute	B	MC	Single Approach	-	-	-	-	x	-	x	-	-
4	CI	ANL	B	MI	Single Approach	x	-	x	-	-	-	-	-	-
5	CIMS	INL	B	DV	Combined Approach	-	A	x	-	x	x	x	x	x
6	CIP/DSS	LANL, SNL, ANL	-	MI	Single Approach	x	-	x	-	x	-	x	-	-
7	CIPMA	Australia	B	MI	Single Approach	x	-	-	-	-	-	-	-	-
8	CISIA	University Roma Tre	-	-	Combined Approach	-	-	-	-	-	-	-	-	-
9	COMM-ASPEN	SNL	IA	DV	Single Approach	-	-	x	-	x	-	-	-	-
10	CounterMeasures	Alion Science & Technology	B	MC	Single Approach	x	-	x	-	x	-	x	-	-
11	DEW	EDO	B	MC	Combined Approach	-	-	-	-	-	-	-	-	-
12	EMCAS	ANL	IA	MI	Combined Approach	-	A	x	-	x	-	-	-	-
13	FAIT	SNL	IA	DV	Single Approach	I	-	x	-	x	-	x	x	-
14	FINSIM	UANL	IA	DV	Combined Approach	x	-	-	A	-	-	x	-	-
15	Fort Future	USACE	B	MC	Combined Approach	-	-	-	-	-	-	-	-	-
16	IEISS	LANL	IA	MI	Combined Approach	x	-	-	A	x	-	x	x	-
17	IIM	UV	IA	RS	Single Approach	I	-	x	-	-	-	-	-	-
18	Knowledge Management and Vizualisation	CMU	IA	RS	Single Approach	-	-	-	-	-	-	-	-	-
19	MIN	Purdue	-	-	Single Approach	-	-	-	-	-	-	-	-	-
20	MUNICIPAL	RPI	-	-	Single Approach	-	-	-	-	-	-	-	-	-
21	N-ABLE	SNL	IA	M	Single Approach	-	A	x	-	-	-	-	-	-
22	NEMO	SPARTA	-	DV	Combined Approach	x	-	x	-	-	-	-	-	-
23	Net-Centric GIS	York University	IA	RS	Single Approach	-	-	-	-	-	-	-	-	-
24	NEXUS Fusion Framework	IntePoint, LCC.	B	MC	Combined Approach	-	-	-	-	-	-	-	-	-
25	Ngtools	ANL	B	MI	Single Approach	-	-	-	-	-	-	-	-	-
26	NSRAM	UMU	-	RS	Single Approach	-	x	-	-	-	-	-	-	-
27	PFNAM	ANL	-	-	Single Approach	-	-	-	-	-	-	-	-	-
28	RAPIDware	Michigan State Uni	B	MI	Single Approach	x	-	-	-	x	-	x	x	-
29	SAPHIRE	Idaho Falls	-	-	Single Approach	x	-	-	-	x	-	x	-	-
30	TRAGIS	ORNL	-	MI	Single Approach	-	-	-	-	-	-	-	-	-
31	TRANSIMS	LANL	B	MC	Combined Approach	-	A	-	A	-	x	-	x	-
32	UIS	LANL	IA	MI	Combined Approach	x	A	-	A	-	x	-	x	-
33	WISE	LANL	IA	DV	Combined Approach	x	A	-	A	-	-	x	-	-

Maturity:
RS: Research
DV: Development
MI: Mature Internal
MC: Mature Commercial

Users:
IA: Internal Analyst
EA: External Analyst
B: Both

Simulation Type:
I: Input-Output Model
A: Agent-based

- no informations available

Figure 2.5: Tools survey for modeling of critical infrastructure (characteristics) [38]

2.3.2 The Service-Focused approach IRRIS

The IRRISS project is from the collaboration of 15 european partners from area of research and private companies in which the Fraunhofer Institute was the project coordinator. It has the goal to determine interdependencies among infrastructures using an agent based simulator and to develop the Middleware Improved Technology (MIT) a collection of software components to facilitate IT-based communication among infrastructures as well as different infrastructure providers. The scenario simulated is based on a real scenario called "Rome Mini Telco Blackout" which refers to the flooding event of a Telecom Italia major telecommunication service node occurred in Rome on 2nd January 2004.

The simulator is named SimCIP and it is baed on the Implementation, Services and Effect (ISE) model (see figure 2.6). The ISE model split the full model in three separated levels:

1. implementation layer: encapsulates the domain specific data, logic and behavior model of the components

2. service layer: exchange of data between different model components (inter or intra-domain)

3. effect layer: effects of the data-processing

SimCIP is an agent-based model simulator that can simulate several variety of infrastructure in an integrated environment. The behavior of an infrastructure can be done in an external specific simulator of the particular domain. SimCIP has the task of defining the dependencies between the components, setting the initial values and collecting and evaluating the results of the simulation done by the external simulators. The agent encapsulates the state of network components and some of them are are transformed into variables that are abstract enough to be exchanged on the service layer of the ISE

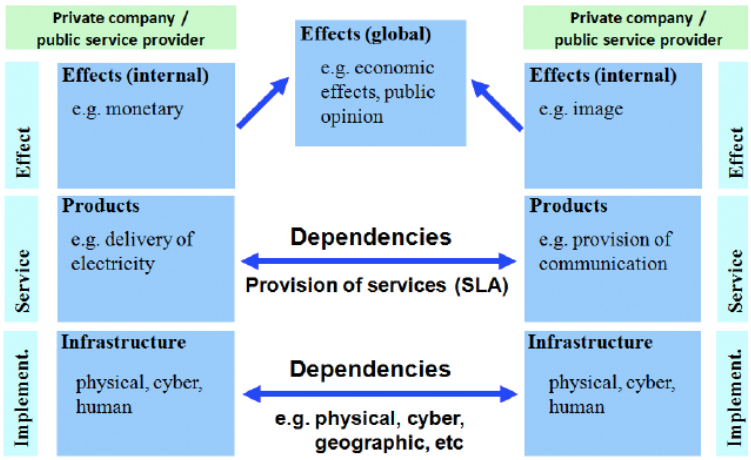


Figure 2.6: The Implementation-Service-Effect CI metamodel [18]

model. The current implementation of SimCIP simulates interdependencies between power grid and telecommunication network and uses as external simulator the PPS-Sincal and ns-2 respectively to the two infrastructures.

2.3.3 The Federative approach DIESIS

The DIESIS is a project supported by the European Community started on 1st February 2008 and of two-years duration. The project has the goal of performing a design study for an e-Infrastructure enabling federated simulations of CI systems and supporting research on CIP. In particular the European e-Infrastructure [6] has the goal to allow researchers to access to unique and distributed facilities (including data, instruments, computing and communication resources). This European e-Infrastructure will support full cooperation of the different partners in charge for studying (inter)dependencies of critical infrastructures, while preserving the confidentiality of the proprietary knowledge embedded into the different models and simulation

packages.

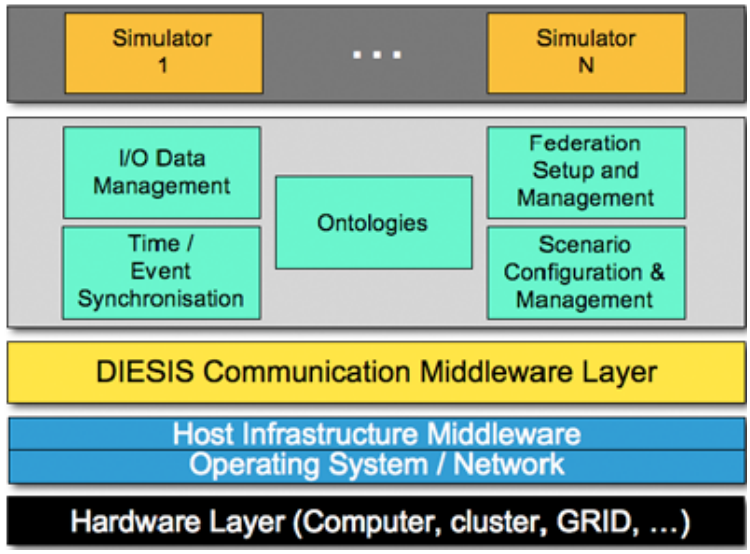


Figure 2.7: DIESIS architecture [77]

As illustrated in figure 2.7 a middleware is developed for the interoperability among different federates similar to the HLA architecture. The middleware will support also the scenario configuration and management as well as the setup of the simulation by means of a workflow. Moreover an ontology has been included in DIESIS [58] which will allow the representation of the knowledge about the heterogenous infrastructures domains, the description of infrastructures networks and the interconnections among these networks. Moreover, the ontological framework allows the definition, through logic rules, of the interconnections semantic. The DIESIS-ontology has been applied to describe three different infrastructures: railway, electric and telecommunication. To prove the ontology, the Rome flooding scenario from the previous IRRISS project have been used.

2.3.4 The Supply and Demand System approach I2Sim

The I2Sim simulation environment was developed at the University of British Columbia the JIIRP project. The system-modeling is based on the modeling of flow resources between component infrastructures without revealing their internal details. Every component uses an internal model to evaluate its status, while the I2Sim combines these operating states into a system-of-systems solution. A main objective of the simulator is to capture the time line events during a large emergent scenario so that is possible to predict the evolution of the global system when a certain decision is taken. Aside the real time decision support, I2Sim has the goal to permit the analysis and discovery of vulnerable points in the system as well as gaps in policies and procedures. Similarly to Petri's nets, I2Sim defines the following entities (see also figure 2.8):

- Cells (Production Units): it is an entity that produces something and requires some inputs (e.g. primary cabin, routers, hospitals)

- Channels (Transportation Units): entities products are transferred to others by some transportation channels (e.g. wires, pipes, roads)

- Tokens (Exchange Units): represents the quantities needed in input or produced in output (e.g. a doctor, water, a phone call)

- Controls (Distributor & Aggregator Units): interface the physical layer with the decisions making layer.

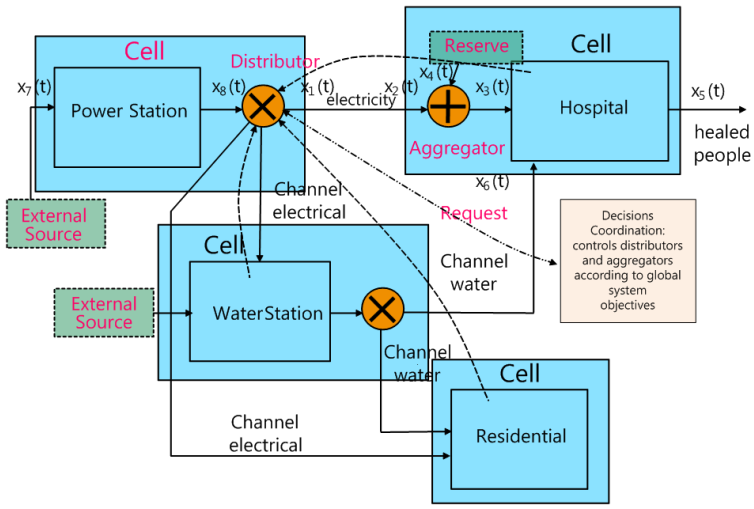
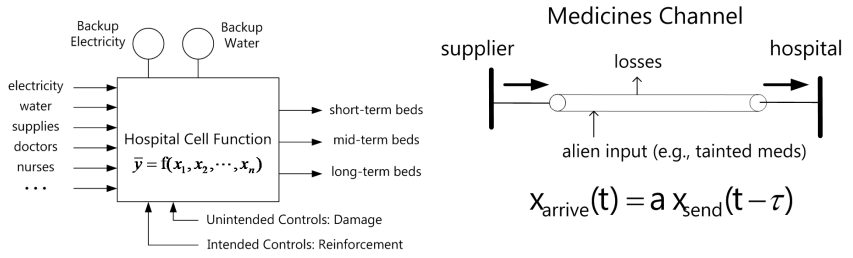


Figure 2.8: I2Sim ontology: cells, channels, tokens and controls [57]

2.3.4.1 Cell and channel model

A cell is a functional unit (see figure 2.9(a)) that needs a certain quantity of a product as input to produce its own relative output. Generally the relation between inputs and outputs is a function multidimensional and not linear. While the internal details are hidden to the simulator, I2Sim needs a Human Readable Table (HRT) which has to be provided by the owner of the infrastructure. HRTs can be represented as multidimensional hyper-surface which can be linearized to represent current state of a cell along time line evolving. Cells are connected each other with specific channels (figure 2.9(b)). Channels represent the means by which tokens are transported from a source cell to destination ones. Channels can have a coefficient of loss to simulate for example leakages or delays.

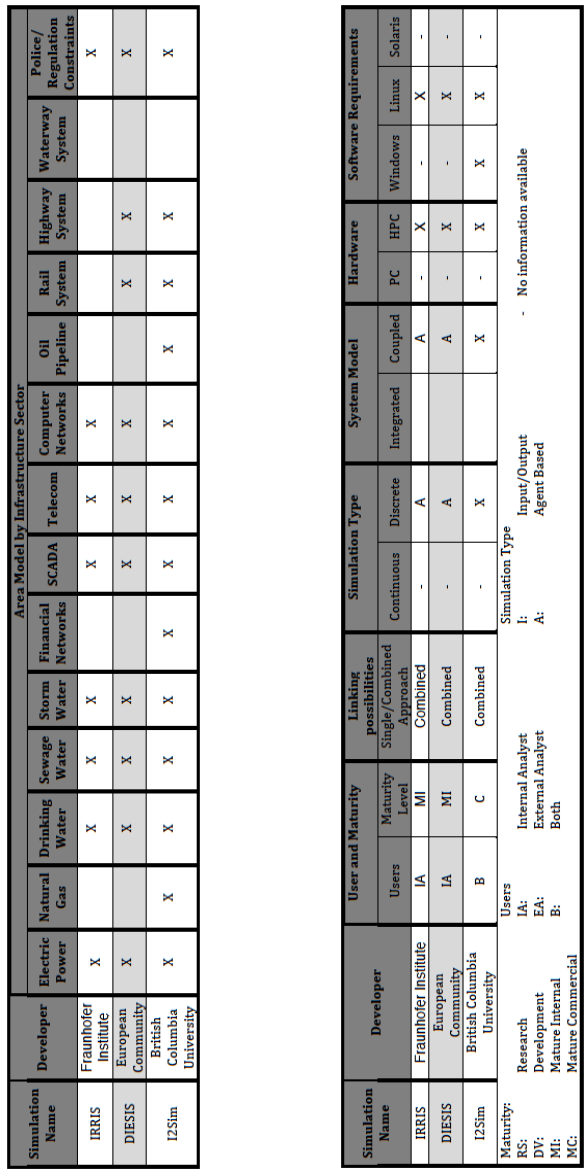


(a) Cell (b) Channel

Figure 2.9: I2Sim: cell and channel model [57]

2.3.5 Evaluation of IRRIS, DIESIS and I2Sim projects

Using the same methodology in [29] and tables presented previously 2.42.5 we evaluate previous presented projects. The IRRIS had the goal of collecting some new ideas from different partners and experimenting some new modeling technique. From such experience, the DIESIS project has been conducted so that it has avoided some of issues that IRRIS project has affronted. So DIESIS can be seen as an evolution of the IRRIS project and, at the same time, as the base for future European projects. Vice versa, I2Sim had the goal to provide a tool for real users, so that it results to be more usable compared to the IRRIS project. Moreover it embraces much more domains and can be more easily extended to other infrastructures.



Simulation Name	Developer	Area Model by Infrastructure Sector												
		Electric Power	Natural Gas	Drinking Water	Sewage Water	Storm Water	Financial Networks	SCADA	Telecom	Computer Networks	Oil Pipeline	Rail System	Highway System	Waterway System	Police / Regulation Constraints
IRRIS	Fraunhofer Institute	X		X	X	X		X	X	X					X
DIESIS	European Community	X		X	X	X		X	X	X		X	X		X
I2Sim	British Columbia University	X	X	X	X	X	X	X	X	X	X	X	X		X

(a) Developers and areas

Simulation Name	Developer	User and Maturity		Linking possibilities	Simulation Type		System Model		Hardware		Software Requirements		
		Users	Maturity Level	Single/Combined Approach	Continuous	Discrete	Integrated	Coupled	PC	HPC	Windows	Linux	Solaris
IRRIS	Fraunhofer Institute	IA	MI	Combined	-	A		A	-	X	-	X	-
DIESIS	European Community	IA	MI	Combined	-	A		A	-	X	-	X	-
I2Sim	British Columbia University	B	C	Combined	-	X		X	-	X	X	X	-

Maturity:
RS: Research
DV: Development
MI: Mature Internal
MC: Mature Commercial

Users
IA: Internal Analyst
EA: External Analyst
B: Both

Simulation Type
I: Input/Output
A: Agent Based

- No information available

(b) Chracteristics

Figure 2.10: Tools survey for modeling of critical infrastructure for IRRIS, DIESIS and I2Sim

Chapter 3

Federated Agent Based Model

Contents

The modeling of critical infrastructure falls within the study and modeling of complex systems. The theory of complex systems is relatively recent and derives from Cybernetics and System Research

which started in the 1940's with people like Norbert Wiener, Warren McCulloch, Margaret Mead, Ross Ashby, Jhon Von Neuman, Heinz von Foerster, and others [46]. A system is generally defined as a collection of elements which interacts each other and make up a whole [24]. To describe the dynamic evolution of a system, scientists use simplified mathematical models which represent a whole system [76]. While a system can be considered complicated, it could not be complex. Actually there is not a shared definition of complex in literature. In [85] it is defined as:

> A system comprised of a (usually large) number of (usually strongly) interacting entities, processes, or agents, the understanding of which requires the development, or the use of, new scientific tools, nonlinear models, out-of equilibrium descriptions and computer simulations.

Herbert Simon [80] defines a complex system as

> A system that can be analyzed into many components having relatively many relations among them, so that the behavior of each component depends on the behavior of others.

While, according to Jerome Singer, [82] it is

> A system that involves numerous interacting agents whose aggregate behaviors are to be understood. Such aggregate activity is nonlinear, hence it cannot simply be derived from summation of individual components behavior.

Anyhow we can generally define a complex adaptive system (CAS) as any system in which there a large numbers of interacting entities, we cannot model the whole system linearly as the simple sum of agents' activities and, lastly, entities can use some sort of hierarchical-organization to reach a shared goal or more simply to survive.

One of the most used technique to model CAS is surely the Agent Based Modeling and Simulation (ABM&S). It is based on proven, highly successful techniques such as discrete-event simulation and object-oriented programming to discover strategic, tactical and operational business solutions [64]. The ABM&S allows to model CAS using a bottom-up approach, in which are modeled the behavior and interactions between single entities and at the same time it provides a view model of the whole.

3.1 Agent-based modeling of interdependent complex systems

As shown in many research works, agents can be used to model interdependent complex systems (e.g. [64]). A general definition of agent is the following [27]:

Definition 1 *An agent is an entity with a location, capabilities and memory. The entity location define where it is in a physical space ... What the entity can perform is defined by its capabilities ... the experience history (for example, overuse or aging) and data defining the entity state represent agent's memory.*

A critical infrastructure is characterized by its location, its behavior, interaction capabilities and its internal state. Then a critical infrastructure can be modeled as an autonomous agent and the system composed of interdependent critical infrastructures can be modeled as interacting agents which cooperate and/or compete to realize a common or an individual goals. In this chapter we present our model based on ABM&S and distributed simulation technique. Successively we adopt such model to simulate interactions between the electrical system and the communication network.

3.1.1 The Federated agent-based model

An agent a is described by the tuple (V_a, S_a, X_a) where:

1. $V_a = \{v_1^a, ..., v_{N_v^a}^a\}$, $v_i^a \in \mathcal{V}_i^a$ and $|V_a| = N_v^a$. V_a is the set of the agent attributes and $\mathcal{V}_i^a$ is the domain of the agent attribute i. The values assumed by the agent attributes at time t represent the state of the agent.

2. $S_a = \{s_1^a, ..., s_{N_s^a}^a\}$, $|S_a| = N_s^a$, is the set of services that the agent a provides to other agents. In our model agents interact exchanging services.

3. $X_a = \{x_1^a, ..., x_{N_x^a}^a\}$ ($|X_a| = N_x^a$) is the set of inputs of the the agent a. Inputs can be services produced by other agents or perturbations. A perturbation is an unpredictable event that modifies the agent state and alters the behavior of the agent a, reducing the a's capabilities to provide services. An input is characterized by the tuple $x_i^a = (t_x, x)$ where $x \in \mathcal{X}_i^a$ is the value of the input and t_x the time at which the value x is available ($t_x \in, \mathbb{R}^+$ or $t_x \in, \mathbb{N}^+$ if we consider continuous or discrete time respectively).

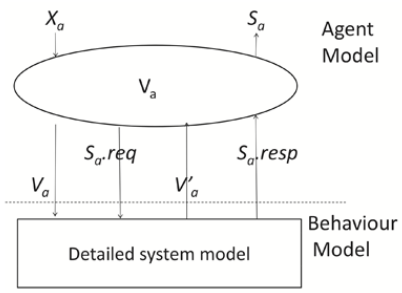
Comparing the proposed federated agent-based model with the Definition 1 we have that:

1. the agent state, memory and location are modeled by the agent attributes V_a;

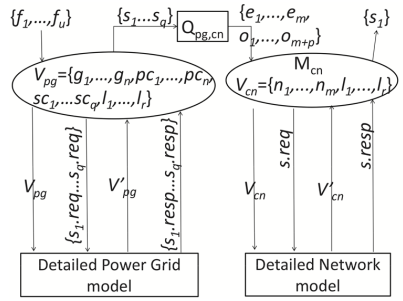
2. S_a and X_a model the capability of the agent to interact with other agents providing services and consuming data or services;

3. the agent behavior, that determines how inputs are processed, how services are provided and how the agent state evolves, is modeled using a sector specific model of the complex system modeled.

Figure 3.1(a) shows the proposed federated agent model. It is worth to note that only the agent a can interact directly with the detailed model of the complex system abstracted by a.



(a) Single agent model view



(b) The federated agent model of a complex interdependent system composed of the power grid (left) and of the communication network (right)

Figure 3.1: The federated agent-based model

Let us now define the relationship among agent attributes, services and inputs.

The agent state V_a is function of the time and of the agent inputs X_a, and implicitly of the agent behavior (as it will be explained in the following). Assuming that the time is discrete, $t \in \mathbb{N}^+$, and that each agent attribute v_i^a depends on a subset of the agent inputs $\{x_{j_1^i}^a, ..., x_{j_n^i}^a\}$ we have:

$$v_i^a = f_i^a(t, x_{j_1}^a, ..., x_{j_n}^a),$$
$$f_i^a : \mathbb{N}^+ \times \mathcal{X}_{j_1}^a \times ... \times \mathcal{X}_{j_n}^a \rightarrow \mathcal{V}_i^a.$$

The dependency of the agent attributes on the agent inputs is defined by the mapping[1]

$$M_a = \{m_{i,j}\}^{N_x^a \times N_v^a},\ m_{i,j} = \begin{cases} 1 & \text{if } x_i^a \in dom(f_j^a) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

It is important to remark that f_i^a and M_a depend on the specific system modeled and on the specific goal of the modeling and simulation study, then it is impossible to provide a generic expression for them. In section 7.3 we give an example of f_i^a and M_a.

The service s_i^a is function of the time, of the agent state, of the agent inputs and of a set of service input parameters $p_1^{i,a}, ..., p_{N_p^{i,a}}^{i,a}$, $p_j^{i,a} \in \mathcal{P}_j^{i,a}$:

$$s_i^a = g_i^a(t, v_{j_1}^a, ..., v_{j_n}^a, p_1^{i,a}, ..., p_{N_p^{i,a}}^{i,a})$$
$$g_i^a : \mathbb{N}^+ \times \mathcal{V} \times \mathcal{P} \rightarrow \mathbb{N}^+ \times [0, 1],$$

where $\mathcal{V} = \mathcal{V}_{j_1}^a \times ... \times \mathcal{V}_{j_n}^a$ and $\mathcal{P} = \mathcal{P}_1^{i,a} \times ... \times \mathcal{P}_{N_p^{i,a}}^{i,a}$.

In our model we assume that $s_i^a = (t, 1)$ if the $i - th$ service, invoked at time t', is delivered at time $t \geq t'$. On the contrary, $s_i^a = (t, 0)$ if the service cannot be delivered. In this later case, the time t is meaningless or, depending on the specific service, it can be interpreted as the service time out.

The proposed on-off model for service delivery can be extended considering that s_i^a can be provided at different QoS levels s, $0 \leq s \leq 1$. The QoS level s=0 means that the service is not delivered and the QoS level s=1 means that the service is delivered at the 100% of the QoS level.

The last step toward the definition of a federated agent-based model is to provide a solution for:

[1] $dom(f)$ is the domain of the function f and $cod(f)$ the co-domain of the function f

1. a model of the agent state evolution,

2. a model of service delivery

3. a model of service delivery time.

We address issues 1-3 using a detailed model of the target complex system. The innovative idea we introduce is to consider the detailed system model as a black-box controlled by the agent model and that computes the new system state, the services delivery time and the service level.

The interaction between the agent model and the detailed system model (see Figure 3.1(a)) is determined as follow. The agent model requests, at the detailed system model, to compute the new system state V'_a on the basis of the current agent state V_a and of the services requested $S_a.req$. The service response and the service delivery time are computed by the detailed system model and returned in $S_a.resp$. In the proposed solution the agent model plays the role of the orchestrator of the simulation, while the detailed system model plays the role of a simulation component that receives, from the orchestrator, the system workload (V_a, $S_a.req$).

3.1.2 Interconnecting Agents: The Interdependencies Model

Interdependencies can be classified as [75]: physical, geographical, cyber, and logical. In our model, physical and cyber interdependencies are modeled as service exchange. Moreover, the concept of perturbation allows to model geographical and logical interdependencies. We concentrate our attention on cyber and physical interdependencies, even though we remarks that a proper use of the concept of perturbation allows also to model geographical and logical interdependencies.

Two agents a and b interact if exist at least a service provided by a that is an input for b: $s_i^a(t) = x_j^b(t)$ for some $1 \leq i \leq N_s^a$ and $1 \leq j \leq N_x^b$. In this case the agent b depends on the behavior and on the services provided by a, then a and b are interdependent. If a depends on b and b on a we have cyclic interdependencies and if a and b does not interact directly but interact through a chain of agents interaction we can say that a and b are indirectly interdependent. Then the interdependencies between agents a and b are modeled by the mappings:

$$Q_{a,b} = \{q_{i,j}\}^{N_s^a \times N_x^b},\ q_{i,j} = \begin{cases} 1 \text{ if } s_i^a = x_j^b \\ 0 \text{ otherwise} \end{cases}$$

and M_b (defined in equation 3.1). The mapping $Q_{a,b}$ defines how a and b interact, while the mapping M_b defines how the b's state is influenced by the b's inputs. In the same way, cyclic interdependencies can be described by four mappings $Q_{a,b}$, M_b, $Q_{b,a}$, M_a.

3.1.3 The Federated ABMS methodology

The steps toward the definition of an federated agent-based model are the following:

1. Identification of the simulation study goals.
2. Identification of the complex systems (e.g. infrastructures) that compose the compound complex system under study.
3. For each component system identified in step 2, identify:
   (a) the set of variables that are representative of the system state;
   (b) the set of services that allow to represent the interaction of the complex system with the other component systems, with the environment and with human beings;

(c) the set of perturbations and inputs that influence the component system behavior;

(d) the relationship among agent inputs and agent state variable.

Steps (a)-(c) should be supported by series of interviews of infrastructures experts.

4. Associate an agent a to each system identified in step 3 and define the related agent model (V_a, X_a, S_a) and M_a. V_a, X_a S_a and M_a are determined in steps 3.(a)-3.(d) respectively.

5. For each agent defined in the previous step identify the sector-specific simulation model useful to simulate the infrastructure behavior.

6. Identify the system interdependencies, for example using interviews of infrastructure experts.

7. For each couple of infrastructures a and b ($a \neq b$) define the interdependencies matrix $Q_{a,b}$.

3.2 The case study

In the following we apply the federated agent-based methodology to a target complex system composed of an IP communication network (cn) and of a power grid (pg). We suppose that the communication network depends on the power grid, and that there are no auxiliary power mechanisms. For lack of space we concentrate our attention on the above described steps 4 and 7.

The network state V_{cn} is represented by $\{n_1, ..., n_m, l_1, ..., l_r\}$ where n_i is a network node (router, access point, switch,...) and l_j is a network link connecting two network nodes; m is the number of nodes

and r the number of links. We assume that $n_i = 1$ ($l_i = 1$) if the node (link) i works and $n_i = 0$ ($l_i = 0$) if the node (link) i does not work.

The agent inputs are $X_{cn} = \{e_1, ..., e_m, o_1, ...o_{m+p}\}$ where e_i models the power supply (electricity) for the network node n_i and o_i models an unpredictable system outage for the network node n_i (link l_i). $e_i = 0$ means that the node n_i cannot be supplied by the power grid. $o_i = 1$ means that n_i (l_i) has experimented an outage and it can not work. The mapping M_{cn} that models dependencies of the state variables on the agents inputs is the following

	$n_1, ..., n_m,$	$l_1, ..., l_p$
e_1		
$\vdots$	I_m	0_p
e_m		
o_1		
$\vdots$	I_{m+p}	
o_{m+p}		

where I_m is an $m \times m$ identity matrix and 0_p is a $p \times p$ null matrix.

In our simplified model the relationship f_{cn} among the agent inputs and agent state is modeled by the following function:

$$f_{cn} = \begin{cases} n_i = 0 \text{ if } (e_i = 0) \text{ or } ((e_i = 1) \text{ and } (o_i = 1)),\ \forall t \\ l_i = 0 \text{ if } o_i = 1,\ \forall t \\ n_i = 1, l_i = 1 \text{ otherwise, } \forall t \end{cases}$$

The service provided by the communication network is "send a message from n_i to n_j" where n_i and n_j are two network access point. Then our simplified network model provides only one service s with two input parameters p_1 and p_2, where p_1 is the source node and p_2 is the destination node. $s = (t_R, 1)$ if the message is delivered at time

t_R (the service response time) and $s = (\cdot, 0)$ if the service can not be delivered because the internal state of the communication network, given by value of $\{n_1, ..., n_m, l_1, ..., l_p\}$.

To determine the internal state evolution of the communication network on the basis of the agent inputs and service requests we use an event-driven network simulation model implemented using OMNeT++[8].

Figure 3.1 (right) shows the connection between the agent model and the detailed network simulation model.

The power grid model considers the following components: power generators (or generation plants) pg, primary cabins pc, secondary cabins sc and distributions/transmission lines d. Then the power grid state is modeled by the set of attributes $V_{pg} = \{pg_1, ...pg_n, pc_1, ..., pc_r, sc_1, ..., sc_q, d_1, ..., d_z\}$ where: $pg_k = 1$ if the generator k work properly and $g_k = 0$ otherwise; $pc_k = 1$ if the primary cabin k work properly and $pc_k = 0$ otherwise; $sc_k = 1$ if the secondary cabin k work properly and $sc_k = 0$ otherwise; and $d_k = 1$ if the distribution or transmission line k work properly and $d_k = 0$ otherwise.

There are many external factors that can influence the power grid behavior, however, for simplicity, we consider only faults $\{y_1, ...y_u\}$, $u = n + r + q + z$. If $y_k = (t, 1)$ the power grid component k will experiment a fault at time t. Otherwise, if $y_k = (t, 0)$, the component k does not experiment any outage or it is repaired at time t after a fault at time $t' < t$.

Then we can define $M_{pg} = \{m_{i,j}\}^{u \times u} = I_{u \times u}$ and

$$f_{pg} = \begin{cases} v_i = 1 \text{ if } y_i = 1, \ \forall t \\ v_i = 0 \text{ otherwise}, \ \forall t \end{cases}$$

where v_i is a power grid component pg_i, pc_i, sc_i, d_i.

The service provided by the power grid is " to provide the electricity to the secondary cabin l_k". We assume that the load is attached

directly to the secondary cabins through a bus. Then we have q services: $s_j = (t, 1)$ if the secondary cabin sc_j is operative at time t and $s_j = (\cdot, 0)$ otherwise. The value of s_j depends on the state of all the power grid components (generators, primary cabins and links). The power grid behavior is modeled using a load flow model (that is a time independent model). At time t the power grid simulation model receives as input V_{pg} and it recomputes the power flow, producing the new values for the model state V'_{pg}.

The interdependencies between the power grid and the communication network, identified in step 6, are defined by the mappings

$$Q_{pg,cn} = \{q_{i,j}\}^{N_s^{pg} \times N_x^{cn}}, \; q_{i,j} = \begin{cases} 1 \text{ if } s_i = e_j \\ 0 \text{ otherwise} \end{cases}$$

and by M_{cn} previously defined.

For simplicity we do not model the power grid control functionalities, that is the dependency of the power grid on the communication network.

3.3 Implementation Issues

The implementation of a federated agent-based simulation model is a challenge and there are many issues that have to be addressed. To mention few: model validation, experiment reproducibility, extendibility to diverse and unforeseen scenarios, simulation scalability, implementation of agents and simulation models federations. In the following we discuss in detail the last two issues.

3.3.1 Implementation of agents

In literature the problem of discrete agent simulation is widely addressed. There are different frameworks that support agent and

multi-agents simulation. Examples are RePast[11], JadeSIM [40], SIM_AGENT [83].

All these approaches have their advantages and disadvantages. Distributed agents (e.g. JadeSIM) allow to design scalable simulation model, some of them are compliant with distributed simulation standard, but they introduce difficulties in designing and testing the simulation logic. Most of the frameworks, such as Repast, do not use distributed agents, thus facilitating the design and testing of the simulation logic, but limiting the simulation scalability.

However, Federated ABMS is independent from the technology used to implement agents. In our prototype we have decided to use RePast as discrete agent simulation framework.

3.3.2 Federation of the agent-based model(s) and sector specific models

The implementation of the proposed federated agent-based simulation model requires the use of distributed simulation technologies. Distributed simulation allows to integrate together heterogeneous simulation model that can be world wide distributed or locally distributed. Moreover, distributed simulation enables the execution of huge simulations.

If a distributed agents technology is used (see figure 3.2) we have, for each infrastructure, a federation composed of the agents model and of the sector specific simulation model (or more then one if needed). The federated agent-based simulation model is obtained federating together all the federations in a unique federation, the *Critical Interdependent Infrastructures Federation*.

If a centralized agent-based simulation framework is used: the framework interacts with all the sector specific simulation models (see figure 3.3), while the agents interacts among them using methods invocations. The agent-based simulation framework has a unique fed-

erate ambassador, that manages the interaction between each agent and the related sector specific simulation models.

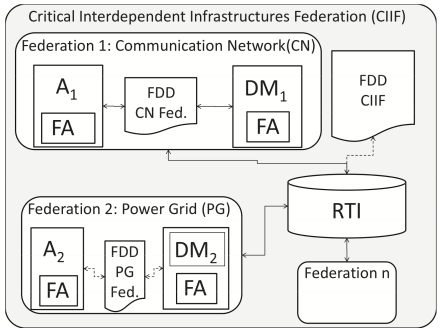


Figure 3.2: The distributed agents implementation. FA is the federate ambassador, A the agent model, DM the sector specific simulation model and FDD the FOM Data Document

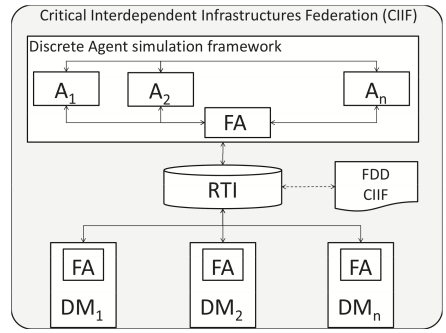


Figure 3.3: The centralized agents implementation. FA is the Federate Ambassador, A the agent model, DM the on model and FDD the FOM Data Document

3.3.3 Interaction between the agent model and the sector specific model

The design of the interaction between an agent based model and the related sector specific model is one of the main challenging problems.

Two aspects have to be considered: the implementation of the physical interaction between models; and the implementation of the logical relationship between the agent state and inputs (V_a and X_a) as well as the state variable and parameters of the detailed simulation models.

Using the DIS terminology, the physical interaction is defined by the Federate Object Model (FOM). The agent model publishes, as

objects, the inputs X_a and the state variables V_a, while the sector specific model publishes V'_a as an object and S_a as an interaction.

The logical relationship is implemented on the agent side. The agent implements the function f_a and the mapping M_a. Each time an agent state variable changes value, the agent model changes the value of the related sector specific model variable. For example, if the network node n_i is a router and $n_i = 0$ at time t, the agent modifies, at time t, the router object published by the OMNeT++ federate.

3.3.4 Orchestration of the Federated agent-based simulation model

As a distributed application needs an orchestrator process that manages the application logic, the distributed simulation needs a process that manages the simulation logic. We named such process the simulation orchestrator.

In federated agent-based modeling and simulation, the agents model plays the natural role of the simulation orchestrator. If a centralized agent based simulation framework is used, the simulation orchestrator can be easily implemented. For example, in RePast, where each agent is implemented by a Java class, the simulation orchestrator is implemented by the *model class* that coordinates the setup and running of the agent model.

On the contrary, if distributed agents are used, a specific agents that works as simulation orchestrator have to be designed.

3.4 Concluding Remarks

In this chapter we argued for an alternative agent-based modeling and simulation approach to study interdependent complex systems. The proposed methodology, that capitalizes the advantages of ABMS

and of distributed simulation, is intended as an aid to whom has the challenging task to design a simulation framework for interdependent complex systems analysis.

With Federated ABMS a modeler can define an abstract model of the target compound complex system ignoring the details of the component systems models (that are used as black-box). This abstraction allows the modeler to concentrate her/his effort in modeling the whole complex system and the system interdependencies. Moreover, the use of distributed simulation allows to build scalable simulation models.

Further, the proposed solution can be easily extended with the models of geographical and political/social interdependencies. In such case, it is enough to add a matrix of correlation between a geography position and a perturbation as well as a geographic position and an agent attribute and/or service. If there is a perturbation inside a geographical area, only agent attributes and services inside such area will be influenced. Similarly, political and social models can be mapped to a specific geographical area which will be activated only if a perturbation inside the correlated area is perturbed.

Chapter 4

Design and implementation of Fed-ABM&S to study IT infrastructure

Contents

In the previous chapter we have seen that our model is basically based on Agent-based Modeling and Simulation which is a promising modeling and simulation technique. It is based on multiple agents that are autonomous, problem-solving computational entities which are capable of effective operations in dynamic and open environments. Agents are often deployed in an environment in which they interact and cooperate with other agents that might have possible conflicting aims.

We have also seen that federated simulation is a simulation technique which: i) allows to reuse existing simulation model, thus reducing the cost to develop a complex system model; ii) allows to distribute the execution of the simulation model over a set of nodes (locally or geographically distributed), in order to increase computational power, resource availability and fault tolerance.

The concept of federated simulation is not new and widely used and investigated in military and defense sector [59, 47, 53]. Consider that the High Level Architecture, actually IEEE std.1516, is the result of a research sponsored by the US Defense Modeling and Simulation Office. Also researchers apply distributed simulation, for example in computer networks, distributed systems design and performance evaluation[73, 28]. The sector, in which federated simulation is less used, is the industry: Boer et al. realized an extensive survey on such topic [25, 26].

Federated ABM&S exploits the ABM&S capabilities, to model the whole complex system as a set of interacting agents, and the Federated simulation capabilities, to reuse existing models which will help in modeling agents behavior at different levels of abstraction. The detailed agent behavior can be embedded into a unique simulation model (see figure 4.1 (a)). The detailed model of each subsystem, represented by an agent, can be provided by existing models and simulation softwares (see figure 4.1 (b)). For example, in the figure, the detailed behavior for the agent $A1$ is implemented by a simulation model running at site$A$. Using the federated simulation terminology, the ABM&S model and the specific simulation model are the federates, which altogether compose a federation.

The chapter is organized as in the following. In section 4.1 we present a case study that facilitates us the presentation of the proposed approach. In section 4.2 we introduce the general methodology and architecture of Federated ABM&S. Here we explain how we have integrated an open source IT discrete event simulator (OMNeT++

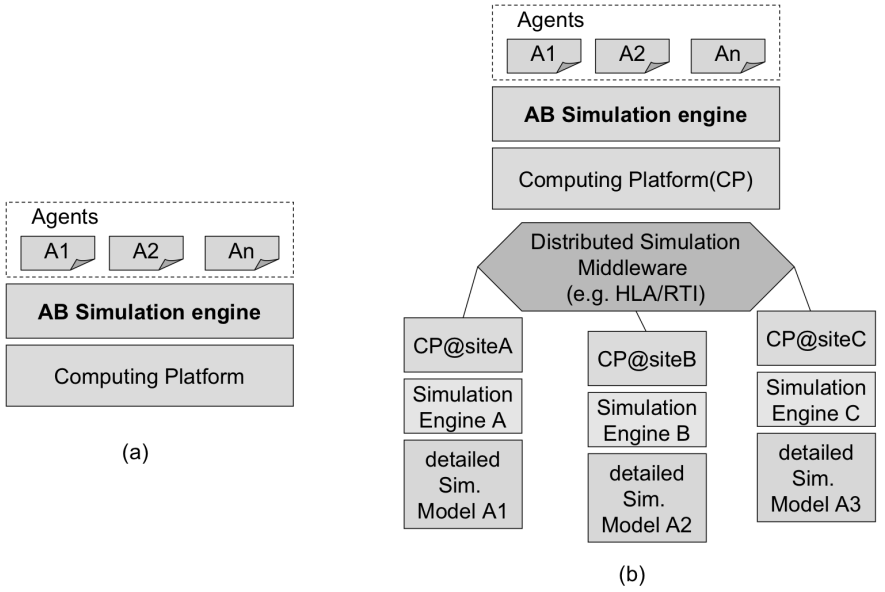


Figure 4.1: Agent-based Modeling & Simulation (a) v.s. Federated ABM&S (b)

[8]) with an open source agent-based simulation engine (Repast [11]). A selection of the simulation results, showing the effectiveness of the proposed approach, is presented in section 4.4. Concluding remarks at the end of the chapter.

4.1 The case study

The proposed FederatedABM&S approach is general enough to be applied to any critical infrastructure scenario. However, for clarity and easy of presentation, we describe how to apply our methodology

through an example-driven approach, which considers as a critical infrastructure application the Information System for Civic Emergency Management (IS4CEM), which, in case of some natural disaster or special events, provides information about the health care centers availability, the transportation network availability, and the event evolution.

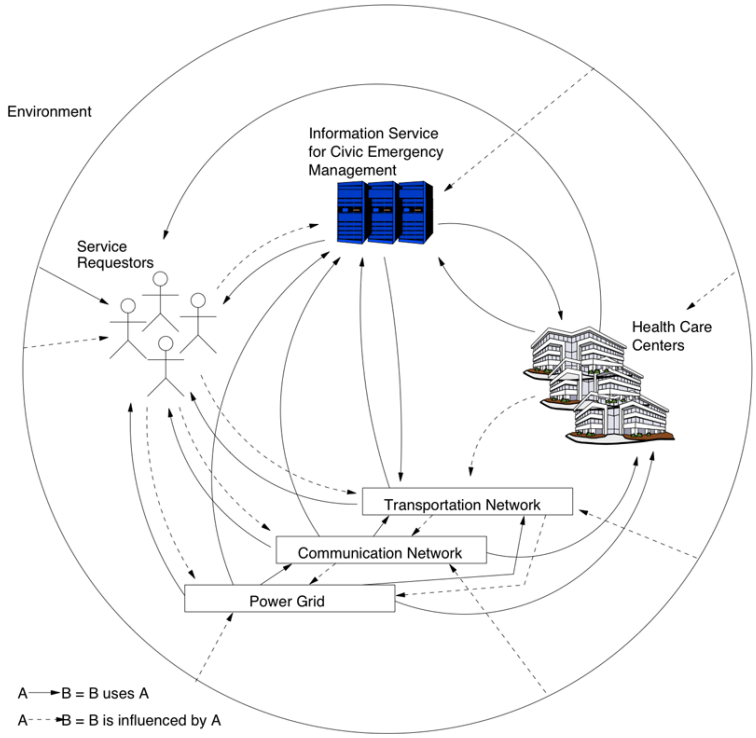


Figure 4.2: The Civic Emergency Management scenario.

Let us consider the scenario illustrated in Figure4.2 and describe its interdependencies. There is a generic service requestor (SR) that accesses the IS4CEM, which is hosted and published by a service provider through a wired network. The SR can be a citizen asking for help (in the following, wounded) or a succorer and she/he uses a set

of critical infrastructures to access the IS4CEM. Specifically, the SR uses a wired or a wireless connection to the communication network; she/he can use the power grid to power its connecting device; finally, she/he may use the transportation system to provide first aid to other service requestors or to reach a Health Care Center (HCC) to obtain assistance.

The IS4CEM is an information system that runs on a service provider platform and provides various information to the citizens. It also operates as a broker for first aid requests by selecting the most appropriate HCC to be reached (e.g., by taking into account the transportation system and HCCs status). The IS4CEM is accessed through the communication network and uses the information system of the transportation network to get information about its availability. The IS4CEM does not directly use the transportation network; however, since the transportation network may be used to provide assistance to the IS4CEM, the latter depends on the transportation network. Finally, the IS4CEM relies on the power grid for its functioning.

Each HCC (specifically, its information system) uses the IS4CEM to obtain availability information on the other HCCs and to get information about emergencies. Each HCC uses the communication network and the power grid.

The transportation network is used by the service requestor and by all citizens. Its information system dialogs with the IS4CEM to update the transportation network status. Finally, the environment influences all the system components.

More details about such scenario can be found in [31] if the reader is familiar with the Unified Modeling Language (UML).

4.2 Federated agent-based modeling and simulation software architecture

The FedABMS is basically based on the Recursive Porous Agent Simulation Toolkit (Repast) [11] for the agent simulations, the High Level Architecture[35] and one of its own RTI-implementation, that is the poRTIco project [9]. Moreover we present OMNeT++[8] as simulator for the communication network infrastructure.

The HLA is an architecture designed to allow computer simulations to communicate to other computer simulations regardless of the computing platforms. HLA is composed of different components:

1. The simulations themselves or more generally a *federate*. A federate can be a simulator as well as a supporting utility or even an interface to instruments;

2. The *Runtime Infrastructure* (RTI) which is the operating system for the *federation* (i.e., the distributed simulation). It provides different services to manage federate-to-federate interactions, data, time synchronization;

3. The runtime interface, also called the *RTI-ambassador*, which allows federate to interact with the RTI and to invoke its own services;

4. The *federate-ambassador* which is an interface for the RTI to communicate with the federate itself;

5. The passive collection of simulation data and monitoring of simulation activities

6. The support of interfaces for live participants (e.g. instrumented platforms, live command and control systems)

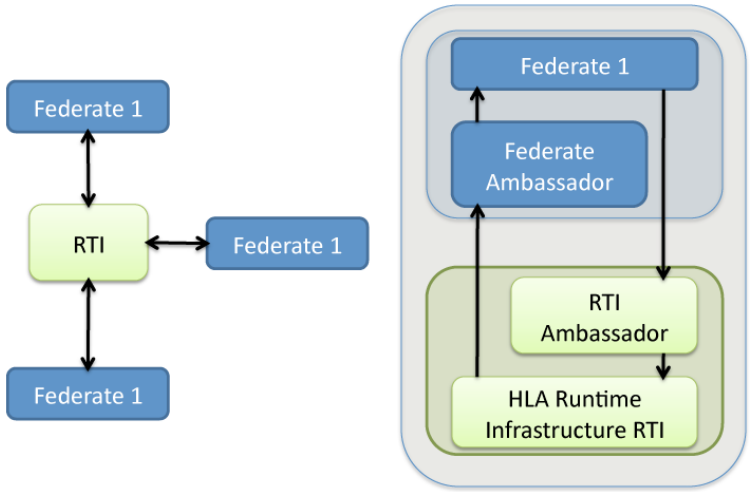


Figure 4.3: HLA architecture of a federation and of a federate

The HLA defines two objects models for the description of the shared elements inside the federation and to describe the federate itself[35]:

Federation Object Model describes the set of objects, attributes and interactions which are shared across a federation

Simulation Object Model describes the simulation (federate) in terms of the types of objects, attributes and interactions it can offer to future federations

Repast is one of the several agent modeling toolkit available on Internet. It is an open-source project developed by the Aragonne National Laboratory. Repast is implemented in different programming languages, and for our framework we have chosen the Java language version. Repast includes different features that are generally present in most of the agent simulation toolkit:

- fully object oriented;

- event driven or time steps approach;
- templates to build agents behavior and agent adaptation (e.g., neural networks, genetic algorithms, regressions);
- context environment where agents reside and interact with;
- projections to describe relation among members inside a context (e.g., network model, grid model, GIS model);
- graph and visualization tools;

Repast cannot be run in parallel, even though in literature there are two projects [60] and [34] in which they applied the HLA approach to distribute and parallelize the simulator. Such an approach can potentially introduce too much delay due to the HLA architecture, since it was not designed to parallelize a simulator itself.

In our framework we have developed an HLA-Repast version, which allows to Repast to exchange information and events/interactions with external sector simulators so that it is possible to conduct a detailed *what-if* analysis applying a *system-of-systems* engineering approach.

The poRTIco project is an Open Source project implemented in Java, which implements most of the RTI interfaces: it lacks of ownership management (possibility to modify the owner of an object) and data management. Moreover it does not support optimistic synchronization algorithm[39] for the time-synchronization of federates inside the federation.

OMNeT++ is an event driven simulation framework to model and to simulate communication networks and, more in general, computer network systems. The OMNeT++ architecture is based on components and it is completely modular. An HLA-OMNeT++ version has been implemented to exchange information and to interact with the HLA-Repast.

4.2.1 Federated ABM&S methodology

Federated ABM&S requires three main steps: 1) system model partitioning; 2) agent-based modeling; 3) agents model refinement.

Let us discuss how Federated ABM&S is applied to critical infrastructures. First of all we define a vertical boundary to separate the organizational and functional model of the whole system, from the physical model of each sub-system. Then, at the physical layer, we draw horizontal boundaries among the sub-system (see fig.4.4).

The ABM&S takes the key role in the modeling of organizational and functional layer. Specialized models and simulation frameworks are used to model the physical layer.

In our example (see figure 4.4) the power grid, the communication network, the IT system, users and operators are modeled as agents. The agent's capabilities and behavior are used to model interaction rules, and system functionalities. A user could interact with an IT system requesting a particular web page, or a person interacts with the power grid requesting a specified amount of electricity. The communication network provide functionalities to route messages, and the IT system provide functionalities to store and to visualize documents or to publish a web site, and so on.

The agent-based model is federated with the specific models of infrastructures (e.g., the HLA-OMNeT++ simulator) as well as of actors (see chapter 5). For example, in our case study, IT system is composed of resources such as disks, CPUs, memories, network interface cards. To implement its high level functionalities (modeled in the agent-based model), the IT system uses its resources. The detailed simulation model of the IT system is defined using queueing network models.

Successively we could model other infrastructure as the Power grid using specific framework such as the Load-flow electrical grid simulator (e.g., e-Agora [5] which we have used in collaboration with

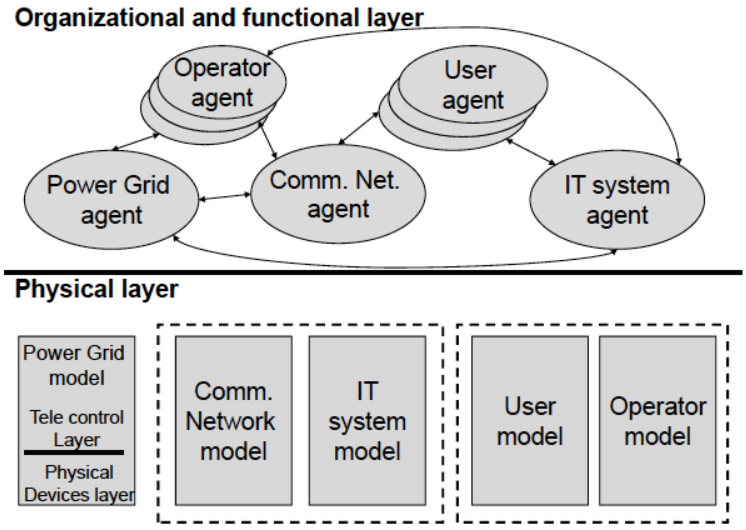


Figure 4.4: Federates Agent-based modeling of critical infrastructure

the ENEA[7] in the CRESCO project [3]). Moreover, applying the recursive property of federated agent-based modeling, the power grid could be again modeled as two interacting sub-systems (as shown in figure 4.4): the tele-control system, modeled with an IT modeling and simulation framework; and the power supply system, modeled with a power grid modeling and simulation framework. Similarly, the behavior of operators and users could be detailed using external specialized models.

4.2.2 The HLA-Repast

We have used Repast to design the model of the organizational and functional layer (see fig. 4.4) as well as to load the simulation scenario. Generally in Repast there are at least two classes: an agent-class that describes the behavior of an agent and a model class that coordinates the setup and running of the model. The ba-

sic class to build a model is the `SimpleModel` class. The developer needs to extend the SimpleModel class to build his/her own model. Specifically, the developer should, at least, override the `setup()` and `buildMoldel()` methods. The `setup()` method is used to load and generate the scenario, while the `buildModel()` is for generating and to setup the agents. The model class will start also the simulation and schedules the agent-events. Schedule can be modified extending the `BasicAction` class in which we define the order and the time of events. All agents are simple Java class that, as in our case, if we are interested in displaying them and defining their relation in a space we have to extend the `Drawable` interface class.

4.2.2.1 The HLA-Repast architecture

The HLA-Repast is composed of the *Repast-Ambassador*, the *CriticalInfrastructure-Model* and the agents classes. The *Repast-Ambassador* is developed in Java and uses directly the library provided by the poRTIco project. The *CriticalInfrastructure-Model*, aside from setting and building the scenario as well as the agents, creates the federation in which all other federates will participate. Once all federates will be connected to the federation, the *CriticalInfrastructure-Model* will start the simulation. Its own method `buildSchedule()` builds the schedule which will synchronize agent activities with the federation time.

The relation among the agents is depicted in the simplified class entity diagram in figure 4.2.2.1. From the figure we can figure out that both humans and infrastructures derive from the agent class. Every agent has a geographic position and three methods that represent the behavior of the agent subdivided in three phases: *pre-step, step and post-step*. The *pre-step* is the phase in which the agent can get information from the external environment and from other agents. The *step* represent the behavior of the agent. The *post-step* is the

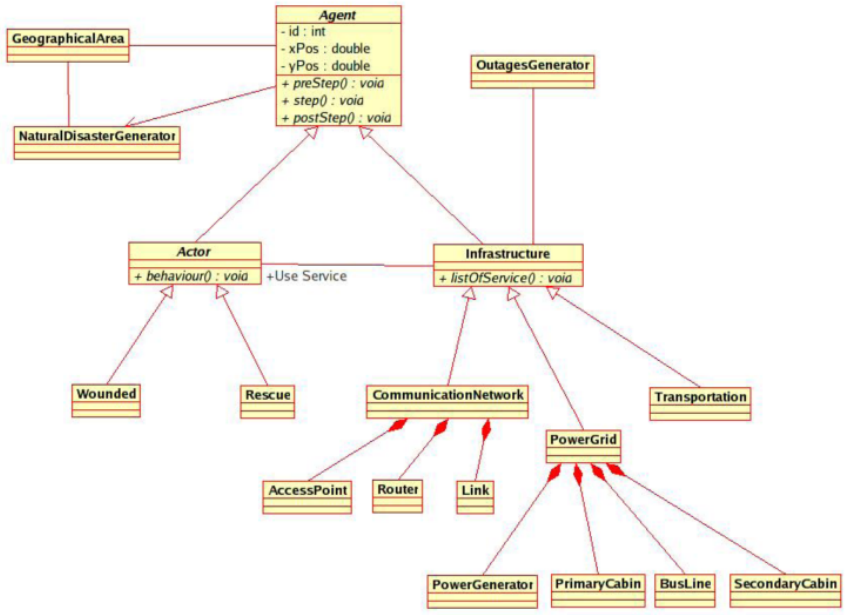


Figure 4.5: Entity Class Diagram of HLA-Repast for the critical infrastructure scenario

phase during which the agent evaluates the decision taken and if all constraint have been respected.

The actor is the class that uses services provided by infrastructures and can be both the workload for infrastructures and specific users for the evaluation of a scenario. The infrastructure can provide one or more services, which can be used by actors as well as by heterogenous infrastructures. An infrastructures is composed of multiple devices, but such devices are generally not agent themselves since they are not adaptive systems.

4.2.2.2 An example with the Communication Network Agent

If an agent wants to communicate with an other one, it calls the method `sendMessage` of CommNet agent, which will prepare a well formatted message and send it to HLA-OMNeT++. HLA-OMNeT++ will simulate the delivery of the message on the physical network model. When the message will reach the destination, HLA-OMNeT++ sends an info message to the CommNet agent notifying that the message has been processed and that it will reach the destination at a specific time. The CommNet agent uses the transmission time calculated by HLA-OMNeT++ to schedule the event "message received". When the simulation time in Repast is equal to the delivery time of the message, the CommNet calls the method `receiveMessage` of the receiver agent, thus to deliver effectively the message to the receiver agent. Moreover the CommNet agent has a map of all components in HLA-OMNeT++. In this way the agents can control and set the status of every component. This solution allows to, for example, turn on/off a network device when there is a failure in the PowerGrid region which supplies such node, or when a hardware or software failure arises.

4.2.3 The HLA-OMNeT++

An OMNeT++ model is obtained composing one or more, hierarchically nested, modules. The lower level modules of the hierarchy, named *Simple Modules*, are programmed in C++ using the simulation library and extending the `cSimpleModule` class. Simple Modules generate and react to events and they implement the behavior of a modeled object (e.g. a network node or a link). Each module is characterized by a set of parameters that can be used to parametrize the module behavior. Simple module can be combined to compose more complex modules, named compound modules. Simple and/or com-

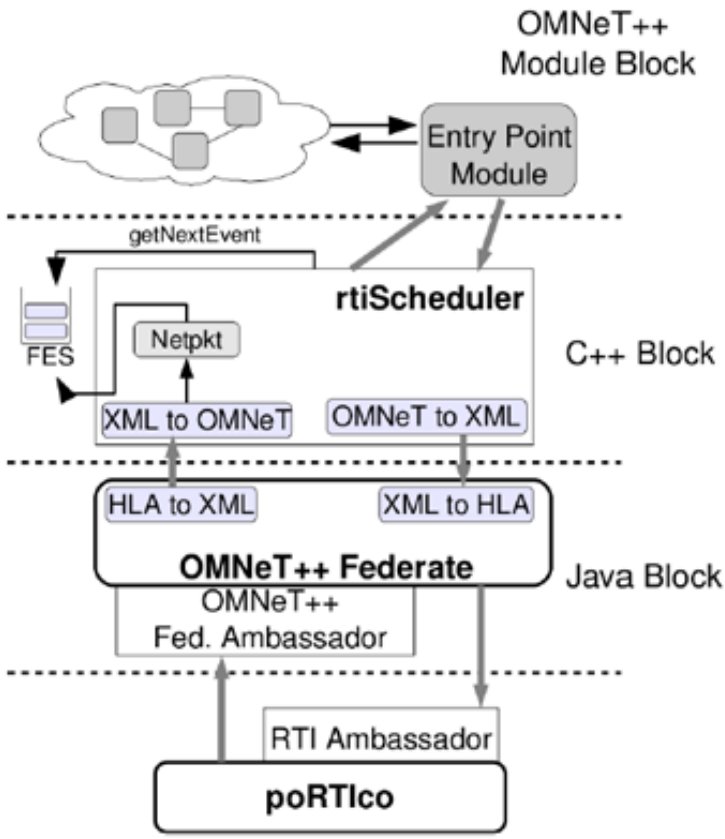


Figure 4.6: HLA-OMNeT++ architecture

pound modules communicate by exchanging messages. Messages can be used to model events, network messages, packets, bits, signals or other. A message in OMNeT++ is an instance of the class `cMessage` and it can be used by any module. Generally it is necessary to define a module which is the traffic generator of the simulation that defines the distribution of traffic, the dimension of packages and so on. All OMNET++ events are queued in the *Future Events Set* (FES) queue. The scheduler extracts messages from the FES in arrival time

and priority order. OMNeT++ offers two methods to send messages: the `sendDelayed()` method to send messages with a specified delay; and the `send()` method to send messages with zero delay. To make OMNeT++ IEEE 1516 compliant we need: 1) to design a new HLA compliant scheduler for OMNeT++, named `rtiScheduler` and 2) to design a Federate Ambassador that allows the interaction between poRTIco and the `rtiScheduler` other than the OMNeT++ modules, written in C++.

4.2.3.1 The HLA-OMNeT++ federate architecture

The HLA-OMNeT++ is composed of three blocks (see figure 4.6). The first one (bottom of the figure) is the interface between the OMNeT++ scheduler and the RTI. It is a Java module that implements the OMNeT++ federate, which communicates directly with the RTI-Ambassador, and the OMNeT++ Federate Ambassador (OFA) that interacts directly with the RTI. The HLA compliant scheduler is implemented by a C++ module, the `rtiScheduler` (middle of the figure). The `rtiScheduler` guarantees the synchronization with other federates. The OMNeT++ Federate and the `rtiScheduler` communicate using the client/server paradigm implemented by a socket based interprocess communication and an XML based communication protocol. The third block is composed by the OMNeT++ modules that implements the communication network simulation model.

```
message NetPkt
{
fields:
        string srcAddress;
        string destAddress;
        string destAddress;
        string payload;
        string destStatus;
```

```
        string id;
        string protocol;
        unsigned int pointerTopology;
}
```

Listing 4.1: The NetPkt defined using the NED language

The network model designer will work only at this level without caring about how events are scheduled; she/he has to define the *Entry Point Module* (the traffic generator module) that will "communicate" with other federates and initializes the simulation of every packet. Moreover the model designer has to define the network topology and the simulation model.

4.2.3.2 The HLA-OMNeT++ Scheduler

OMNeT++ offers the abstract class `cScheduler` to customize the scheduler. The `cScheduler` class defines four virtual method that are: `startRun()`, to start a simulation; `endRun()`, to terminate the simulation; `executionResumed()`, to restart the simulation from a pause; `getNextEvent()`, to extract the next event to process from the the FES (`NULL` is returned if the FES is empty).

Moreover we have defined other two methods that make the `rtiScheduler` independent from the modeled network topology and the Local RTI Component. Such virtual methods are: `setInterfModule(cModule *entryPointModule)`, used to set the OMNeT++ module that will receive the arriving request from the federation; `sendResponseToRTI(cMessage *msg)`, used to reply requests from the federation. In the following we describe the main virtual methods implemented.

startRun() This method is used to initialize the simulation and to activate the `msgServer` (see figure 4.6), a thread responsible to

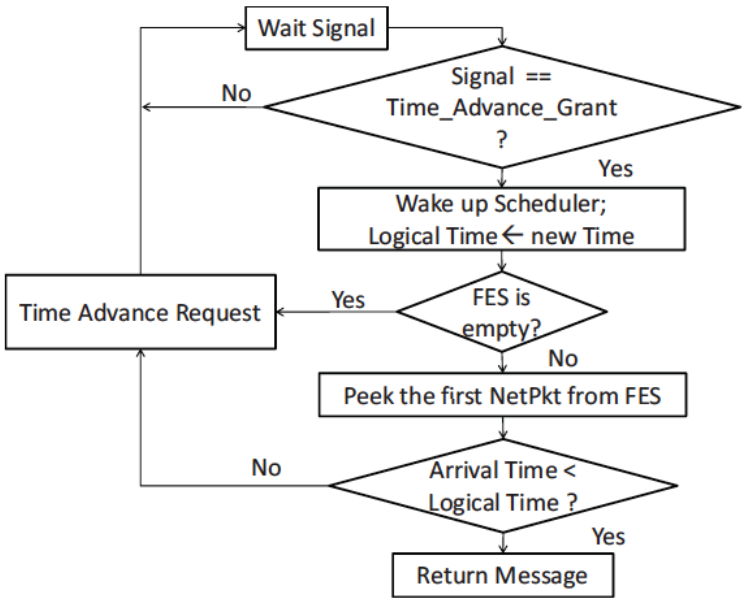


Figure 4.7: Flow chart of `*getNextEvent()` method

manage the XML messages received/ sent from/to the Federate Ambassador.

setInterfModule(cModule *entryPointModule) This method defines which is the OMNET++ module that receives messages from the federation. All messages from federation are always received from the `entryPointModule` module (see figure 4.6).

sendResponseToRTI(cMessage *msg) This method can be used by any OMNET++ module that wants to send a message to the federation. Usually the `entryPointModule` is delegated to manage and to send the reply to the federation. The `rtiScheduler` receives the message (a NetPkt message) from the `entryPointModule`. Then it parses the NetPkt and creates the `commRespToRTI` message that is sent to the Federate

Ambassador.

***getNextEvent()** This method implement the selection of the next event from the FES (see figure 4.7). If there is an event in the queue, and the current time corresponds to the delivery time, the message is returned, otherwise the method return `NULL`. In a distributed simulation, an empty FES does not necessarily means that the simulation is terminated. Then we have modified the scheduler to generate, in case of an empty FES, a `Time Advance Request` to the RTI and to wait the arrival of a `Time Advance Grant` signal from the RTI. The `Time Advance Grant` signal wakes up the scheduler that checks again the FES.

4.3 Preliminary Interdependencies Analysis

Starting from the case study presented in section 4.1, we have studied four different scenarios that consider different crisis which effect the communication network and the transportation system. We suppose that all scenarios are characterized by a burst of help request, generated by the wounded agents at the beginning of the simulation. The other characteristics of the scenarios are defined as in the following:

Basic scenario. Here no failures on the communication network nodes and traffic jams are considered. This scenario is taken as reference scenario for the comparisons of results.

CommNet scenario. In this scenario only the communication network nodes and links experiments respectively faults and congestions. We do not model the details of a failure, that could be hardware and/or software. The average time to failure of a node or link is 5 minutes and the recovery time is 10 minutes. Nodes or links subject to failure are chosen randomly.

Parameter	Value
Num. of HCCs	3
Num. of rescue agents	10
Avg. route round trip time (rtt)	5 min.
Avg. route rtt (during congestions)	7-30 min.
Route inter-congestion time	30 min.
Num. of wounded	10 - 50
Node/link mean time to failure	10 min.
Node/link mean recovery time	5 min.

Table 4.1: The main simulation parameters

TranspSys scenario. Here we introduce only traffic congestion (the communication network work properly). The round trip time of a congested route range from 14 to 60 minutes. The routes which will experiment a congestion are chosen randomly and the inter-congestion time is 30 minutes. When a route is congested, it still in this state forever.

CommNet plus TranspSys scenario This is the more complex scenario while both communication network nodes/links and transportation network will experiment weakness, as defined in CommNet and TranspSys scenarios.

In all scenarios we fix the number of Health Care Centers, the number of rescue agents, and the average route round trip time. Each node of the power grid supply a subset of the communication network nodes, HCCs and routes (traffic light). Some simulation parameters are summarized in table 4.1.

To reduce the scenario complexity we suppose that during a simulation no failures effect the power grid. We suppose also that the rescue agents are uniformly distributed among the HCCs, that is we have about 3 rescue agents for each HCC. We remember that each wounded agent could be reached from at least one HCC. The transportation

Priority	Percent of wounded agents	Time to live
LOW	40%	12h
MEDIUM	30%	6h
HIGH	20%	30min
VERY HIGH	10%	15min

Table 4.2: Classes of wounded agents, their percentage and time to live

network is modeled as a reachability matrix $TN = \{r_{i,j}\}^{n \times m}$, where $r_{i,j} = 0$ means that it is impossible (or prohibitive) to reach the wounded agent j from HCC i, otherwise $r_{i,j} = k$ means that the rescue agent from HCC i will pick up the wounded agent j using route k. n and m denote respectively the number of HCCs and wounded agents.

In all scenarios the goal is to rescue as much wounded agents as possible, as fast as possible. From this study is possible to know how much wounded agents are rescued and in how much time. The simulation results could help in making decisions that will improve the performances of a first hid service.

Each wounded agent is characterized by an average time to live, that is the remaining time to live after that the help request is generated. We suppose that the help requests are not equals. Some wounded agents have a time to live lower than other, and a request generated from a seriously wounded agent has an higher priority, The request priority is used by the HCCs and by the IS4CEM to schedule the aids. Table 4.2 resumes the classes of help requests.

We have studied all the scenarios ranging the number of wounded agents from 10 to 50, thus to considered different loads on the infrastructures. After the first request, each wounded agent could generate a new one if he/she is not picked up in the time estimated by the

IS4CEM, on the basis of the state information received from HCCs, rescue agents and the transportation network.

We suppose that the crisis is resolved when all the wounded agents are rescued or are died (because not picked up in time or not picked up at all).

We can instrument the simulator to measure any kind of information, from high level, as the number of died wounded agent, to low level, as the number of lost packets in the communication network or the IT system nodes utilization. While the low level information could be used to get knowledge of the individual infrastructure state, high level information could be useful to determine emergent phenomena or could be used by a crisis management team that performs what-if analysis to plane the aids. Therefore we have decided to use the following metrics (and related indexes):

- time needed to resolve the crisis (or crisis resolution time - T_c). In particular we measure the absolute value of the crisis resolution time and, for each scenario, the downgrade of the crisis resolution time (δT_c) respect the reference scenario. δT_c is defined as follow: $\delta T_c = \frac{T_c^i - T_c^0}{T_c^i}$, where T_c^0 is the crisis resolution time for basic scenario and T_c^i is the crisis resolution time for the other scenarios.

- Time to rescue a wounded agent (or wounded agent pick up time - T_r). What we measure is the average time that a rescue agent needs to rescue a wounded agent. Also for this metric, in each scenario we measure the downgrade respect the reference scenario. ($\delta T_r \frac{T_r^i - T_r^0}{T_r^i}$, where T_r^0 is the wounded pick up time for basic scenario and T_c^i is the wounded pick up time for the other scenarios.

- Number of rescued wounded agents, that is the number of live wounded agents at the end of the simulation. In particular we

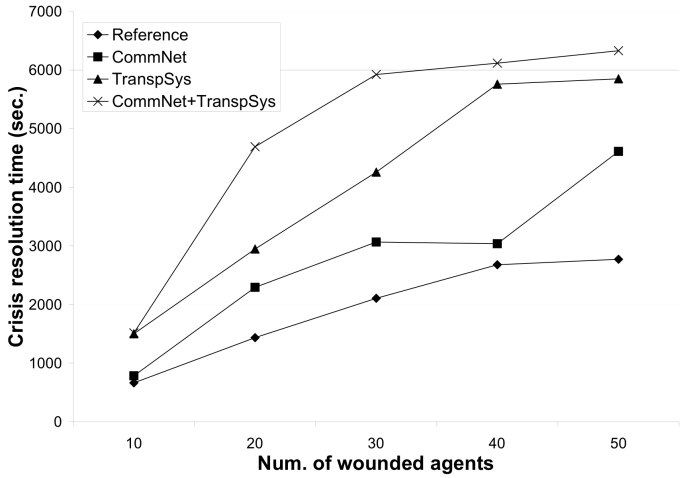


Figure 4.8: Crisis resolution time.

measure the percentage of wounded agents that still live after the crisis.

- Number of died wounded agents. In particular we measure the percentage of wounded agents died during the crisis. We classify who died without any aid and who died after a rescue agent arrives.

The trend of the crisis resolution time is reported in figure 4.8. As expected, T_c degrades when the number of wounded agents increase and also when weaknesses in the different infrastructures are introduced. We point out that the communication network infrastructure is more flexible and adaptable than the transportation system. Indeed the communication network is initially capable to manage the fault of different nodes adapting the packet routing, degrading its performance only when a significant number of link is disrupted, or when the load generated by the wounded agents is too high to be

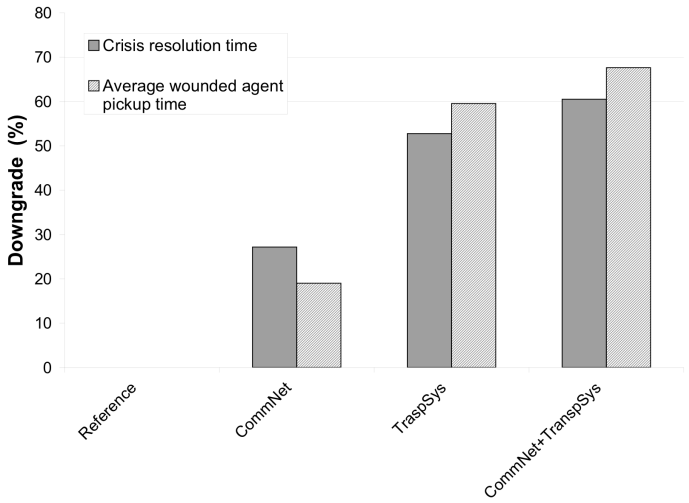


Figure 4.9: Downgrade of the crisis resolution time and of the average wounded agent pickup time.

managed in presence of reduced capacity. In our scenarios, faults in the communication network results only in a 27% degrade of crisis resolution time, while the congestion of the transportation system has a more severe impact. Indeed in the TransSys scenario $\delta T_c = 52.7\%$ (see fig.4.9). We observe an unexpected results when we observe the combination of the CommNet and TranspSys scenarios. Here T_c is not the sum of the crisis resolution time measured in the CommNet and TranspSys scenarios, indeed the measure a downgrade is 60.5%.

Also, the average wounded agent pick up time degrades as weaknesses are introduced in the system (see fig. 4.9. Respect the reference scenario, failures in the communication network produce a downgrade of the 19%, while congestions in the transportation system is more critical, degrading the time to pick up a wounded agent of the 59.53%. This metric also catch the emergent phenomena previ-

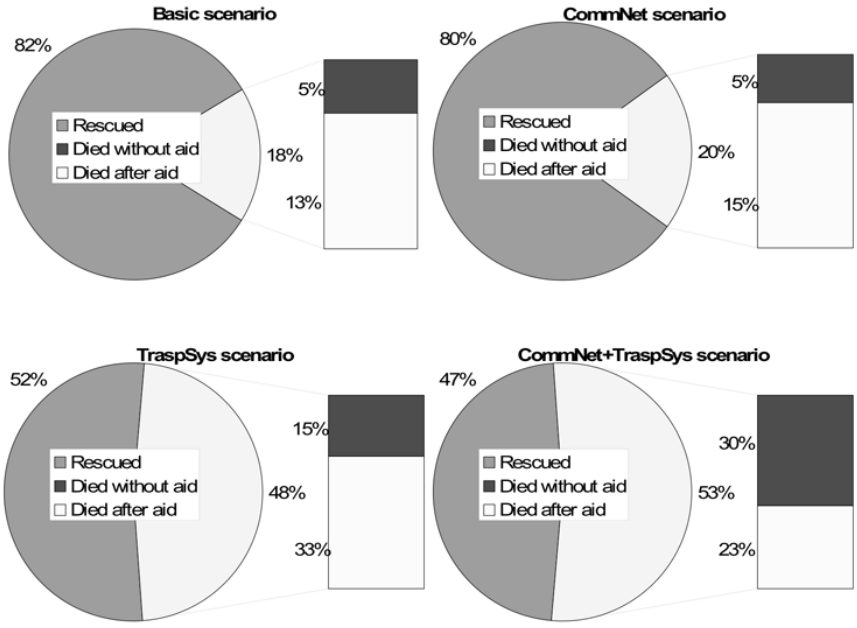


Figure 4.10: Percentage of the rescued and died wounded agents. Died wounded agents are classified as died because not picked up and died because picked up too late.

ously observed that is, the combination of weaknesses in the communication network and in the transportation system does not results in a linear combination of the downgrade, indeed in this scenario we observe $\delta T_r = 67\%$.

Weaknesses in the communication network, as defined in our model, do not impact the percentage of wounded agent that die during the crisis. In figure 4.10 we observe the 18% of wounded agents died in the reference scenario and 20% died in the CommNet scenario. A strong impact is indeed observed when the transportation system is congested. In this case the half of wounded agents died (48%),

15% because do not receive any aid and 33% because picked up too late. Adding faults in the communication network do not increase too much the number of wounded agent died (53%) but heavily increase the number of wounded agents that do not receive any aid (30%). While the congestion of the transportation system slow down the rescue agents, the wounded agents generate more aid requests that congested the communication network, which capacity is already reduced by fault on link and routers. Then the rescue agents do not receive help requests in time, or not at all.

4.4 Concluding Remarks

In this chapter we have presented an implementation of our Fed-ABMS model. We have adopted the standard HLA for the time synchronization and objects as well as interactions management. We have studied a specific case study that points the attention on how to details the model of the IT systems in critical infrastructures. Despite its simplicity, the considered case study is sufficient to show how emergent phenomena could be observed and how simulation helps in quantifying them. For example, we have observed that combining two scenarios that introduce crisis on different infrastructure do not produce foreseeable results. Simulation helps in quantifying them.

Chapter 5

ActivitySim: workload generator for Infrastructure Simulation

5.1 Introduction

Most infrastructure sectors rely on an underlying network that gets used by individual people and business entities, or alternatively speaking: there is a demand for the service that the network supplies. A non-exhaustive list of examples includes the phone network and the Internet that satisfy our the communication needs of the population, the road network that meets the demand for mobility, the electric power grid that satisfies our thirst for electricity. A full-fledged simulation capability that allows to run what-if scenarios as part of a course-of-action analysis requires the following, using the Internet as an illustrative example:

1. Accurate models of the *network topology*. For the Internet, the IP-level connectivity graph with capacity information satisfies this requirement.

2. Abstract models of the sector-specific *processes* on the network. For the Internet, we need models for the protocols used on the Internet (http, TCP, IP, email, Ethernet, 802.11, etc.).

3. Dynamic models of *demand* for the network service where changes in demand are an emergent property (as opposed to an input). For the Internet, we need a tool that provides realistic sets of Internet traffic sessions with origin, destination, and transmission size information. While some preliminary models for demand generation in this sense exist (e.g., call models such as described in [71]), this is a largely open research area.

Demand on networks is largely generated by people as part of their daily activities, such as driving to work, using energy to cook or to heat the house, using water and sewage systems, making phone calls, etc.[1] Thus, an accurate *model for the daily activities* of individuals is a pre-requisite for our simulation capability. An agent-based approach is the only modeling paradigm that allows us to generate demand shocks as an emergent property of the simulation. To stick to communication networks as an example, communication demand in emergency situations is different from a baseline demand because (i) individuals have evacuated in large numbers leading to a geographic demand shift, (ii) logistics, organization (such as organizing a return) and emotional turmoil leads to increased call volumes.

We describe our agent-based approach to activity modeling, called ActivitySim. Activity modeling is ActivitySim is a scalable simulation tool. It relies on a synthetic, but statistically accurate population of the US that was obtained using disaggregation methods applied to US census data [15].

The ActivitySim modeling paradigm is based on a first-principle approach with respect to social modeling. Our main focus in this report is the model methodology, software implementation and scaling. Thorough validation and testing is reserved for future work.

[1] There are cases, where demand is harder to attribute directly to activities of individuals, such as the water use in nuclear power plants. The point here is that a sizable fraction of demand is generated and directly attributable by individuals and individual households.

ActivitySim is part of a family of simulation applications that follow the SimCore modeling paradigm. SimCore [51] is a scalable open-source discrete event-driven simulation engine. SimCore applications seamlessly integrate with each other by exchanging events. Other SimCore applications include sector-specific simulators, such as MIITS-NetSim, and the transportation simulator FastTrans as well as individual demand generation simulators such as SessionSim [52] for communication simulations. We give two examples of how SimCore applications work together: (i) ActivitySim provides input to SessionSim, which generates calls based on activities of agents. The SessionSim output (ie Internet sessions) are sent as events to MIITS -NetSim [89], which then routes these sessions over the network topology. (ii) Activities from ActivitySim that lead to location changes create a demand on the transportation network by generating a trip between locations. This can be fed as an event to FastTrans, which routes this trip over the transportation network and feeds back to ActivitySim at what time the trip was completed.

ActivitySim agents are *utility-driven*: each activity gives a certain amount of utility to an agent depending on how long the activity is being executed. Agents also have *priority functions* for activity types, where the priority of an activity intuitively increases usually with the time that has passed since the activity was last executed. Activity types have *constraints* that allow us to guide the timing of certain activities (such as work should happen during the business hours). As optional modules, we (i) allow agents to have *personality types* (guided by standard models from social sciences) and (ii) let agents guide their activity type selection by the *needs* that they want to satisfy. We describe the ActivitySim modeling in more detail in section 5.3.

The main loop of an agent consists of planning and re-planning its scheduled activities and evaluation the resulting updated schedule with respect to the agent's objective function. The objective function

takes into account predicted utility as well as priority and constraints violations. The schedule optimization step can be performed through your favorite optimization method, such as the gradient method, local search, simulated annealing, or taboo search. We describe the optimization loop in section 5.3

We have implemented and tested ActivitySim on agent populations of up to 2.6M agents in a case for Twin Cities, MN. We present scaling results in section 5.4.

5.2 The ActivitySim Architecture

ActivitySim is a C++ agent-based model that can run on workstations as well as high performance computing clusters. The supporting software architecture consists of agent and discrete event simulation (DES) frameworks, and libraries for graph processing, logging, partitioning, asserts, random number generation, and message passing (see Figure 5.1). More details follow.

5.2.1 The SimCore DES Framework

SimCore is a library for building large-scale distributed-memory, discrete event simulations (DES)[51] using the discrete event engine from the Parallel Real-time Immersive Modeling Environment (PRIME)[10] or passing events, event queue maintenance, and synchronization. It has previously been used for packet-level and session-level telecommunications network simulations[89] and fast queue-based transportation simulations, called FastTrans. The important concepts and classes within SimCore are Entity, Service, Info, and Profile. An Entity is a class that represents a simulation object such as a person, location, or facility. A Service is a class that is used to implement the behavior of an entity and operates like an event handler. Services are attached to Entities. An Info is a class that

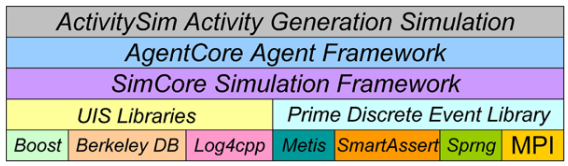


Figure 5.1: Overview of Software Architecture

represents an event that can be scheduled and supplies additional data items and is processed by a Service. Infos are passed between Entities (more typically between the Services) to trigger an action. A Profile is a way of providing runtime specification of default parameter settings for different types of Entities, Services, and Infos.

5.2.2 The SimCore Agent Layer - AgentCore

AgentCore is a "reactive agent" extension to SimCore which adds classes for agent implementation. It is based on the behavior-based layer (BBL) from Muller's agent architecture[62]. The Agent class extends an Entity to include functionality (as methods) to perceive, think, and act and a Cognitor for processing rules or patterns of behavior. A Cognitor Service (known as the CognitorHandler) to the agent-part of a simulation object processes Think Infos (events) by calling the Agent's think method. An implementation would typically call its Cognitor's think method, but could certainly do more. Perceive means to gather the current values of the simulation world's facts or state variables. Think means to process the agent's production-rules or patterns of behavior by means of a Cognitor to perform actions and to cause other events to be scheduled. Act means to execute the actions determined by thinking. In implementations, the think method typically encompasses thinking and acting.

A pattern of behavior (POB) class has state, being active or inactive, has an activation condition based on the Agent's current facts,

action code to execute if it is activated, a success condition that determines successful termination, and a failure condition that determines that a failure has occurred. A POB can have multiple execution steps. Each step can be interrupted or interruptions can occur between steps. Multiple POBs can be active at a time, but not all will be executing.

The Cognitor implements the control cycle model for thinking. An InterRap version is supplied (from Muller's agent architecture), though others can be added based on the beliefs, desires, and intentions (BDI) model, etc. InterRap processes POBs in its think method by managing completed POBs, checking for newly triggered POBs, and executing active POB steps.

Only Entities in a simulation that perform "intelligent" behavior should be agents, such as a Person entity changing and adapting their activity schedule. Other Entities such as locations and households do not need to be agents, though Agent-Entities and Entities can still interact through their Services.

5.2.3 ActivitySim

The *ActivitySim* agent-based simulation software provides daily activity schedule generation and execution for a synthetic population. It operates as a standalone model for population analysis, as well as coupled with other SimCore infrastructure models such as transportation and telecommunications to study inter-dependency effects in baseline and emergency scenarios. Persons, Locations, Households, and Zones comprise the entity types used to represent a model of a geographical area. A Person is also an agent that reasons about daily activity schedules. State (current activity and location), demographics, activity location choices, and a current schedule are all part of a Person. A Location tracks Persons as they participate in its' activities. A Household is associated with a Location, has aggregated

income, and members (ex. family). A Zone is an aggregation of Locations used when selecting where an activity will take place.

The Persons, Locations, Households, and Zones are provided as input at runtime along with a specification of a set of activity types including utility and priority function parameters. An activity set can be very specific (ex. *sleep, personal care, lunch, dinner, leisure*) or more general (ex. *home, work, school, shopping, social recreation, daycare*). A Person's schedule consists of a sequence of these activities with start time, activity, location, and duration. A "Next Activity" POB triggered by a Think event causes each Person to reevaluate their activity schedule and add new activities as required (see figure 5.2). A Person will "think again" during their next activity. In addition to the utility-based activity selection (described in Section 5.3), other methodologies can be added easily. Use of pre-generated schedules and random activity generation are also supported.

Single activity execution is an independent process from activity schedule generation. Each scheduled activity is executed as a sequence of four Person-level events and two Location-level events. A PersonDepart event causes a Person to complete their current activity and leave that Location (initiating a RemoveFrom Location event). A PersonTransport event allows a Person travel time between locations. A PersonArrive event places a person in the new activity at the new location (initiating a MoveTo Location event). Finally, a PersonDone event updates a Person's state per their new activity and location. When running on a parallel cluster, entities are distributed randomly across processors. The Person-level events are executed on the processor where the Person entity resides. Messaging is required between processors when "moving to" or "removing from" an activity at a Location.

The model output consists of configurable logging of details of entity and service creation, along with simulation progression. Event files show the individual event details for each Person and Location

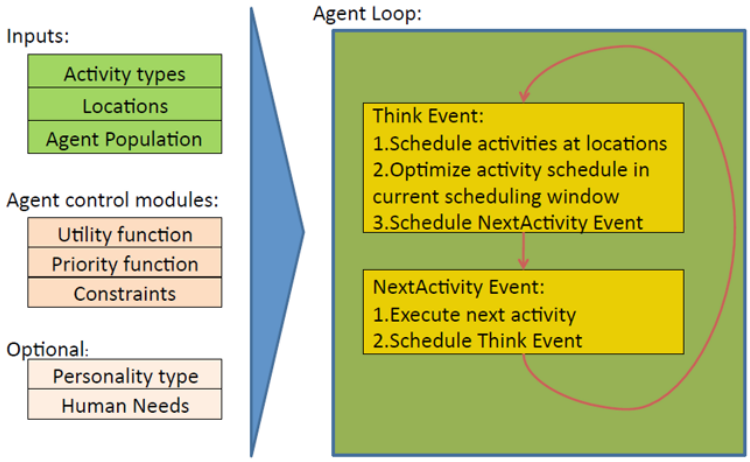


Figure 5.2: ActivitySim Inputs, Modules, and Think Loop

during activity selection and execution. Output is selectable for Person schedules and/or counts of Persons per activity for each Location at a single time or at regular intervals.

5.3 ActivitySim Modeling Paradigms

We explain the ActivitySim modeling philosophy along the main building blocks as illustrated in Figure 5.2. Recall that ActivitySim takes as input a population of agents and a set of locations. Each resulting ActivitySim agent is characterized by a set of demographic attributes, such as age, gender, social status, personality type and home location. ActivitySim creates agents based on data sets derived from US census data (see Section 7.4 for details). Input locations are physical locations, characterized through latitude/longitude coordinates that represent either a business location (obtained from standard business data sources such as the Dunn&Bradstreet database) or a private residence.

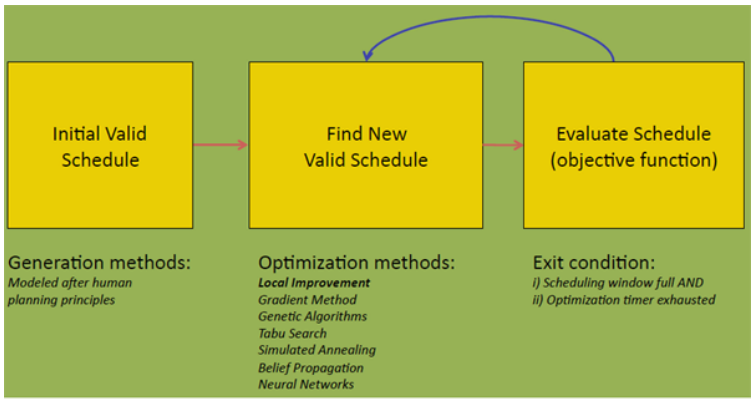


Figure 5.3: Optimization Loop

As part of the main loop of an agent, it re-plans its scheduled activities during a Think-event that is executed at the end of an activity or at any time during an activity. In a Think-event, an agent optimizes its planned future schedule and evaluates it with respect to the its objective function. The objective function is influenced by the notions of takes utility functions, priority functions, constraints, needs function, and personality types, which we will explain towards the end of this section. Figure 5.3 illustrates the optimization loop that forms the Think-event in more detail. The schedule optimization step can be performed through your favorite optimization method, such as the gradient method, local search, simulated annealing, or taboo search.

The utility-driven activity selection model uses the modules of utility, priority and constraints with additional (optional) modules of needs and personality types that impact the schedule optimization loop shown in Figure 5.3. The notion of utility functions for activity selection builds on earlier work [88]. Generally an individual tries to optimize just a little set of activities while he/she has just a general idea of what he will do in the next days. So agents try to improve

their current schedules, deciding often at last minute which activities they will perform. To grasp this concept intuitively, consider that we have a generic idea of what we will do in the next days (we know that we will go to sleep at the end of the day) but we can not say with great precision at what time we will perform it. Moreover an individual generally can schedule some particular activities that will happen in the far future (vacation, checkup, meeting) but will optimize every particular activity only when she/he is really close to the event. Inspired by these concepts, we have created a general model where given an initial schedule, every agent tries to optimize its schedule according to his personal characteristics. Starting with a basic set of already scheduled activities, the agent searches for a valid better schedule inside a limited future time window, which we call the sliding window. The new schedule is then evaluated by an objective function and if the new schedule is better then previous, the process is iterated. So the general algorithm can be described as follows:

1. *Starting Schedule*: Either a starting schedule is selected or it is the scheduled calculated in a previous step;

2. *Local Optimization*: The schedule is modified within the sliding window according to local optimization rules;

3. *Validity Check*: All activities that violate constraints are deleted from schedule;

4. *Schedule Evaluation*: The objective function evaluates the total schedule. If the objective function value is greater than the previous schedule, it becomes the new starting schedule. If the number of optimization steps has not yet reached a maximum threshold, the algorithm loops back to step 2;

5. *Append*: If no modifications to the original schedules have been

accepted, a random activity is appended at the end of scheduled queue at a random future point in time.

The optimization algorithm that was used for our test runs is a local optimization scheme that uses priority functions to order unscheduled activities, utility functions to choose the duration, and the needs and personality type functions influence the sorting of unscheduled activities. The output of the local optimization (i.e., a new schedule) is evaluated by the objective function. The idea behind the objective function is that we want to penalize schedules that do not consider the value of utility, the priority, and/or the location. The modules that impact the objective function evaluation are described in more detail in the following subsections. To introduce some notation, write $X = \{x_1, x_2, ..., x_N\}$ for the set of all possible activities, which we call activity types, e.g. {sleep, personal care, work, lunch, leisure, dinner}. This set is arbitrary, but fixed. Formally, let $S = \{s_1, s_2, ..., s_n\}$ be the set of scheduled activities, then the $i - th$ scheduled activity s_i is characterized as,

$$si = (a_i, l_i, t_i, d_i)$$

where

a_i indicates that the $i - th$ scheduled activity is of type a

l_i is the location

t_i is the starting time

d_i is the duration.

5.3.1 Utility Functions

The utility that an agent gains from performing an activity can ideally be explained in monetary terms. In our case the level of satisfaction

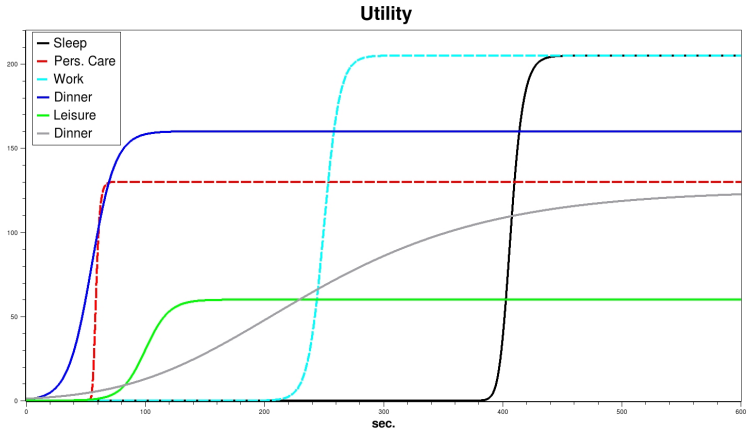


Figure 5.4: Example Utility Functions

cannot be mapped to a unit, but what matters are relative values of satisfaction. Every activity type is associated with an utility function. The utility of performing an activity is typically a function of the duration for which the activity is performed. There usually are lower and upper bounds for the utility that an agent can gain from performing an activity: consider sleep as an example, where even 5 minutes of sleep can have high utility and maximum utility is reached after about 8 hours for most people; sleeping 15 hours usually comes with less utility than sleeping only 8 hours. Utility functions have the following characteristics:

i) $U\prime(t) \geq 0$;

ii) $U : \mathcal{R} \rightarrow [U_{min}, U_{max}]$ with U_{max} maximum marginal utility and with U_{min} minimum marginal utility;

iii) Let $y \in \mathcal{R} : U\prime\prime(y) = 0 \Rightarrow$ for every $x \in \mathcal{R} : x \geq y\ U''(x) \leq 0$.

Figure 5.4 shows the utility functions as a function of duration that we have chosen for some of our test runs. Our utility functions

have relatively steep transitions from a low level to a maximum level of utility. We have consulted with social scientists and economists to obtain these utility functions, but they should be considered a preliminary set. The functional form of the utility (adopted from [88]) is defined as follows for an activity type a:

$$U_a = U_a^{min} + \frac{U_a^{max} - U_a^{min}}{(1 + \exp[-\beta\ (\nu_a - \alpha_a)])^{\gamma_a}}$$

where

ν is the duration ($\nu \geq 0$)

U^{max} is the upper asymptote of the curve ($U^{max} > 0$)

U^{min} is the lower asymptote of the curve ($U^{min} \leq 0$)

α is the parameter of x-translation

β is the parameter of the slope

γ is the parameter of the inflection point.

5.3.2 Constraints

Constraints are non-negotiable conditions that a schedule must satisfy in order to be considered valid. Activities can be different from person to person and can be biased by age, personalities, marital status, family degree, employed status and so on. Thus, every agent chooses its activities from a set of activity types that is specifically tuned to the individual agent. ActivitySim imposes constraints on every activity type that must be obeyed during the activity selection. These constrains are minimum duration, maximum duration, earliest start time, latest start time, earliest end time and latest end time. Every location has a set of activity types that can be performed at

that location (such as work, if it is a business location; or shopping, if it is a retail location). Thus similar constraints are imposed on a per-location-per-activity type basis regarding possible start and end times of activities. These constraints are akin to opening hours.

5.3.3 Priority functions

A priority function is a function of time and represents the priority of an activity in a particular instant in time. This concept is particularly important during an emergency scenario where even though, for example, a person needs to eat, all evacuation activities must precede the activity eating. The priority function characteristics can be summarized as follows:

i) is function of time;

ii) is monotonically increasing;

iii) $\lim_{t\to\infty} P(t) = 1$;

iv) $\lim_{t\to 0} P(t) = 0$.

The priority function is set to zero as soon as an activity is performed. If an activity is selected always at the same time, the priority function is also a periodic function. Performing an activity type usually becomes more urgent with the time that has passed since the last execution of the activity. Thus, the time axis defined in these function represents the time since last execution of the activity type.

Figure 5.5 illustrates the sigmoid priority functions we have used in some our test runs. More formally, the priority function is similar to the previous utility function. If an *activies* has been already scheduled then the priority value is equal to zero. Let a be an unscheduled activity we have:

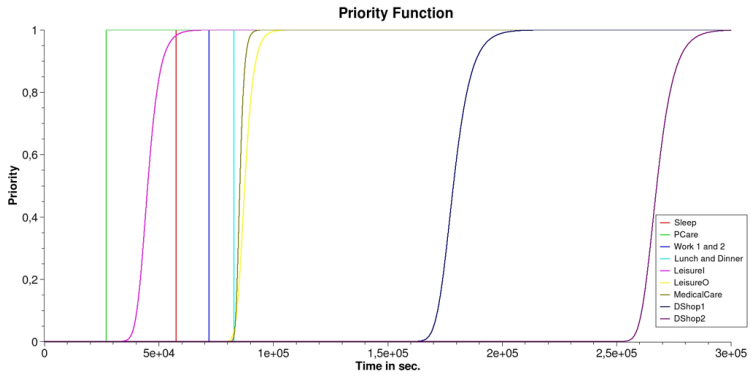


Figure 5.5: Example Priority Functions

$$P_a = \frac{1}{(1 + \exp[-\beta\ ((t_a - T_a) - \alpha_a)])^{\gamma_a}}$$

where:

$T_a = startTime_a + duration_a$ is the last time that activity a was performed

t_a is the current time

α, β, γ have the same meaning presented in the utility function.

5.3.4 Needs function

An individual is driven by his needs. A need is a dynamic characteristic of a person that decrease in the space of one or more days and it is directly influenced by some activities. So a person performs some activities to satisfy his needs. At the same time an activities can also influence negatively other needs. For example, the activity "work" influences negatively needs as "energy". A need is described by the inverse of the presented priority function. Performing specific

activities will bring the need value to his optimum while others will decrease it to 0.

5.3.5 Location selection function

The location function represents the attractiveness of a zone. The function decrease with distance, time spent to reach it and/or other cost parameters (money, energy, ecc..) and increase with the number of activities that is possible to perform in such location as well as with personal preferences.

In our test runs the location function is simply a decreasing function of the distance. Let a be the selected activity and l_i, l_j the current and next location respectively. We thus have that

$$L^a_{i,j} = \frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$$

where (x_i, y_i) and (x_j, y_j) are the coordinates of locations l_i and l_j respectively.

5.3.6 Objective function

The objective function is a linear function that evaluates the new schedule. It takes as an input all scheduled and unscheduled activities, the utility, the priority, and the location functions. Recall that $X = \{x_1, ..., x_N\}$ is the set of all the activity types considered, and that $S = \{s_1, ..., s_n\}$ is the set of n such scheduled activities (possibly with repetitions), where the $i - th$ activity is characterized as $s_i = (a_i, l_i, t_i, d_i)$. Then we define the objective function as follows:

$$Obj(S) = \frac{1}{n} \sum_{i=1}^{n} \left[U_{a_i}(d_i) - \alpha \sum_{k=1}^{N} P_{x_k}(t_i) \right] + \frac{1}{n} \sum_{i=1}^{n} \beta L(l_1, l_{i+1})$$

where U, P and L are the utility, priority, and the location functions defined above, and α and β are weighting parameters. The objective

function evaluates a proposed schedule at the beginning of each new activity by giving positive points for achieved utility, negative points for incurred non-zero priority values at the beginning of each scheduled activity for every activity type, and by penalizing long travel times to new locations. Finding good schedules is obviously quite a challenge that calls for smart optimization schemes, which shall be presented in the next section.

5.3.7 Optimization Algorithm

The optimization loop can either try to evaluate a large number of new schedules thru many iterations or it can attempt to invest cycles in finding a relatively small number of good new schedules by taking into account the utility, priority, and constraint modules. It is open which strategy is better and we are currently experimenting with light-weight simulated annealing approaches with very cheap neighbor functions. However, we have used the local optimization algorithm described below for our test runs, which invests many cycles into finding a few good new schedules.

The currently implemented algorithm uses set operators to modify the schedule. These operators are: *insert*, *substitution* and *adjustment*. The insert operator puts an unscheduled activities, respecting its constraints, between two already scheduled activities or between the last activities and the end of sliding window. The substitution tries to substitute a random scheduled activity with an unscheduled one. In such case is used the same start time and duration of substituted activity. The adjustment sets the duration close to its optimum value that is:

$$\nu = \alpha - \frac{1}{\beta} \lg\left(\frac{1}{\sqrt[\gamma]{1 - \frac{\varepsilon}{U_{max} - U_{min}}}} - 1\right) \text{ with } \varepsilon > 0$$

Algorithm 1 Algorithm for the adjustment operator

$N \leftarrow |S|$ {With S the set of scheduled activities}
for $i \leftarrow 1$ to $N - 1$ **do**
  $j \leftarrow i + 1$
  $u_i \leftarrow U(s_i)$ {with U the utility function and s_i i-th activity}
  $u_j \leftarrow U(s_j)$
  if $u_i \geq u_j$ **then**
    $\nu_i \leftarrow calcOptimumDuration(s_i)$ {Adjustment on ith activity duration}
    if $\nu_i < s_{i,dur}$ **then**
      $s_{i,dur} \leftarrow \min(\nu_i, s_{i,MAX\_DUR})$
    else
      $s_{i,dur} \leftarrow \min(s_{j,start} - s_{i,start}, s_{i,MAX\_DUR})$
    $gap = s_{j,start} - (s_{i,start} s_{i,dur})$
    $move(s_{j,start}, gap)$
  else
    $\nu_j = calcOptimumDuration(s_j)$ {Adjustment on j-th activity start time}
    if $\nu_j < (s_{j,dur}$ **then**
      $s_{j,start} \leftarrow s_{j,start} - \min(\nu_j, s_{j,MAX\_DUR})$
    else
      $s_(j, dur) \leftarrow s_{j,start} - \min(s_{j,start} - (s_{i,start} + s_{i,dur}), s_{j,MAX\_DUR})$
    $gap \leftarrow s_{j,start} - (s_{i,start} s_{i,dur})$
    $expand(s_{i,dur}, gap)$

In our experiments we have used $\varepsilon = 10\%$ of U_{max}. The adjustment is always first applied to the activity that has the highest utility value and then to the other one. The goal of the algorithm is to fill the sliding window with the highest priority activities. Once the activity type has been selected, the location with highest location function

value will be selected. The algorithm tries to fill the sliding window using insert and substitution operators. For every operator, the total utility function is computed as:
let $A = a_1, ..a_N$ be the scheduled activities inside the sliding window then:

$$U_{tot} = \sum_{i=1}^{N} U_{a_i}$$

We select the operator with the highest total utility. Since the insert and substitution operators do not consider optimum value for duration of an activity, the adjustment operator is applied to all scheduled activities.

5.4 Large example results

We used ActivitySim with the utility-driven scheduling to model daily activities in Twin Cities, MN. The synthetic population was constructed to statistically match the 2000 population demographics at the census block group level. The synthetic population consists of 2,592,906 individuals (as agents) living in about one million households, with an additional 487,725 locations representing actual schools, businesses, shops, or restaurants. A schedule of activities to undertake each day is created, each with a start and stop time, activity type, and location. There are sixteen types of activities: *home, work, shopping, visiting, social recreation, other, passenger server, school, college, dining out, service appointments, medical appointments, daycare, elementary school, junior high school, and high school.*

Information about the time, duration, and location of activities was obtained from the National Transportation Survey[15]. Each person agent was given an assigned set of locations based on the surveys. Locations were not provided for all activities, but only for a subset

that were relevant to an agent's demographics and associated survey. Only one location was provided for home, work, passenger server, school, college, daycare, elementary school, junior high school, and high school. Four or more were provided for the remaining activity types. The *home* activity was allowed to start at any hour during the day and for any length of time. *Work* was limited to 4-10 hours at a time at any time during the day, while *junior high school* as limited to 4-6 hours at a time starting between 8 AM and 10 AM, ending between 11 AM and 4 PM. Two example schedules are shown in Figure 5.6. The first shows a child's schedule going to *junior high school* every morning. The second shows an adult's schedule who goes to *work*, spends time at *home*, has *medical appointments*, *goes shopping*, and participates in *social recreation*.

Though multiple activities of the same type appear in sequence (ex. *home*), these initial results on a larger population are promising. More tuning of the utility and priority parameters is required to reduce the gaps.

The Twin Cities synthetic population was run on the LANL Institutional Computing parallel Coyote cluster (2,580 x AMD opteron nodes @ 2.6 GHz with 2 processors per node, Voltaire InfiniBand interconnect, 10.2 TeraBytes RAM) distributed across 8, 16, 32, 64, and 128 processors. Each was run for 10 simulated days. In all cases reading the input data took about 1.5 minutes. The average runtime per simulated day is shown in Figure 5.7.

We see that using 32 processors is sufficient for running this problem size, with little additional gains for more.

5.5 Concluding remarks

In this chapter we have seen the developing of a framework to develop a robust activity generator as part of infrastructure simulations for

Day	Start Time	Location	Activity	Duration
0	8:00:08	600542	JrHighSchool	6:14:06
0	22:47:22	267425	Home	0:47:00
1	6:58:30	600542	JrHighSchool	4:28:04
1	19:26:43	267425	Home	2:52:02
2	7:42:55	600542	JrHighSchool	5:20:05
2	17:27:05	267425	Home	3:10:03
2	23:06:11	267425	Home	0:52:00

Day	Start Time	Location	Activity	Duration
0	8:09:08	435630	Medical	3:12:03
0	12:22:12	435630	Medical	3:14:03
0	18:34:18	264485	Home	3:22:03
0	23:05:23	313655	Work	7:02:07
1	7:17:31	264485	Home	4:51:04
1	14:06:38	459512	Medical	1:21:01
1	16:45:40	264485	Home	1:29:01
1	20:43:44	470567	Social	0:30:00
1	21:24:45	264485	Home	1:05:01
1	23:49:47	313655	Work	9:38:09
2	10:11:58	264485	Home	1:11:01
2	11:57:59	313655	Work	4:47:04
2	18:31:06	264485	Home	2:04:02
2	21:44:09	264485	Home	2:04:02
3	13:54:25	313655	Work	6:05:06
3	21:58:33	264485	Home	1:36:01
3	23:56:35	313655	Work	8:22:08
4	10:10:46	406496	Retail	2:08:02
4	12:59:48	330051	Social	2:19:02
4	15:52:51	264485	Home	5:24:05

Figure 5.6: Example of produced schedule

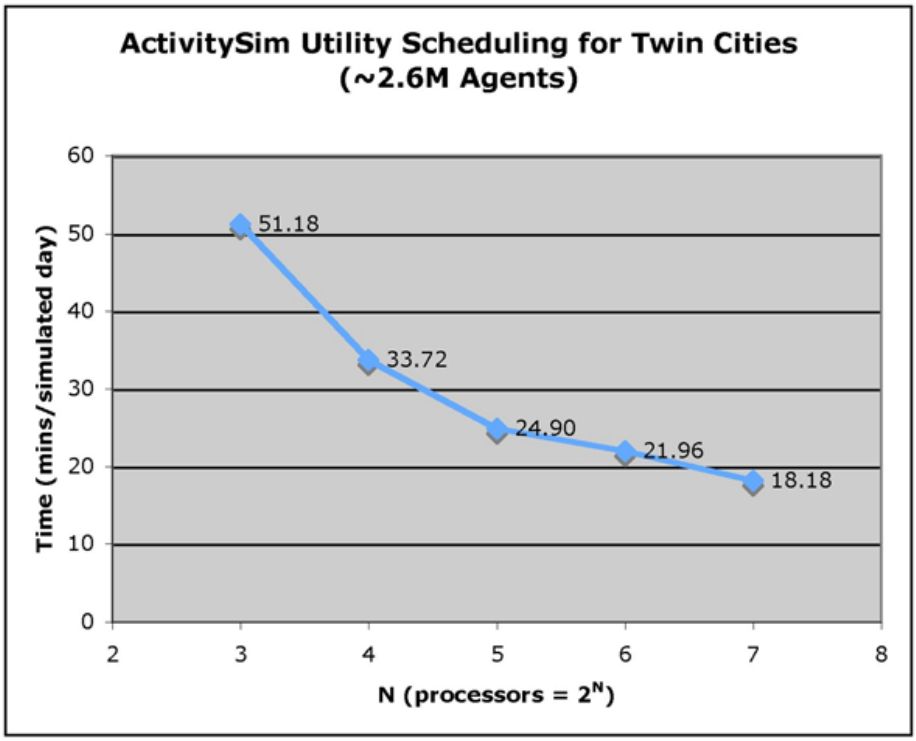


Figure 5.7: Running time per simulated day

baseline and emergency scenarios. Tunable utility functions, priority functions, needs functions, and constraints allow for variability in each agent's daily activities adding a touch of realism. The framework allows to include additional scheduler generators and optimization strategies (such as the gradient method, local search, simulated annealing, or taboo search), and it is applicable to large populations like New York or Twin Cities. Evaluation and analysis metrics for schedule evaluation is still an open question and could be a future challenge in this research field.

Chapter 6

FastTrans: a distributed, large scale simulator for Transportation System

6.1 Introduction

How do we build systems that can realistically simulate, at a high level of detail, the traffic patterns resulting from the activities of tens of millions of people within a geographic region? What strategies do we employ such that we maximize the usage of the processor cycles available to us? And how do we scale these systems to hundreds, even thousands of processors on high performance computing clusters, so that they execute in a fraction of real time?

In this chapter we present various aspects of designing, building and optimizing FastTrans – a scalable, parallel microsimulator for transportation networks that can route and simulate tens of millions of vehicular trips on real-world road networks. Using parallel, discrete-event simulation techniques [39] and distributed-memory algorithms, we are able to model transportation networks of large geographic regions – consisting of millions of road network elements and over 20 million vehicles – and scale these simulations to execute on large, high performance clusters up to 20 times faster than real time.

Such applications are used in the emerging field of infrastructure modeling, where simulating the behavior of millions of entities and

their interactions with the various interdependent infrastructure networks – transportation, communication and electric power, to name a few – demand significant computational resources and scalable implementations. At the Los Alamos National Laboratory, FastTrans is one of the key modules in a suite of simulators developed for infrastructure modeling that have been built using a common parallel simulation framework, which allows for easy integration of the various modules. For instance, we have been able to integrate FastTrans with ActivitySim, the agent-based simulator introduced in chapter 5, that provides daily activity generation, scheduling and execution for a synthetic population of intelligent agents. This allows us to generate realistic activity schedules for millions of intelligent agents, route the tens of millions of vehicular trips generated from these activities, and observe how traffic conditions and the variations they cause in expected travel times impact activity scheduling and vice versa. Organization. We present the key aspects of the road-network and activity modeling philosophy in Section 6.2. We discuss the software architecture of FastTrans, and ActivitySim, and the interaction between these frameworks in Section 6.3. In Section 6.4, we describe the various design choices for routing, partitioning, and load balancing that were used to optimize the performance of our simulations. Section 6.5 contains scaling results for parallel runs. Conclusions are in Section 6.6.

6.2 Traffic Modeling Through Realistic Activity Generation

Approaches to transportation simulation have spanned a variety of simulation paradigms: from fluid-based aggregate models to detailed microsimulations (where the behavior of each individual vehicle is simulated) and from time-stepped approaches to discrete-event mod-

els. Fluid models nicely describe the macroscopic behavior of networks, while microsimulations are more suitable for problems that require a higher level of spatial granularity – study of vehicular emissions, for instance.

Traditionally, traffic microsimulations have employed a time-stepped cellular-automata approach [30], [14], [70], [20].

A prominent example of this is TRANSIMS [20], developed at the Los Alamos National Laboratory which models vehicular dynamics such as lane changing and emissions, but at a high computational cost. The presented traffic model involves a queue-based discrete-event approach to microsimulation – which sacrifices some amount of spatial granularity for speed – starting with the premise that the questions of interest are vehicular dynamics at the intersection and link levels. A queue-based approach can be used to quickly answer time-critical questions like evacuation times and optimal exit routes from a city in an emergency situation. Further, such models can also be used to guide infrastructure planning activities like the impact of building a new road or a relief-route, or conversely, the impact of disabling a route. The queue-based traffic simulation model was first developed in [37]. A time-stepped parallel implementation of this approach was later developed in [32]. Charypar et al. [33] observed that time-stepped computations are frequently unnecessary because in a given road network, there are large numbers of links on which the traffic flow densities are very low. Updating these links every time step are often "null ops" and therefore, waste computational cycles. To overcome this inefficiency, the authors proposed a discrete-event queue based model for a sequential, single-processor environment. A parallel discrete-event approach to traffic microsimulations was first developed in [68], though the modeling paradigm employed here is conceptually closer to the cellular-automata approach. Experimental results presented in that paper on a 1000-node grid network indicate speed-ups of up to 1000 over realtime.

The FastTrans approach is to combine the discrete-event queue model with scalable parallelization. This allows us to simulate large-scale, real-world networks and realistic traffic scenarios involving tens of millions of vehicles in a fraction of real time. Also, since FastTrans simulates the behavior of each vehicle or traveler at the individual entity level, it retains some of the advantages of microsimulations. In addition, the congestion model of FastTrans captures the non-localized effects of congestion, allowing us to observe the macroscopic nature of the network.

6.2.1 Queue Model of Road Networks

In the queue model, each road link is modeled as a queue, whose properties are described by two main parameters: their physical capacity – i.e., the number of bumper-to-bumper vehicles that can be accommodated on the link – and the flow rate of the link. The flow rate indicates the number of vehicles that can transit through the link and is calculated by the procedures established in the Highway Capacity Manual [23]. Each queue is attached to a network node which represents a traffic intersection, or a point where the road link diverges (a freeway exit, for example). The scheduling policy for vehicle departures from a node is determined by the type of intersection that is being modeled. Further, to model congestion, FastTrans builds upon the techniques introduced in the transportation simulation literature [30]–[33], [68]. The parameters of flow rate and physical capacity allow us to capture congestion by dynamically adjusting the flow rate (as happens during a lane-closure, for instance) and also by blocking the link when its physical capacity has been reached. In this case, upstream nodes are blocked from adding any further vehicles onto the link, mimicking the behavior of traffic jams that spill backwards. Once vehicles start to leave the downstream congested links, this information is propagated to the upstream links.

6.2.2 Activity Modeling

The workload for the simulator FastTrans is generated by the ActivitySim, presented in the previous chapter. For every activity is associated a *departure* and an *arrive* event. The departure is described by the current and next location in which the action has to be performed, the departure-time and travel-type (walking, car, public transportation, etc...). The arrival time is calculated by the FrastTrans which will invoke the corresponding event in the ActivitySim.

6.3 Software Architecture

All our simulation modules are built on top of SimCore, a generic framework written in C++ that provides application programming interfaces (APIs) for building distributed memory, discrete-event simulation applications. SimCore provides generic constructs like entities and services that can be adapted to build objects in a simulation model. SimCore also provides message objects for communicating between simulation instances in a parallel environment.

For message passing and synchronization, we use the Prime Scalable Simulation Framework (PrimeSSF) [10], a parallel simulation engine that employs a conservative synchronization mechanism. PrimeSSF supports both shared-memory and distributed-memory implementations though, for scalability reasons, all the simulators described in this chapter are pure distributed-memory applications. Message-passing is implemented using the MPI message passing interface [13].

6.3.1 FastTrans Architecture

FastTrans is written in C++ and built using the constructs in the SimCore library. In FastTrans, simulation entities are the fixed elements of the road network – road links and traffic intersections. All the properties of the network – capacity, flow rate, etc – are members of the relevant entity class. The scheduling logic at a traffic intersection is implemented as a service on the traffic-node entity. The modular design allows the scheduling policy to be easily changed by simply replacing one scheduling service with another.

The mobile elements of the simulation (vehicles) are represented using messages. Vehicle objects (messages) are created and destroyed during the start and end of a trip, respectively. The main state variables associated with a vehicle are source, destination and the route vector. Route vectors are computed at the start of the trip by the FastTrans routing module; for this, we maintain a copy of the connectivity graph on each simulation process. The routing algorithm is described in more detail in Section 6.4.1. The input data to FastTrans is the roadnetwork graph for the region being simulated – indicating connectivity, road length, speed limits, lane capacity, etc – and vehicular itineraries indicating source, destination and start time of a given trip. The trip inputs can be read from a file, or when FastTrans is coupled to the activity module, can be sent using messages.

Since FastTrans uses a distributed-memory model, different entities of the road network are created in different memory spaces during simulation start-up. Each simulation process (also known as Logical Process, or LP) in the simulation is an instance of a FastTrans executable running on a compute node.

A design schematic of the distributed architecture of FastTrans is shown in Figure 6.1. Traffic intersections are distributed across LPs according to the partitioning strategy employed – geographic, random, etc., which we describe in more detail in Section 6.4.2. Links

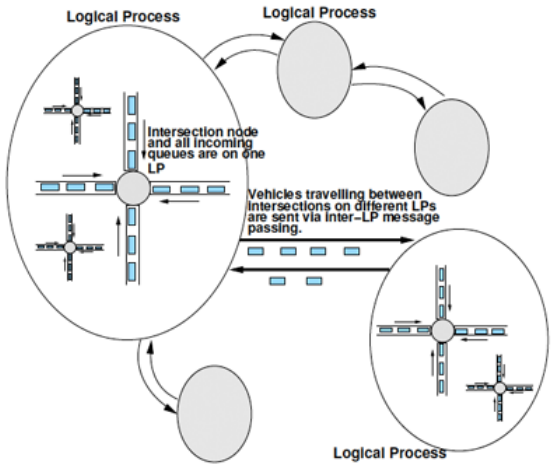


Figure 6.1: Distributed-memory model of FastTrans

(queues) are partitioned in a slightly different manner: each link is placed on the same LP as its terminating point[1] based on the observation that more messages are exchanged between the sink node and the link (vehicle arrival, vehicle dequeue and so on) than between the source node and the link. Placing the link and the sink node on the same process allows us to reduce message-passing overhead.

6.3.2 Integrated Simulations

Both FastTrans and ActivitySim can be compiled as separate applications and run independently of each other, each binary being statically linked against the SimCore, PrimeSSF and MPI libraries. In this case, vehicle trips are pre-computed and fed into the FastTrans module, while for ActivitySim running in independent mode, activities that cause individuals to travel from one location to another are

[1]Note that each link is attached to two intersection end-points – the source node, from where the link originates and the sink node, where the link terminates

processed internally, without feedback regarding actual travel times that would have depended on traffic conditions in the road network.

A more realistic (and interesting) approach would be to combine the two simulators, and observe how traffic conditions and the variations they cause in expected travel times impact activity scheduling, and vice versa. The modular architecture of the simulators and the use of a common simulation framework allows us to integrate the two modules and achieve this feedback mechanism with relative ease.

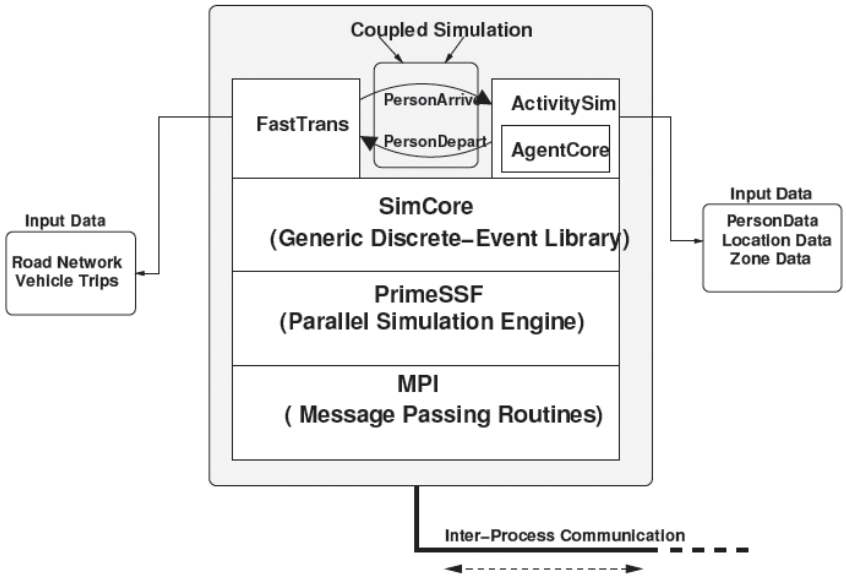


Figure 6.2: Modular software architecture of the integrated simulator

In the coupled simulation scenario, events in ActivitySim that trigger a road trip ("PersonDepart" events, figure 6.2) are sent to FastTrans via messages. The actual trip is simulated inside FastTrans, and upon arrival at the destination, a "Person- Arrive" event is sent to ActivitySim. In this case, unexpected delays in arrival time

due to road network congestion inside FastTrans can trigger recomputing of activity schedules – behavior that is not observed when the simulations are running independently. We note here that the entity partitioning scheme used inside the two modules are completely independent of each other, and so spatial mapping information is needed to map the activity locations inside ActivitySim to elements in the road network.

6.4 Performance Tuning

Detailed microsimulations are frequently used in modeling critical infrastructures such as transportation and communication networks in a given region. Simulations involving natural and man-made disasters often require iterations through multiple scenarios – to determine the best evacuation strategy during an earthquake, for instance – implying that such simulations need to execute much faster than real-time to be useful. When combining the two simulation modules, ActivitySim and FastTrans, we observe that the execution time is overwhelmingly dominated by the FastTrans transportation module (figure 6.3). Consequently, we focus our performance optimization efforts on improving the performance of FastTrans.

6.4.1 Routing in FastTrans

Routing is the most important module of FastTrans – the algorithm and parameters used to route vehicles are critical to the validity of the simulation model. In addition, it also accounts for more than half of the total execution time. We use dynamic routing in FastTrans; calculating routes dynamically allows us to achieve fast responses to congestion with the additional benefit of reduction in the input data

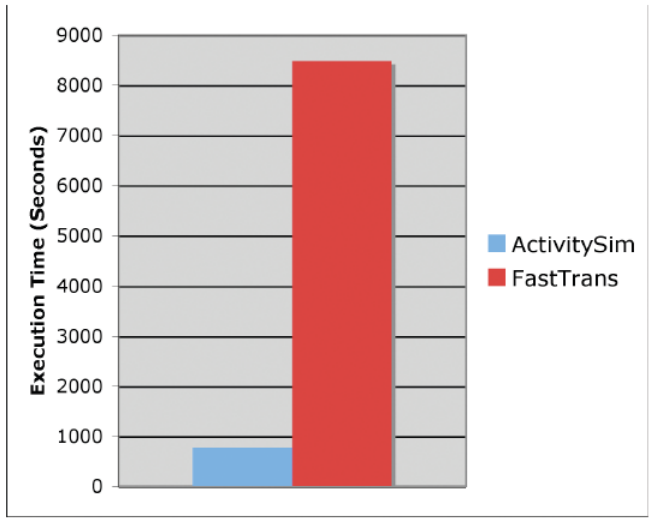


Figure 6.3: Comparison of the execution times of the two modules for a 32-CPU parallel run of a medium sized US city

size[2]. However, route computations on large graphs can easily become a bottleneck, leading to severe performance degradation. We use several optimizations to make online route computations feasible in FastTrans where we require the shortest path between a pair of nodes. While in the worst-case, the asymptotic time complexity of path-computation for a source-destination pair is same as computing a shortest-path tree, we observe that in practice, there are often significant differences in running times, especially when the source and destination are geographically close. The FastTrans framework also allows us to use paths that are not necessarily optimal. This motivates investigation of very fast heuristic algorithms that obtain only near optimal paths.

Routing experiments were conducted with different variations of

[2]If routes were pre-calculated, input file sizes would be significantly bigger

the standard Dijkstra's algorithm [36]. These algorithms were chosen due to recommendations made in Jacob et al. [48]. The algorithms studied were: (a) *Dijkstra*, where shortest-path trees, rooted at the source are constructed for each routing query, (b) *Optimized Dijkstra*, where the search loop is terminated upon finding the shortest path to the destination, (c) $A^\star$ search[45], a variant of Dijkstra that employs a heuristic cost-function to bias the direction of the search towards the destination. Further optimizations, that were used in all these implementations include smart label reset (where only nodes explored in a previous routing computation are re-initialized) and use of efficient data structures.

$A^\star$ search was first proposed in AI literature [45],[79]. In road network graphs, $A^\star$ exploits the near-Euclidean property to expand the shortest path tree in the direction of the destination. This results in the search arriving at the destination node much quicker than Dijkstra, where the search tree is expanded in a circular manner centered at the source node. To bias the search towards the destination, we assign a cost to each intermediate vertex as follows: given source s, and destination t, for each intermediate vertex v, cost $C(v)$ is defined as:

$$C(v) = l(s, v) + D(v, t)$$

where $l(s, v)$ is the shortest path length from s to v, and $D(v, t)$ is the estimated cost from v to t computed as a function of the Euclidean distance between v and t. Since we are interested in the shortest path in terms of time rather than distance, $l(s, v)$ is the time-cost of the path from s to v based on link-speeds and flow-rates (speed), and $D(v, t)$ is defined as:

$$D(v, t) = \frac{E(v,t)}{V_max}$$

where $E(v, t)$ is the Euclidean distance from v to t and V_{max} is the maximum allowable speed in the network. Note that the resulting

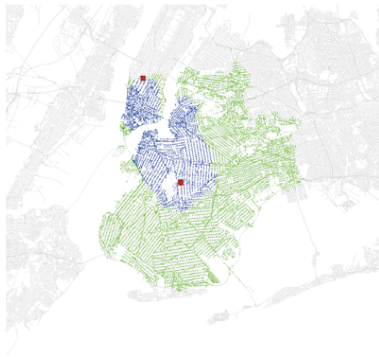


Figure 6.4: Nodes expanded in $A^\star$ (shown in blue) vs. Dijkstra (shown in green). The red squares are the source and destination nodes, with the source being at the center. Dijkstra expands the search tree in all directions from the source node, while $A^\star$ is more directed

Figure 6.5: A routing query where $A^\star$ performs markedly better, expanding only about one percent of the nodes (blue) compared to Dijkstra (green). On average, in our computations $A^\star$ expands 82% lesser nodes than Dijkstra

paths using $A^\star$ are not provably optimal; however with road graph being almost Euclidean, our experiments showed that the paths computed by $A^\star$ are on average only 0.02% longer than the paths computed by Dijkstra.

Figures 6.4 and 6.5 illustrate the performance of $A^\star$ versus an optimized version of Dijkstra on a real-world road network (Northeast region of the United States). Notice that $A^\star$ explores a much smaller fraction of nodes than (even) the optimized Dijkstra.

Figure 6.6 shows, on a logarithmic scale, the speed-up as well as the routing overhead for the three implementations (Dijkstra, op-

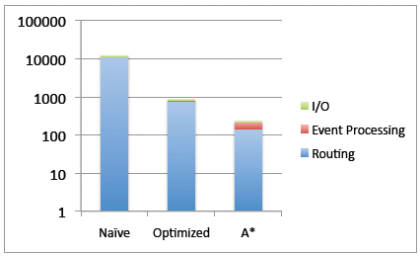


Figure 6.6: Logarithmic chart showing various overheads for Dijkstra, optimized Dijkstra, and $A^\star$ in a serial simulation run

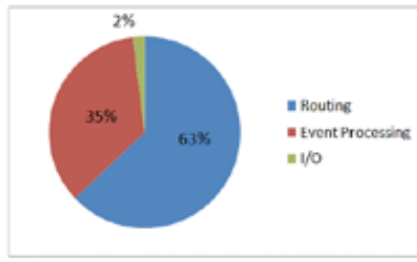


Figure 6.7: Execution profile of FastTrans with $A^\star$ in a serial simulation on a 3 GHz Mac-Pro work-station

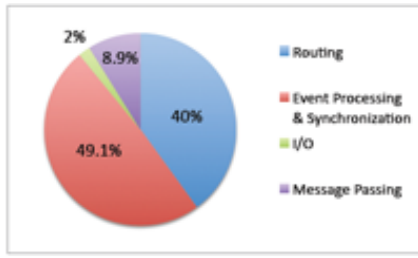


Figure 6.8: Execution profile of FastTrans with $A^\star$ in a parallel simulation on a 32-CPU Linux infiniband I/O cluster.

timized Dijkstra, and $A^\star$) from a code-profiling exercise of a serial simulation run. The serial run was executed on a 3 GHz Mac-Pro work-station for 20000 itineraries that were randomly sampled from the study-set. Clearly, $A^\star$ gives much better performance-accuracy

trade-off than other schemes. Figures 6.7 and 6.8 show the execution profiles for FastTrans with $A^\star$ in serial and parallel settings. In the serial setting (figure 6.7), about 63 percent of the execution time is spent inside the routing module of FastTrans. In the parallel setting, where each simulation process keeps a copy of the entire routing graph, the fraction of time spent in routing decreases (figure 6.8) due to the additional overhead from message passing and synchronization.

6.4.2 Partitioning and Load Balancing

In (standard) synchronized parallel discrete-event simulations, the difference between the simulation-clock of any two logical processes cannot be greater than the look-ahead time value specified for the simulation[3]. Thus, a simulation process with a relatively high computational load, whose simulation clock progresses at a slower (wall-clock) rate than the other processes, slows down the entire system. Essentially, the speed of execution of the simulation is determined by the slowest process. The computational load on a simulation process is determined by the partitioning scheme used to assign simulation objects to compute nodes. In this section, we explore different strategies for partitioning the simulation work load on a highperformance cluster. An ideal partitioning algorithm achieves balanced load, while keeping messaging overhead to a minimum.

1. *Geographic Partitioning*: In FastTrans, the bulk of interprocess messaging occurs when vehicles travel between locations in the road network that have been assigned to different processors. Since vehicular trajectories are, by nature, spatially constrained, a partitioning algorithm that minimizes messaging overhead would assign road-network locations that are geographically close to the same processor. A simple geographic

[3]The look-ahead time values specifies the minimum amount of simulated time it takes for messages to travel between two simulation processes

partitioning scheme divides the simulated geographic region into a uniform rectangular grid, and assigns all road-network entities (road intersections in our case) belonging to a grid cell to the same processor.

2. *Geographic Partitioning with Balanced Entity Distribution*: In real-world road networks, spatial distribution of network elements is far from uniform; node density is much higher within urban downtowns, and decreases significantly as one moves away from the core regions (figure 6.9). Thus, a pure geographic partitioning scheme would result in unequal entity distribution. To over come this, we use a non-uniform rectangular partitioning of the spatial region such that each grid cell now contains an approximately equal numbers of entities. Again all entities in the same grid cell are assigned to the same processor.

3. *Geographic Partitioning with Balanced Event Load Distribution*: From our earlier profiling (figure 6.8), we observe that event-processing accounts for approximately 50% of execution time in parallel scenarios. Furthermore, the number of events generated by the simulation entities also varies with location. The event load distribution across the road network has a pronounced spatial characteristic - busy roads and intersections often tend to be geographically clustered. Figure 6.10 depicts a a heat map of the road network in the New York region, with areas in red being the busiest points in the network, and areas in blue being regions with low traffic volume. While it is generally not possible to accurately determine the event load associated with a given entity without actually running the entire simulation, we could assign an event-weight to an entity by simulating a small sample of the vehicular traffic. Once event-weights are assigned to all entities, we once again do a geographic partitioning assigning entities to processors while balancing event

load distribution across processors.

4. *Geographic Partitioning with Balanced Routing Load Distribution*: This is similar to the previous scheme, with weight assigned to an entity being the number of routing computations (rather than event computations) performed at an entity. Recall that routing also accounts for a significant fraction of execution time. Since, for a given trip, the entire route from source to destination is calculated at the starting point, locations that serve as trip originators – residential locations, business districts, etc. – will generate more routing computations than transit locations with high traffic volume – busy freeways, etc. Figure 6.11 illustrates the spatial distribution of the routing load in the road network.

5. *Geographic Partitioning with Balanced Weighted-Sum of Loads*: Event processing and routing both account for significant chunks of execution time; however, on a percomputation basis, a single routing computation can be up to two orders of magnitude slower than processing a single event. By assigning appropriate weights to each of these computational tasks, we can assign a weighted sum to an entity that approximately represents it is total computational cost. Then, while partitioning geographically, we balance this computational weight across processors. However, choosing the appropriate weights for routing and event processing is a complex task; routing calculations themselves can display enormous variations in running time depending on source and destination. Further total routing overhead itself can change with the size of the graph. Because of the complexities involved, we have not yet completely explored this scheme.

6. *Scatter Partitioning*: In contrast to the previous schemes, this

partitioning scheme assigns entities that lie close to each other to different processors; that is, nearby entities are scattered across the cluster. In the road network data, successive nodes (intersections) are often numbered consecutively. If we define en entity-to-processor assignment as $P = m \bmod N$ (P is the processor to which entity m is assigned and N is the number of processors), then nearby entities will be assigned to different processors. This scheme is motivated by the spatial nature of load distribution on the network – since entities that lie close to each other have similar load characteristics (figures 6.10 and 6.11), by assigning nearby entities to different processors, we spread the computational load across the cluster. The obvious disadvantage to this scheme is the significant number of inter-process messages; now for every node-hop that a vehicle makes, an interprocess message will be generated. However, as we shall see in the following section, this scheme performs surprisingly well.

6.5 Experimental Results

We carried out our experiments on two high performance clusters Coyote [1] and Lobo [2] available at the Los Alamos National Laboratory. Coyote, running 64-bit Fedora Core 3, has 1290 compute nodes, with each node consisting of two 64bit AMD Opteron 2.6 GHz processors and 8GB memory. Lobo has a total of 272 compute nodes with each node containing 16 cores and 32 GB memory, for a total of 4352 cores. Both these clusters use a Voltaire Infiniband high-speed interconnect, and the Panasas parallel file system which provides a theoretical transfer rate of 20 GB/sec. For our scaling studies, we ran our experiments starting from a minimum of 32 processors to a maximum of 1024 processors. Interprocess communication was carried

out using Open MPI message passing interface [13].

6.5.1 Partitioning and Load Balancing

We first present experimental results for the various partitioning schemes for FastTrans described in section 6.4.2.

The various partitioning schemes were tested on the road network in the New York region, consisting of approximately half a million intersections, 1.1 million road links, and over 25 million vehicular trips. Together, these result in about four billion simulation events. All partitioning experiments described in this section were conducted on the Coyote cluster for different processor configurations, ranging from 32 to 512 processors.

Figure 6.15 illustrates the basic performance of the various partitioning schemes in terms of execution speed. Pure geographic partitioning performs the worst, while scatter partitioning is the fastest, outperforming geographic partitioning by about an order of magnitude. Interestingly, performance in terms of message-passing overhead is almost the reverse of execution time – the number of inter-process messages being passed in scatter partitioning (highest overhead) is an order of magnitude higher than geographic (lowest overhead). What we can observe is that message-passing does not take up a significant chunk of execution time.

Execution times for partitioning based on routing-load and event-load are comparable to scatter, especially in the larger processor configurations (256 and 512). At these sizes, the geographical areas assigned to each grid cell become rapidly smaller, resulting in a more balanced load. Ultimately, the fairness of load distribution is the most important criterion for performance, as is made amply clear in Figures 6.17 through 6.21. These figures illustrate the computational load on each processor in the 256 processor set-up, with load

being defined as a weighted sum of event and routing load[4]. Each bar represents the load on one CPU over the entire simulation. The load profile in the best performing partitioning scheme (scatter) is more or less flat, while that in the poorly performing schemes are highly uneven. The Min/Max load ratio depicted in these figures is an useful metric for fairness comparison since the speed of execution in a synchronized simulation is determined by the slowest process; it follows that a low Min/Max ratio will severely degrade performance, with ideal ratio being 1. Figure 6.22 compares (inverse of) this ratio for the various partitioning schemes; note how these correlate with speed of execution (figure 6.15).

To conclude this section, we note that the geographic partitioning schemes based on routing-load and event-load perform comparably to scatter in terms of execution time, and also have very low messaging overheads. Thus as noted earlier, for a specific scenario it may be possible to tune these schemes to outperform scatter partitioning. The tuning parameters, however, would be highly dependent on the specific scenario to be simulated – the event and routing load profile may differ significantly in a simulation of a disaster scenario from the simulation of a normal day. The load patterns would still exhibit spatial clustering – for e.g., there are usually only a few main evacuation routes from a city – and scatter partitioning would continue to achieve a balanced load profile. Generally, for problem domains that exhibit spatial clustering of load, scatter partitioning is a simple and highly effective approach.

[4]For a fixed network and simulation scenario, we can determine, through profiling, the ratio of the relative cost of event processing to routing. The ratio in this scenario is about $1 : 94$

6.5.2 Computational Scaling Results

In the experiments in this section, we test the scaling performance of FastTrans, ActivitySim, and the integrated simulation on different processor configurations. For the FastTrans experiments, in addition to the New York region, we also simulated the smaller region of Twin Cities in Minnesota. This allows us to observe the scaling behavior of FastTrans as a function of the size of the road (routing) graph. The Twin Cities road network consists of approximately 300000 road links and 150000 intersections; the New York graph consists of half a million intersections and about 1.1 million road links. All the experiments in this section use the scatter partitioning scheme, and simulated an entire day is worth of vehicular trips – approximately six million for Twin Cities and 25 million for New York. All partitioning experiments described in this section were conducted on the Lobo cluster for different processor configurations, ranging from 32 to 1024 processors.

Figure 6.23 shows the scaling performance in terms of execution time for FastTrans for the two scenarios. As expected, the size of the New York road graph implies that routing calculations dominate the New York simulation and consequently, execution time falls much more rapidly (as we add more processors) for the New York scenario compared to the Twin Cities scenario. The performance levels off at about 512 CPUs for New York indicating that this may be close to the ideal number of processors for a scenario of comparable size. We expect further improvements with bigger cluster sizes (> 512) for larger graphs.

Memory usage per processor, depicted in figure 6.24 is fairly constant for mid-size cluster runs (< 256), and increases for larger cluster sizes, even though one would expect decreasing memory burden per processor as more processors are used. The reason for this behavior is that memory usage on a compute node is dominated by the size of

the routing data structures. Since each processor keeps a copy of the routing graph (as explained in section 6.4.1), memory usage does not decrease. At larger cluster sizes, the size of the interprocess message buffers[5] causes the memory usage per node to increase, though still very much within the memory capacity of a compute node.

For both cities, the messaging overhead for FastTrans does not vary with the number of simulation processes. This behavior is entirely due to scatter partitioning (see also figure 6.16) – since nearby entities are assigned to different processors, an inter-process message is usually generated for each node (intersection) that a vehicle traverses. Thus, the number of interprocess messages in scatter partitioning essentially depends only on the number of trips and the average path-length (node traversals) for each trip – both of which depend only on the routing and modeling aspects of the simulation. Consequently, messaging overhead does not vary with cluster size. Figures 6.26 through 6.28 show the scaling behavior performance of the integrated simulation in the Twin-Cities scenario, compared to the performance of the modules when run separately. The performance of the integrated simulation follows the performance of the dominant module – thus, the execution time of the integrated simulation is similar to FastTrans, while memory usage resembles that of ActivitySim. Once again, as a result of scatter partitioning, messaging overhead is relatively constant for all three simulators irrespective of the cluster size. The absolute number of messages generated in the integrated simulation is less than that of FastTrans, since the number of vehicular trips that are created dynamically from within ActivitySim is less than the stand-alone version of FastTrans. All simulations for both cities run significantly faster than realtime even on 32 processors. The high realtime speedups allow us to simulate multiple scenarios and provide timely feedback and analysis in real-world

[5]Each process maintains $2N$ communication buffers, where N is the number of processes in the simulation

situations.

6.6 Concluding Remarks

In this chapter we have presented the use of large scale, parallel, discrete-event simulation systems for realistic, microsimulation-based modeling of large urban areas. These included modules for realistic activity generation at the individual level, and a discrete-event queue-based parallel transportation simulator. The modular software architecture employed in building these systems allowed us to easily combine the two modules into a high-fidelity, activity-driven simulation of the road network.

Optimizations in the routing module through heuristics-based routing allow us to perform simulations with significant speed-ups over real time. Further, we discovered that the optimal way to partition the computational workload in our case was to exploit the spatial nature of the road network in a counter-intuitive way – namely to scatter the entities, that is, to explicitly assign nearby entities to different processors in the cluster. A possible direction for further investigation here is to observe if scatter partitioning or similar approaches perform as effectively in other types of networks that exhibit spatial clustering of load.

Experiments on HPC clusters illustrate the scalable nature of the simulation paradigms and software engineering principles.

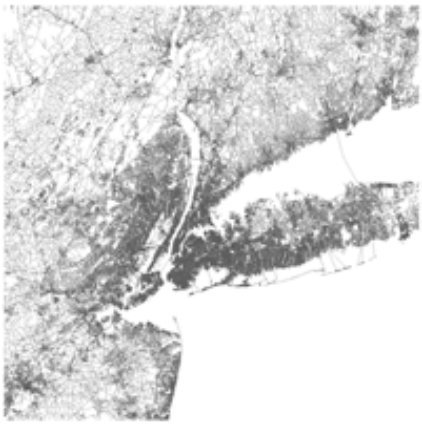


Figure 6.9: Figure illustrating road-network density in the New York region

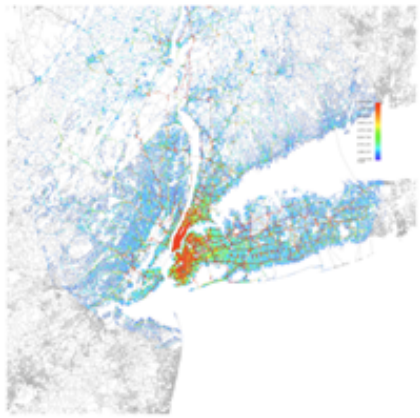


Figure 6.10: Figure illustrating distribution of event load in the New York region

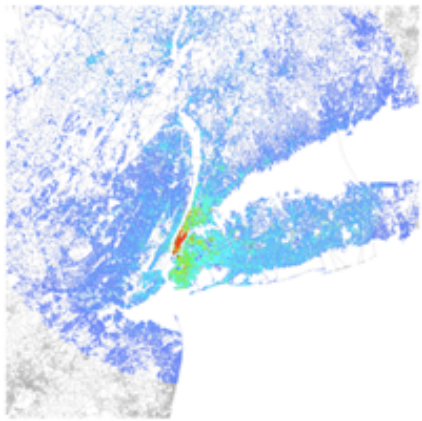


Figure 6.11: Figure illustrating distribution of routing load in the New York region

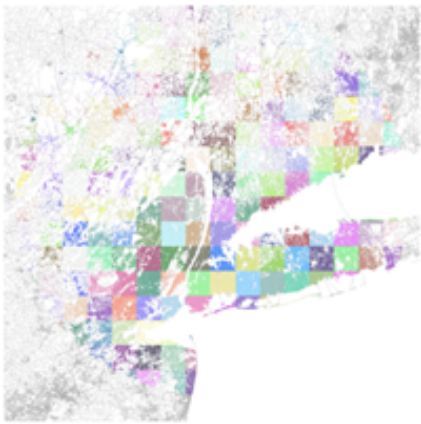


Figure 6.12: Figure illustrating assignment of entities under geographic partitioning scheme. All entities with the same color are assigned to the same processor

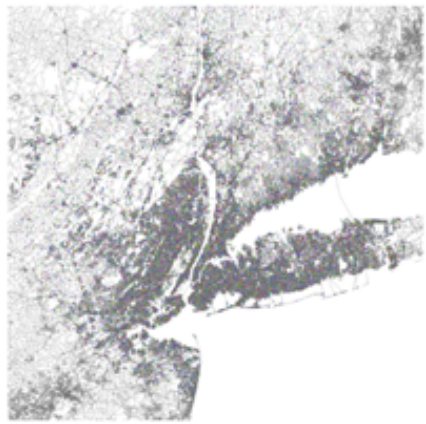


Figure 6.13: Figure illustrating assignment of entities under scatter partitioning scheme. All entities with the same color are assigned to the same processor

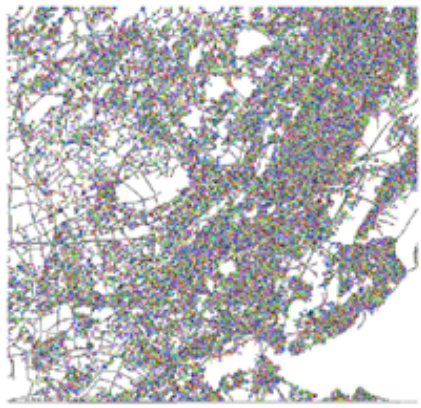


Figure 6.14: A zoomed in version of the Figure 6.13. The scatter scheme assigns close by entities to different processors

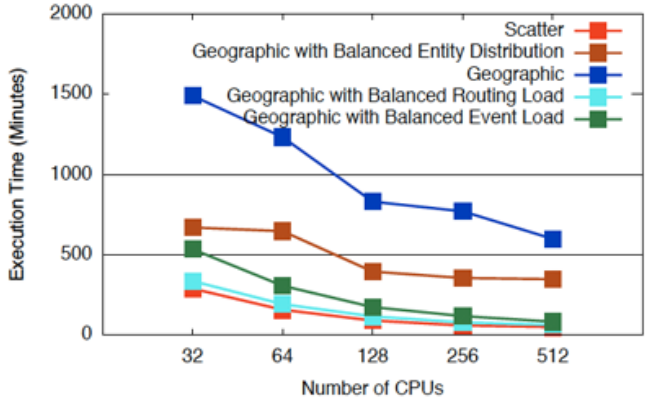


Figure 6.15: Comparison of execution times of FastTrans (as a function of #CPUs) under different partitioning schemes

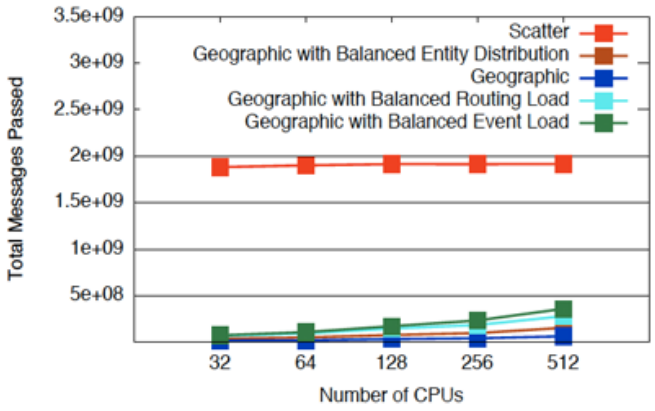


Figure 6.16: Comparison of the number of messages passed in FastTrans (as a function of #CPUs) under different partitioning schemes

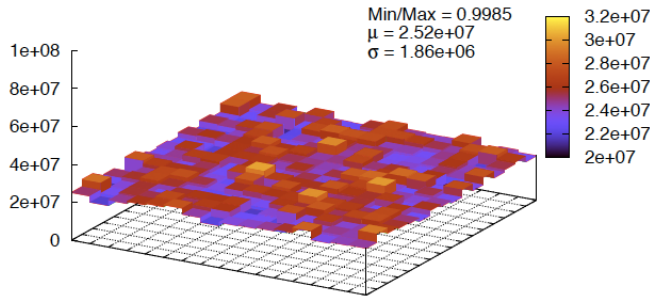


Figure 6.17: Computational load distribution of FastTrans in a 256 CPU run under scatter partitioning. Each bar represents the load on one CPU

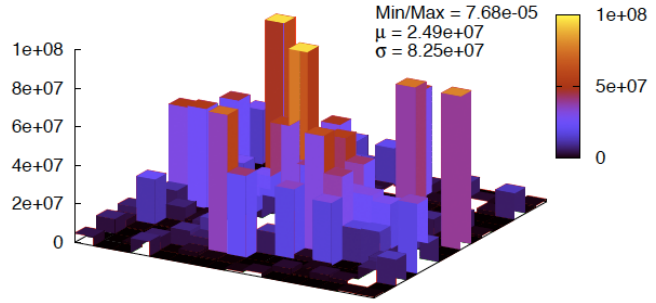


Figure 6.18: Computational load distribution in FastTrans in a 256 CPU run under pure geographic partitioning

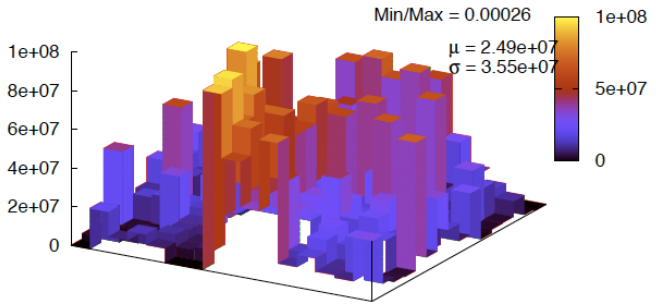


Figure 6.19: Computational load distribution in FastTrans in a 256 CPU run under geographic partitioning with balanced entities

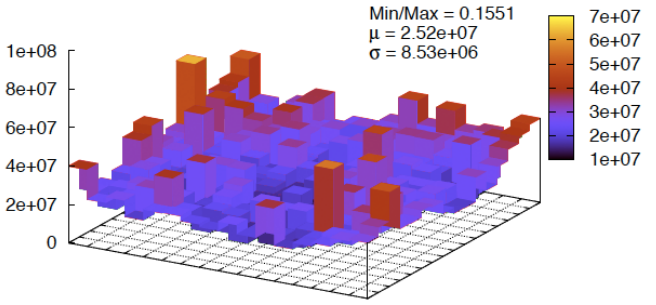


Figure 6.20: Computational load distribution in FastTrans in a 256 CPU run under geographic partitioning with balanced event load (estimated)

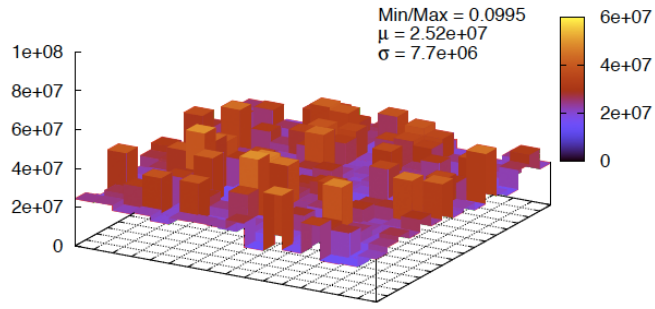


Figure 6.21: Computational load distribution in FastTrans in a 256 CPU run under geographic partitioning with balanced routing load

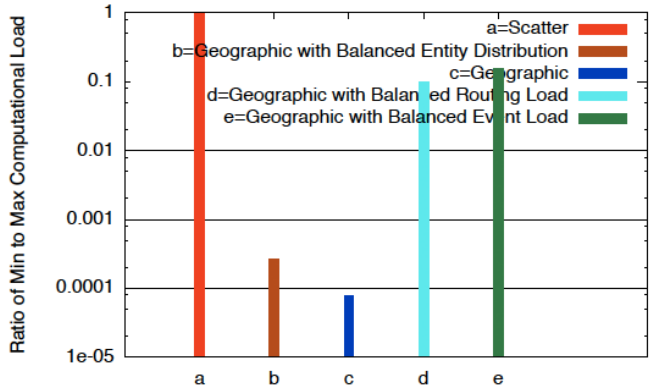


Figure 6.22: Comparison of fairness of computation load in FastTrans under different partitioning schemes

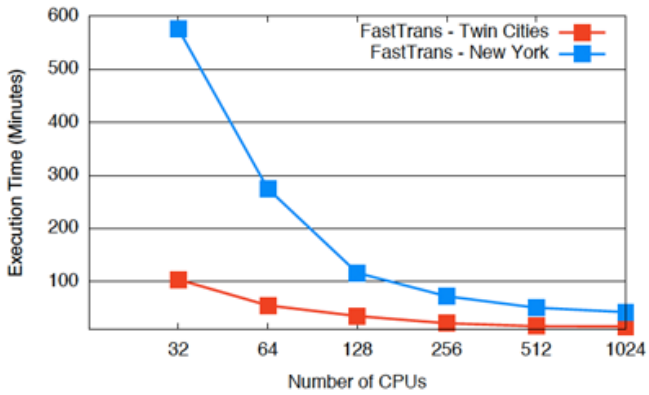


Figure 6.23: Execution time of FastTrans as a function of #CPUs

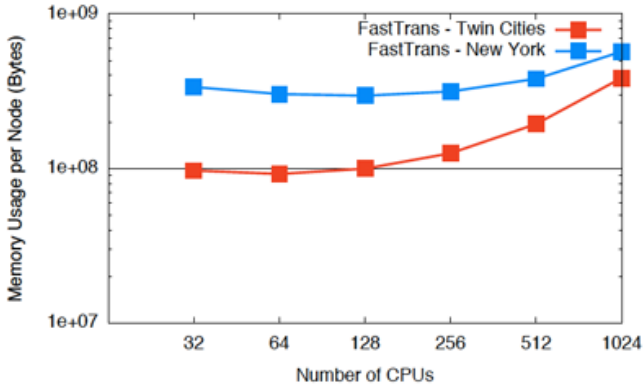


Figure 6.24: Memory usage per node in Fast- Trans as a function of #CPUs

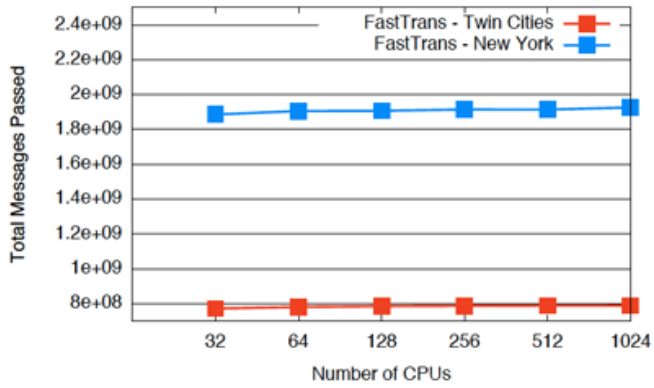


Figure 6.25: Messages passed in FastTrans as a function of #CPUs

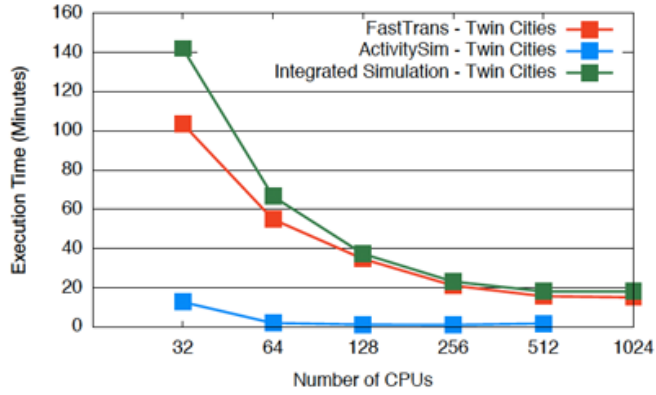


Figure 6.26: Comparison of execution times of FastTrans, ActivitySim, and the integrated simulation as a function of #CPUs

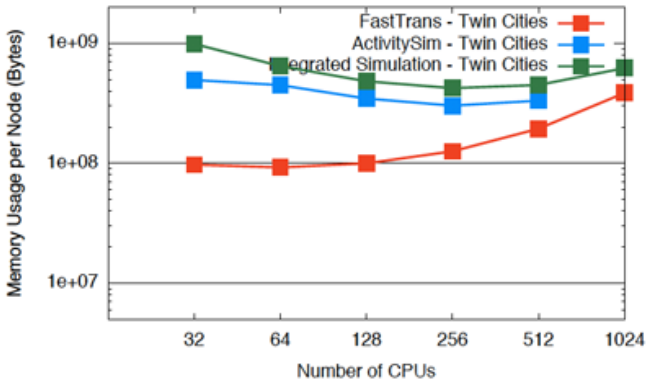


Figure 6.27: Comparison of the memory usage per node in FastTrans, ActivitySim, and the integrated simulation as a function of #CPUs

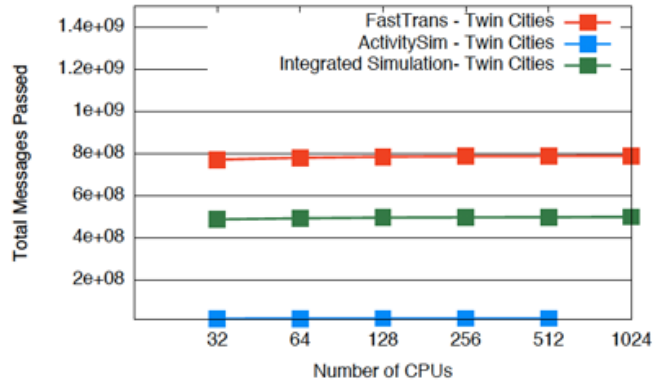


Figure 6.28: Comparison of the number of messages passed in FastTrans, ActivitySim, and the integrated simulation as a function of #CPUs

Chapter 7

Metrics

7.1 Introduction

While there are several research works on Critical Infrastructures Interdependencies modeling and analysis, at the best of our knowledge, there are very few works on interdependencies modeling and quantification. Zimmerman et al. proposed explicit metrics to quantify interdependencies [91, 92]. In [91] the author measures the direction of infrastructure failures as the ratio between the number of times one type of infrastructure caused damage to other type and the number of time other infrastructure damaged that type of infrastructure. A metric to measure the duration of cascading effects is proposed in [92], where the authors quantify the effect of main US power grid outages on other infrastructures.

In this chapter we contribute to the challenging problem of interdependencies quantification proposing a taxonomy that classifies interdependencies metrics on the basis of their information contents, and their capabilities to support decision making and risk analysis. As support for the decision makers we propose also: i) a systematical approach to compute metrics from system and/or system model observations; and ii) a set of statistical measures that can be applied to the chosen interdependencies metrics, and that help in evaluating the goodness of strategies and mechanisms acting at improve critical infrastructure protection and/or resilience.

Metrics to quantify interdependencies can be classified as: metrics that measure the macro characteristics of interdependencies and

of their impact on system behavior; and metrics that allow to go insight the infrastructure behavior allowing to quantify the strength or weakness of infrastructures or infrastructures' components. The first family of metrics support decision making at organizational or strategical level, while the second category helps decision maker at engineering or practical level.

The statistical measures we propose, as tool to measure the goodness of a choice, are: the Xth-percentile, the Cumulative Distribution Function (CDF) and the Complementary Cumulative Distribution Function (CCDF). Xth-percentile and CDF of the observed system state variable are used to evaluate the degree of satisfaction or the goodness of a choice, while the CCDF of a set of observed outcomes is used in survivability analysis of both human beings and infrastructure components.

The proposed systematical approach to compute interdependencies metrics is based on system model observation and on their functional transformation.

To validate and to show how the proposed approach works, we have decided to use an example driven approach. We have chosen the case study we used in chapter 4 to model and simulate the complex system in deep details.

The chapter is organized as in the following. Section 7.2 introduces the interdependencies metrics taxonomy and discuss in details the proposed metrics. Here we discuss also the statistical measures for interdependencies analysis and the methodology to compute the metrics. In section 7.3 we present the toy case study and in section 7.4 we discuss the interdependencies analysis results. Section 7.5 concludes the chapter.

7.2 Metrics to quantify Critical Infrastructure interdependencies

We classify metrics to quantify critical infrastructure interdependencies using three dimensions: the information content, the provided decision support and the computational cost. In the specific we identify the following categories. *Shape metrics*, which quantify macro, or "shape", characteristics of interdependencies (see figure 7.2(a)), as direction [91] or duration [92]. *Core metrics*, which measure both the causes and effects of an outages at a specific infrastructure's component (see figure 7.2(b-c)), and the the goodness of strategies/mechanisms designed to improve critical infrastructures protection and resilience. *Sector specific metrics*, which measure the infrastructure's state at global and component level.

Figure 7.1 shows how core, shape and sector specific metrics are positioned in the three dimensional space (*decision support capabilities*, *information content*, *cost*). The decision support range from engineering level (low) to strategical level (high). The Information content range from micro level (low) to macro level (hign). And the cost range from low to high. The first to dimension are qualitative, while the values for the cost depends on the specific implementation and case study.

7.2.1 Shape metrics

Let us consider the direct metric *relative duration* ($R_{i,j}$) to measure the cascading effect of an outage [92]. $R_{i,j} = \frac{T_j}{T_i}$ is defined as the ratio between the duration T_j of the infrastructure outage j that is consequence of the outage on the infrastructure i and the duration T_i of the outage on infrastructure i.

The computation of shape metrics requires to face 2 main problems: *how to measure* $R_{i,j}$ and *How to quantify the impact on infras-*

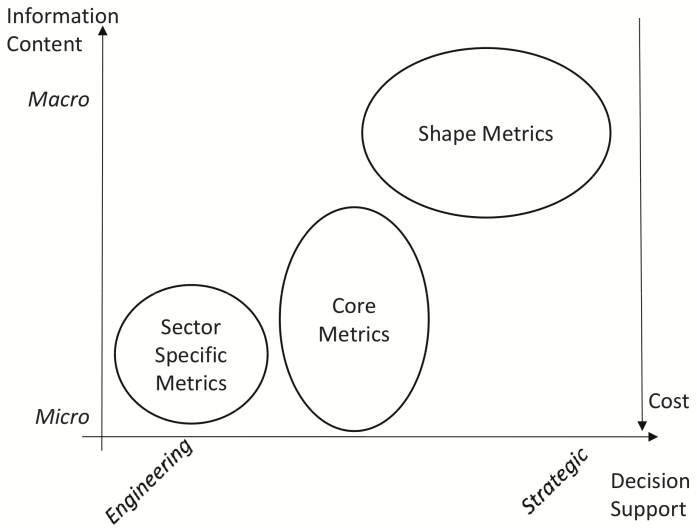


Figure 7.1: The proposed taxonomy for Interdependency quantification metrics

tructure j.

We argue that the answer is in the observation of sector specific metrics, and that in general $R_{i,j}$, is a function $f(\cdot)$ of the time t and of the set of sector specific metrics M_j used to measure the performance level or the capabilities of the infrastructure j: $R_{i,j} = f(t, m_j^1, m_j^2, ..., m_j^p)$ where $m_j^k \in M_j$.

Let us consider the example in figure 7.3. Suppose that at time t_1 there is an outage on the power grid (infrastructure i), and at time $t_1' \geq t_1$ we start observing a decrease in X, the overall throughput of the communication network (infrastructure j): $X(t) = X_0$ if $t \leq t_1$ and $X(t) \leq X_1$ if $t \geq t_1'$. X_1 is the *critical threshold* for the network performances, that is when $X < X_1$ the network lost the capabilities to correctly provide its services. If the power grid outage is repaired at time t_2 and after time t_3 we observe that the throughput is going back to X_0, $X(t) \geq X_2$ at time t_3 and $X(t) \to X_0$ for $t \geq t_3$, we

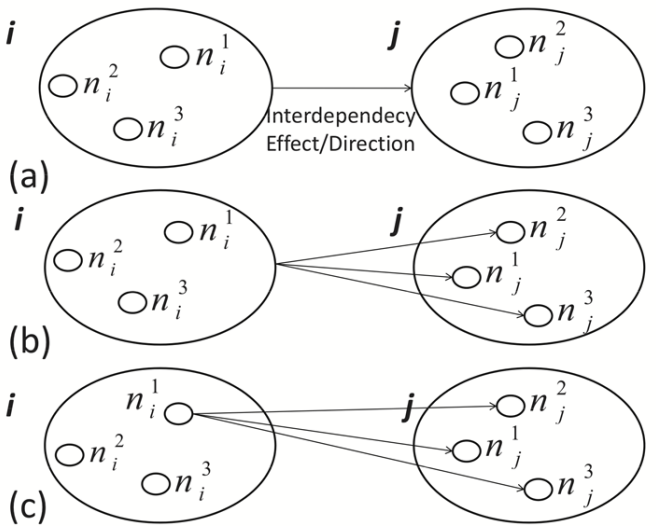


Figure 7.2: Difference between shape metrics (a) and core metrics (b-c)

assert that at time t_3 the cascading effect of the power grid outage can be considered terminated. In this example we suppose that when $X(t) \geq X_2$ the communication network is capable to correctly provide its services. Obviously $X_1 \leq X_2 \leq X_0$. $R_{i,j}$ is measured as a function $f(t, X)$ of the the time and of the throughput, $R_{g,n} = \frac{t_3 - t'_1}{t_2 - t_1}$, where t'_1 is such that $X(t) \leq X_1$ for $t \geq t'_1$ and t_3 is such that $X(t) \geq X_2$ for $t \geq t_3$.

As observed in [92], if $R_{i,j} < 1$ the infrastructure j is capable to react by themselves to the outage, for example reconfiguring its services. Otherwise if $R_{i,j} > 1$, the infrastructure j is heavily dependent on the outage and it needs an overhead time to restore their services after the outage is restored.

Another interesting question is about aggregate measures of shape metrics. For example: *how to compute the total relative duration $R_{i,I}$ of an outage on infrastructure i on a set of infrastructures I.* Let us

consider that the outage on the power grid has impact on the communication network, and transportation system and that the communication network outage has impact on the credit card transaction systems. The duration of the cascading effect terminates when all the infrastructures restore their normal working conditions. Then, in a simulation model, the *total relative duration* of an outage on infrastructure i could be measured as $R_{i,I} = \max_{j \in I, j \neq i} \{R_{i,j}\}$, where $R_{i,j}$ is a function of sector specific performance indexes of infrastructure j, as discussed before.

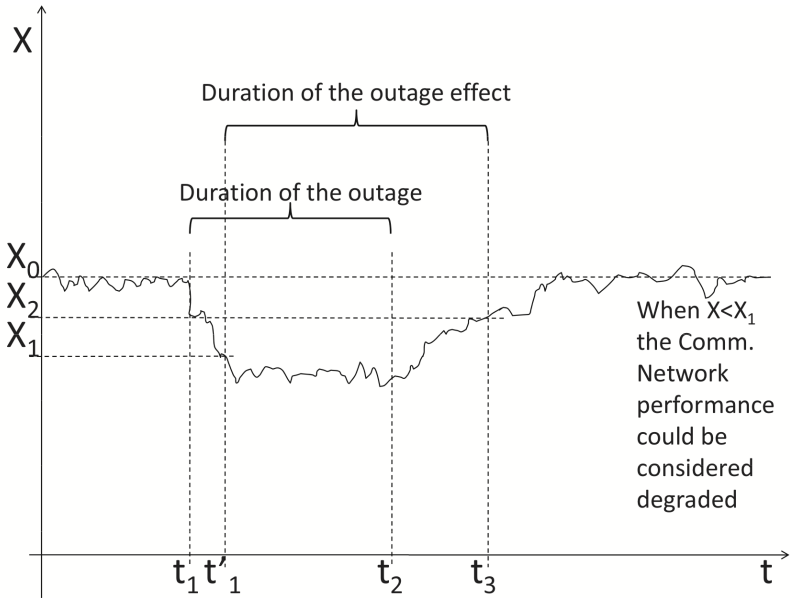


Figure 7.3: An example of relationship between sector specific metrics and direct metric.

7.2.2 Core metrics

$R_{i,j}$ of course quantifies a macro characteristic of the interdependencies between i and j, but it does not give any information about who are the infrastructure nodes involved in the outage propagation or effected by the outage propagation or what is the impact of a specific node's failure on an infrastructure or on a single infrastructure component.

Core metrics are useful to quantify the effect of interdependencies at infrastructure's node level, or more deeply at node's component level. This capability allows to go insight the causes and effects of an outage.

Two examples of core metrics can be obtained refining the shape metric $R_{i,j}$, to answer question such as: "*what is the weakest node of infrastructure j*" or "*the more important node of infrastructure i*". To identify the weakest node of infrastructure j respect to an outage on infrastructure i we define $R_{i,n_j^k} = f(t, M_j^k)$, where n_j^k is the k^{th} node of infrastructure j and M_j^k is the set of metrics used to measure the performance or capabilities of n_j^k. The weakest node is then obtained evaluating the expression $n_j^l = max_{k \in N_j} \left\{ R_{i,n_j^k} \right\}$, where N_j is the set of nodes that compose the infrastructure j.

In the same way, to identify the most important node of infrastructure i on the behavior of infrastructure j, we define $R_{n_i^h,n_j^k} = f(t, M_j^k)$, where n_i^h is the h^{th} node of infrastructure i and n_j^k and M_j^k are defined above. The most important node of i can be determined evaluating the expression $n_i^l = max_{h \in N_i} \left\{ R_{n_i^h,n_j^k} \right\}$ for each $k \in N_j$.

More in general, if we have a sector specific metric $m_j^k \in M_j^k$, the above mentioned question can be answered evaluating the new expressions: $n_j^l = max_{k \in N_j} \left\{ \Delta m_j^k \right\}$ and $n_i^l = max_{h \in N_i} \left\{ \Delta m_j^k \right\} \forall k \in N_j$, where Δm_j^k is the variation of the sector specific metric considered. Obviously, depending on the metric considered, the maximization

problem can be turned into a minimization problem.

Besides measuring the loss of performance/capability of an infrastructure, the core metrics can also be used to measure the goodness of strategies/mechanisms designed to protect or to enhance the resilience of critical infrastructures. For example, core metrics help in answering questions such as: what are the effects produced by changing a rescue plan? What are the consequences of a network re-engineering? How to quantify the effects of a new service reconfiguration strategy? But also: what is the probability to rescue the X percent of the population? What is the percentage of population died? How long do the rescue actions take ? All this questions cannot be answered using direct metrics. Other examples of core metrics are: number of nodes damaged, time to recover the node functionalities, time to reconfigure a plane/system/network, number of died/rescued humans being, and more.

7.2.3 Statistical measures for interdependencies quantification

A statistical measure typically used to describe the degree of satisfaction is the Xth-percentile of a performance index. The Xth-percentile of a dataset is defined as the value that is lager than $X\%$ of the data. To compute the Xth-percentile of a random variable is extremely useful the Cumulative Distribution Function (CDF) defined as $F_X(x) = P\{x \leq X\}$, indeed inverting F_X we obtain the desired percentile. For example the $95th$-percentile of X is $\tau = F_X^{-1}(0.95)$. Plotting the CDF is then a visual support to immediately measure different percentile of interest.

For performance indexes such as crisis resolution time, rescue time and number of failed nodes it makes sense to measure the degree of satisfaction of a new adopted countermeasure, while for the number of repaired nodes it makes more sense to compute the value of X

such that $P\{x > X\} = Y$. X could be easily computed using the Complementary Cumulative Distribution Function (CCDF). CCDF is largely used in survival analysis and is defined as $F_c(x) = 1 - F_X(x) = Pr\{x > X\}$. Inverting F_c we obtain $X = F_c^{-1}(Y)$.

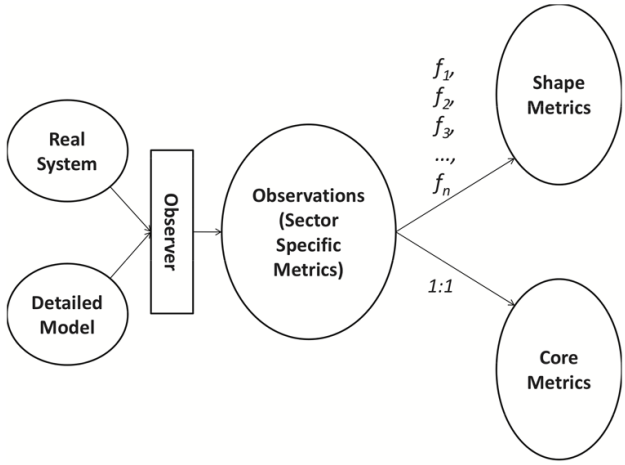


Figure 7.4: The proposed methodology to compute shape and core metrics from sector specific metrics.

7.2.4 A Methodology to compute Core and Shape metrics

As introduced before, the fundamental to compute Shape and Core metrics are sector specific metrics. Figure 7.4 shows the relationship among sector specific metrics, core metrics and shape metrics and the process to compute shape and core metrics. After the observation of a real system or of a detailed model (or both), must be identified the observed parameters that can be directly used as core metric (without any further transformation). To compute the shape metrics, is necessary to identify the useful sector specific metrics and the

most appropriate transformation function $(f_1, ..., f_n)$ as, for example, described in section 7.2.1.

7.3 The Case Study

Considering the model and implementation presented in chapter 3 and 4 as well as the scenario of chapter 4, we propose a case study allows us to simulate two different scenarios: *rescue of wounded* in case of disaster; and *propagation of outages* through infrastructures.

In the *propagation of outages* scenario we consider only the main infrastructures (power grid, transportation network and communication network) and we investigate how shape metrics allow to quantify the effect of a power grid outage on the behavior of the communication network.

In the *wounded–rescue* scenario we investigate how core metrics allow to study the evolution of a crisis in presence of different types of outages on the power grid. We suppose that in a given location, e.g. effect of a terroristic attack or other kind of disaster, there are wounded to rescue. The communication network is used by wounded, citizens, authorities, rescue crews and hospitals. Both hospitals and rescue crews use the IS4CEM to coordinate the rescue operations. The transportation network is used by the rescue crews to reach the wounded and bring them to the hospitals, as is used by the wounded capable to move by himself to the hospitals for first aid. The power grid supplies the communication network, the IS4CEM, hospitals and rescue crew stations, as the traffic lights.

7.4 Interdependencies analysis

The goal of our experiments is twofold. Through Federated ABMS we show: how sector specific metrics are used to measure shape metrics;

how core metrics are capable to quantify interdependencies and if the chosen statistical measures for performance indexes are appropriate.

Both for *rescue of injured* and *propagation of outages* scenarios we compare results from three different cases: no outages, one outage and two outages on the power grid. The node that will experiment an outage are randomly selected.

In *rescue of wounded* scenario, we suppose that the outage is never repaired, while in *propagation of outages* scenario we suppose that the time to repair the outage is constant.

In each scenario, and for each case, experiments are conducted running 50 simulations whit different seeds for random numbers generation. We consider 3 HHCs, 10 power grid nodes, 10 routers and access points, 10 rescue agents and 50 wounded agents. In the following discussion, numbers are only a way to show how the proposed methodology works.

7.4.1 Propagation of outages scenario

We suppose that at time $t = 100$ tics, one or two randomly selected power grid nodes fails. We suppose also that the network has not auxiliary power systems and then, when a power grid node fails, one or more network nodes (routers or access points) will go out of service, until they are powered up again. The time to repair the power grid node outage is fixed to 300 tics.

Figure 7.5 compares the *overall throughput* X of the communication network. $X = \sum_{i \in \mathcal{N}} X_i$, where $\mathcal{N}$ is the set of nodes in the network and X_i is the throughput of node i. As expected, if one or more routers fail, X decreases. We suppose that the critical threshold for network performances is 8000 Mbps.

In case of one outage, X degrades at $t = 100$, after three tics (see fig. 7.6, left) it is below 8000 Mbps and after 100 more tics stabilizes around 7000 Mbps. When the outage is repaired, at $t =$

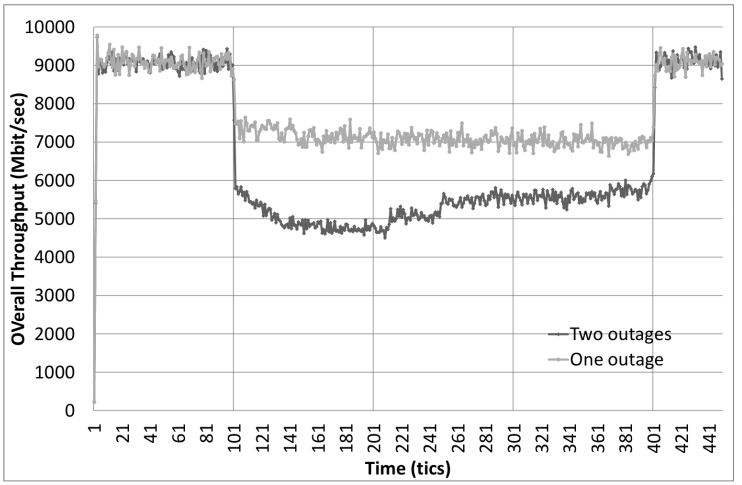
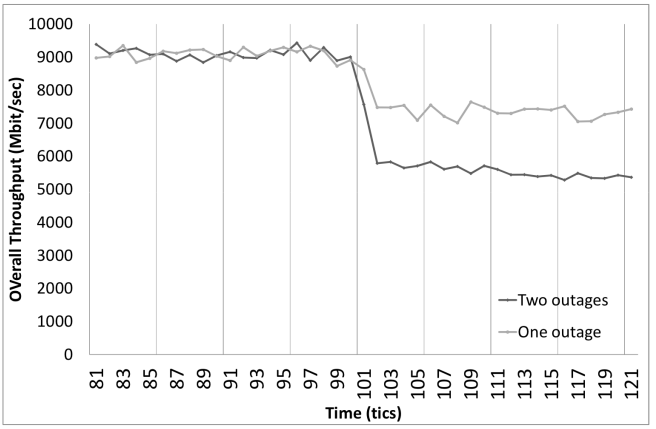


Figure 7.5: Overall throughput of the communication network

400, four tics are enough to reestablish normal working conditions (see fig. 7.6, right), then $R_{g,n} \approx 1$. The capability to rapidly restore normal working conditions is due to the adaptiveness of the routing algorithm, but also because we do not consider delays in nodes reboot and possible nodes damages due to inappropriate shutdown.

In case of two outages, the overall throughput degrade is heavier. After three tics it is below 6000 Mbps (see figure 7.6, left). After 30 more tics X is below 5000 Mbps but at $t = 214$ the reconfiguring capabilities of the routing algorithm starts working and at $t = 250$ the overall throughput definitely stabilize around 5500 Mbps (see fig. 7.6, right). Also in this case, $R_{g,n} \approx 1$, indeed few tics are enough to reestablish normal working condition after that the outages are restored (see fig. 7.6, right).

Despite the numerical results, is evident how the time plot of



sector specific metrics helps to compute the duration of an outage. When the critical threshold for the throughput is 8000Mbps $R_{g,n} \approx 1$, the duration of the outage on the communication network is about the same that the outage duration on the power grid. But if we lower the critical threshold to 5300Mbps $R_{g,n} = 0$ in case of one outage and $R_{g,n} \approx 0.35$ in case of two outages (see fig. 7.5).

7.4.2 Rescue of wounded scenario

In this scenario we suppose that, in a given location, at time $t = 0$, there are $N_w = 50$ wounded, victims of a terroristic attack. We compare the results for the three cases (no outages, one outage and two outages), using the $90th$-percentile, the CDF and the CCDF.

Figures 7.7, 7.8 and 7.9 show the cumulative distribution function of T_c, T_r and $W_d(\%)$ respectively. The experiments show that the CDF gives an immediate idea of what is the system behavior and how outages heavily increase T_c, T_r and W_d.

The values for the $90th$-percentile for T_c, T_r and W_d are shown

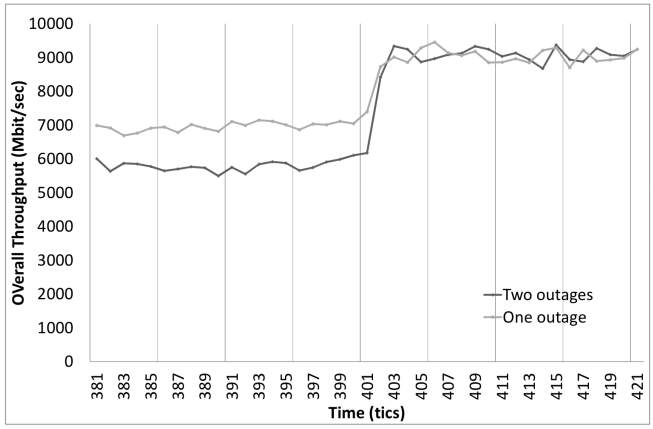


Figure 7.6: Zoom of the overall throughput at the beginning of the power grid outage, $t = 100$tics (left), and at the end, $t = 400$ (right)

in table 7.1. Using again the CDF and the concept of percentile is easy to compute the probability p_d that w percent of wounded will die and the probabilities p_r and p_c that T_r and T_c, respectively, are less then T seconds: $p_d = F_d(W)$, $p_r = F_r(T)$ and $p_c = F_c(T)$, where F_d, F_r and F_c are the CDF of W_d, T_r and T_c respectively.

Table 7.1: $90th$ percentile of T_c (sec.), T_r (sec.) and $W_d(\%)$

	Outages		
Metric	No	One	Two
T_r	165.83	187.96	269.44
T_c	350	725	725
W_d	34.66	99	100

Table 7.2: Values of W such that $P\{W_r > W\} = p$

	Outages		
$P\{W_r > W\}$	No	One	Two
0.9	64%	6%	0%
0.75	66%	22%	0%
0.5	72%	68%	6%

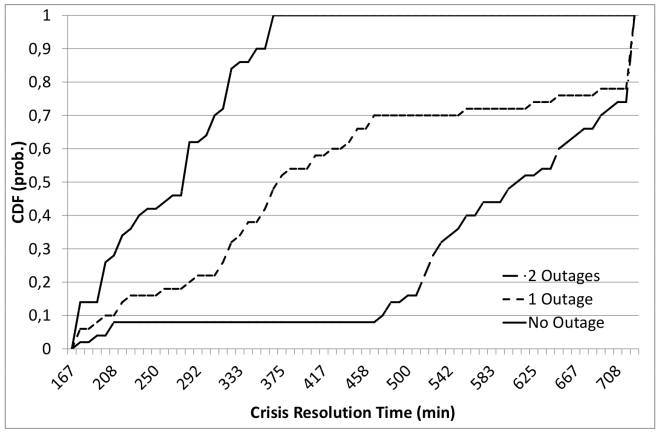


Figure 7.7: CDF of the crisis resolution time T_c

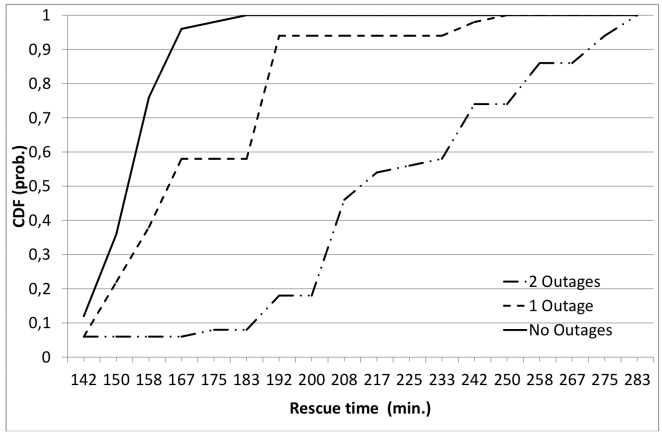


Figure 7.8: CDF of the wounded rescue time T_r

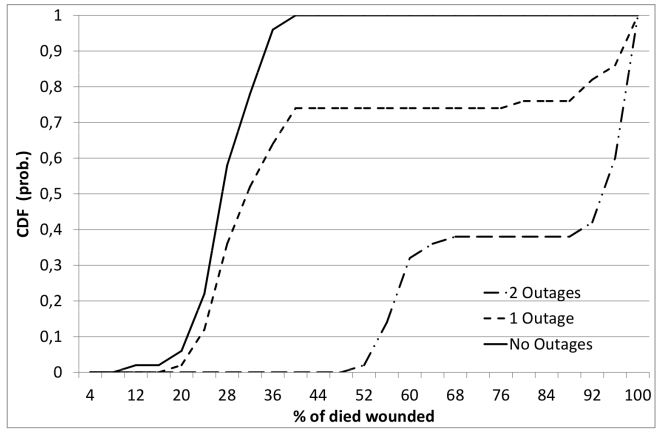


Figure 7.9: CDF of the percentage of died agents W_d

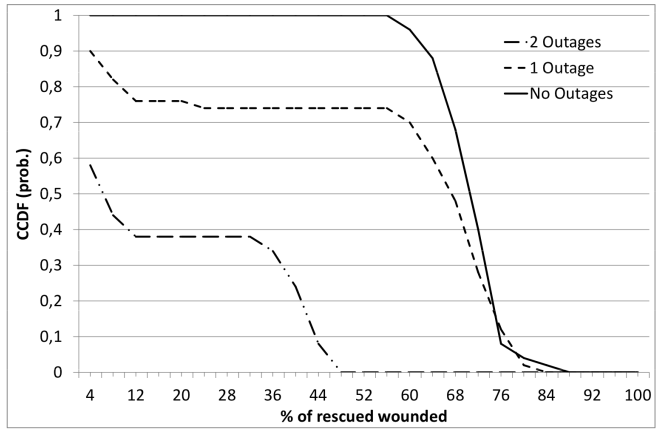


Figure 7.10: CCDF of the percentage of rescued wounded

To analyze the number of rescued wounded we use the CCDF (see figure 7.10), which gives us information about the probability to rescue more than W percent of wounded, $P\{W_r > W\}$, but also, inverting the CCDF, the value of W such that $P\{W_r > W\} = p$. Table 7.2 shows some simulation results. We fix the three different values of p (0.9, 0.75 and 0.5) and we extract the respective values for $W(\%)$

7.5 Concluding Remarks

We conclude the chapter summarizing the properties of the proposed metrics classes, comparing them on the basis of their information content, decision support capabilities and computational cost (see figure 7.11).s

Shape metrics, with their macro level information content, support decision makers at organizational and strategical level. Shape metrics can be computed in two ways: using engineering level observation, as discussed in section 7.2.1, or using higher level system observation, that can be obtained from high level (simulative or analytical) model. Obviously in the later case the computational cost is lower than the computational cost for core metrics, with all the drawback of the results obtained from a simplified model.

On the contrary, the computation of core metrics is a very expensive process. Indeed, there is the need of detailed system observation, that can be obtained instrumenting a real system observer or implementing a detailed simulation model. We remark that the higher cost of core metrics pays decision maker back with information on causes and effects of outages, due to the interdependencies among metrics, and the core metrics give a direct or indirect quantification of interdependencies.

It is important to remark that, while shape metrics could have

<table>
<tr><th>Metrics</th><th>Information Content</th><th>Decision Support</th><th>Cost</th></tr>
<tr><td>Shape Metrics</td><td>Macro Level</td><td>Support decision makers at organizational and strategical level</td><td>Low/Medium
A detailed models is not mandatory</td></tr>
<tr><td>Core Metrics</td><td>Micro Level</td><td rowspan="3">Support decision makers at engineering and practical level.

Quantify the causes and effects of outages.</td><td rowspan="3">Medium/High
A detailed model is mandatory.
Typically a simulation model.</td></tr>
<tr><td rowspan="2">Sector Specific Metrics</td><td>Engineering level</td></tr>
<tr><td>Input for Shape and Core metrics computation</td></tr>
</table>

Figure 7.11: Summary of the interdependencies metrics characteristics

an universal meaning, because they characterize the shape of an interdependency, the core metrics cannot be universal, because they characterize the cause-effect characteristics of an interdependency.

Chapter 8
Conclusions

In this thesis we have adopted the simulation approach as a tool for studying critical infrastructure interdependency, analysis and protection.

In particular, we have used two relatively new approaches: the FedABM&S is based on Agent Based Modeling and distributed simulation, while the second one uses the micro and parallel simulation approach. FedABM&S has been used to simulate critical scenarios where communication network and power grid infrastructures are involved. The micro-simulation approach has been used to simulate realistic scenario based on statistically accurate population of the US and on real data of transportation network in New York and Twin Cities (MN).

FedABM&S has the goal to reduce implementation efforts and costs reusing already implemented sectors simulators, and putting in the background scalability and performance. Conversely, micro parallel simulation puts the performance and scalability in foreground, increasing drastically costs and efforts. In terms of abstract abilities, FedABM&S allows to give a higher level of details and cover all levels of interdependency (physical, cyber, logical and geographical). Vice versa, micro-simulation adopts a bottom-up approach that does not allow easily to define in advance all interdependencies, but it could be much powerful to find hidden dependencies.

FedABM&S has been widely used in the project CRESCO at the ENEA center of research in Italy, and it has been cited in different international papers and journals. ActivitySim and FastTrans simu-

lators are widely used at the Los Alamos National Laboratory, USA in particular for the MIITS project that models large-scale infrastructure networks.

Thanks to these tools, we have been able to use some of our suggested metrics, contributing to the challenging problem of interdependencies quantification. We proposed a taxonomy that classifies interdependencies metrics on the basis of their information contents, and their capabilities to support decision making and risk analysis.

Finally, we can say that the micro parallel simulation seems to be a bigger challenge compared to the Federated ABMS but, in case of success, it could be much more powerful in terms of performance and scalability.

Bibliography

[1] http://computing.lanl.gov/article/505. 117

[2] http://computing.lanl.gov/article/570. 117

[3] The computational research center for complex systems (CRESCO) project. http://www.cresco.enea.it/. 62

[4] Cyber war in estonia. http://news.cnet.com/8301-17938_105-9721429-1.html. 2

[5] e–Agora. http://www.aia.es/internet/website_eng/Agora.html. 61

[6] The e–infrastructure project. http://cordis.europa.eu/fp7/ict/e-infrastructure/home_en.html. 31

[7] Italian national agency for new technologies, energy and sustainable economic development (ENEA). http://www.enea.it/com/ingl/default.htm. 62

[8] OMNeT++. http://www.omnetpp.org. 47, 55, 58

[9] The portico project. http://porticoproject.org. 58

[10] PRIME. Parallel Real-time Immersive Modeling Environment (PRIME). http://lynx.cis.fiu.edu:8000/twiki/bin/view/Public/PRIMEProject. 82, 105

[11] Repast. http://repast.sourceforge.net. 49, 55, 58

[12] USA patriot act (h.r. 3162). http://epic.org/privacy/terrorism/hr3162.html. 2

[13] MPI the message passing interface. http://www.mcs.anl.gov/research/projects/mpi, 2000. 105, 118

[14] CONCEPTS, I. T. VISSIM simulation tool. http://www.itc-world.com/VISSIMinfo.htm, 2001. 103

[15] US Department of Transportation (DOT), 2003. Bureau of Transportation Statistics. NHTS 2001 Highlights report BTS03-05. 80, 97

[16] Communication from the commission on a european programme for critical infrastructure protection, 2006. 3

[17] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. In *Proceeding of the National Academy of Sciences*, 2000. 20

[18] Usov Andrij and Beyel Cèsaire. Simulating interdependent critical infrastructures with SimCIP, 2008. xi, 31

[19] R. Baldick, B. Chowdhury, I. Dobson, Zhaoyang Dong, Bei Gou, D. Hawkins, H. Huang, M. Joung, D. Kirschen, Fangxing Li, Juan Li, Zuyi Li, Chen-Ching Liu, L. Mili, S. Miller, R. Podmore, K. Schneider, Kai Sun, D. Wang, Zhigang Wu, Pei Zhang, Wenjie Zhang, and Xiaoping Zhang. Initial review of methods for cascading failure analysis in electric power transmission systems IEEE PES CAMS task force on understanding, prediction, mitigation and restoration of cascading failures. pages 1 –8, July 2008. 15

[20] R. J. Beckman, C. L. Barrett, K. B. Berkbigler, K. R. Burris, B. W. Bush, S. D. Hull, J. M. Hurford, P. Medvick, D. A. Kubicek, M. Marathe, J. D. Morgeson, K. Nagel, D. J. Roberts,

L. L. Smith, M. J. Stein, P. E. Stretz, S. J. Sydoriak, K. Cervenka, and R. Donnelly. Transportation analysis simulation system (TRANSIMS) - the dallas-ft. worth case study. In *Proceedings of the Fifth National Conference on Transportation Planning Methods*, 1997. 103

[21] R. Bloomfield, N. Chozos, and P. Nobles. Infrastructure interdependency analysis: Introductory research review, feasibility study on interdependency analysis, report no. 1. Technical report, City University London, 2009. 15

[22] Robin Bloomfield, Peter Popov, Kizito Salako, David Wright, Lubos Buzna, Ester Ciancamerla, Saverio Di Blasi, Michele Minichino, and Vittorio Rosato. Analysis of critical infrastructure dependence - an IRRIIS perspective. In *Critical Infrastructure Protection*, 2008. xi, 18

[23] Transportation Research Board. Highway capacity manual, 2000. 104

[24] Nino Boccara. *Modeling Complex Systems*. Springer, 2003. 38

[25] Csaba Attila Boer, Arie de Bruin, and Alexander Verbraeck. Distributed simulation in industry – a survey: part 1 – the COTS vendors. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1053–1060. Winter Simulation Conference, 2006. 54

[26] Csaba Attila Boer, Arie de Bruin, and Alexander Verbraeck. Distributed simulation in industry – a survey: part 2 – experts on distributed simulation. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1061–1068. Winter Simulation Conference, 2006. 54

[27] Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(3):7280–7287, May 2002. 39

[28] Luciano Bononi, Michele Bracuto, Gabriele D'Angelo, and Lorenzo Donatiello. A new adaptive middleware for parallel and distributed simulation of dynamically interacting systems. In *DS–RT '04: Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS–RT'04)*, pages 178–187, Washington, DC, USA, 2004. IEEE Computer Society. 54

[29] Andrei Borshchev, Yuri Karpov, and Vladimir Kharitonov. Distributed simulation of hybrid systems with AnyLogic and HLA. *Future Gener. Comput. Syst.*, 18(6):829–839, 2002. xi, 19, 22, 35

[30] Gordon Cameron, Brian J. N. Wylie, and David McArthur. Paramics: moving vehicles on the connection machine. In *Supercomputing '94: Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 291–300, New York, NY, USA, 1994. ACM. 103, 104

[31] Valeria Cardellini, Emiliano Casalicchio, and Emanuele Galli. Agent-based modeling of interdependencies in critical infrastructures through uml. In *SpringSim '07: Proceedings of the 2007 spring simulation multiconference*, pages 119–126, San Diego, CA, USA, 2007. Society for Computer Simulation International. 57

[32] Nurhan Cetin, Adrian Burri, and Kai Nagel. Parallel queue model approach to traffic microsimulations. In *In Proceedings of Swiss Transportation Research Conference*, 2002. 103

[33] D. CHARYPAR, K. AXHAUSEN, and K. NAGEL. An event-driven queue-based microsimulation of traffic flow, 2006. 103, 104

[34] Franco Cicirelli, Angelo Furfaro, Andrea Giordano, and Libero Nigro. Distributed simulation of repast models over hla/actors. *Distributed Simulation and Real-Time Applications, IEEE International Symposium on*, 0:184–191, 2009. 60

[35] Judith S. Dahmann, Richard M. Fujimoto, and Richard M. Weatherly. The department of defense high level architecture. In *WSC '97: Proceedings of the 29th conference on Winter simulation*, pages 142–149, Washington, DC, USA, 1997. IEEE Computer Society. 58, 59

[36] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 111

[37] N. Eissfeldt, Krajzewicz D., K. Nagel, and P. Wagner. Simulating traffic flow with queues, 2001. 103

[38] Irene Eusgeld, David Henzi, and Wolfgang Kröger. Comparative evaluation of modeling and simulation techniques for interdependent critical infrastructures. Technical report, Laboratorium für Sicherheitsanalytik, Zurich, Switzerland, 2000. xi, 19, 28, 29

[39] Richard M. Fujimoto. *Parallel and Distribution Simulation Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1999. 60, 101

[40] Daniele Gianni. Bringing discrete event simulation concepts into multi-agent systems. In *UKSIM 08: Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, pages 186–191, Washington, DC, USA, 2008. IEEE Computer Society. 49

[41] Jack Goldstone A. *Using Quantitative and Qualitative Models to Forecast Instability*. United States Institute of Peace, 2008. 16

[42] O. Gursesli and A.A. Desrochers. Modeling infrastructure interdependencies using Petri nets. volume 2, pages 1506 – 1512 vol.2, oct. 2003. xvii, 24

[43] Y.Y. Haimes. Hierarchical holographic modelling. *IEEE Trans. System, Man, and Cybernetics*, 11(9):606–617, 1981. 20

[44] Y.Y. Haimes, H.H. Lambert, and S. Kaplan. Risk filtering, ranking, and management using hierarchical holographic modelling framework, tech report 22-2000. Technical report, Risk Management of Eng. Systems, Univ. of Virginia, 2000. 20

[45] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, February 2007. 111

[46] Francis Heylighen and Cliff Joslyn. Cybernetics and second-order cybernetics. In Academic Press, editor, *Encyclopedia of Physical Science and Technology*. New York, 2001. 38

[47] R.C. Hofer and M.L. Loper. DIS today [distributed interactive simulation]. In *Proceedings of the IEEE*, volume 83, pages 1124–1137, August 1995. 54

[48] R. Jacob, M. Marathe, and K. Nagel. A computational study of routing algorithms for realistic transportation networks. *J. Exp. Algorithmics*, 4:6, 1999. 111

[49] Pu Jiang and Yacov Y. Haimes. Risk management for leontief-based interdependent systems. volume 24, pages 1215–1229, University of Virginia, Charlottesville, VA, USA, 2001. 17

[50] A. Krings and P. Oman. A simple GSPN for modelling common mode failures in critical infrastructures. page 10 pp., jan. 2003. 23

[51] Lucas Kroc, Stephan Eidenbenz, and Va Ramaswamy. Simcore. Technical Report 07-0590, Los Alamos National Laboratory, 2007. Unclassified Report. 81, 82

[52] Lucas Kroc, Stephna Eidenbenz, and Va Ramaswamy. Sessionsim. Technical Report 07-0592, Los Alamos National Laboratory, 2007. Unclassified Report. 81

[53] Frederick Kuhl, Richard Weatherly, and Judith Dahmann. *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. 54

[54] Brasca C. L. and Ciapessoni E. Definition of a methodology for the assessment of mutual interdependencies between ICT and electricity generation transmission infrastructures. Technical report, ENEA, 2008. 17

[55] Jie Liu, Xiaojun Liu, Tak kuen J. Koo, Bruno Sinopoli, Shankar Sastry, and Edward A. Lee. A hierarchical hybrid system model and its simulation. In *In 38th IEEE conference on Decision and Control*, pages 3508–3513, 1999. 19

[56] Schläpfer M., Kessler T., and Kröger W. Reliability analysis of electric power systems using an object-oriented hybrid modeling approach. In *16th Power Systems Computation Conference*, Glasgow, 2008. 21

[57] José Martıi, Carlos E. Ventura, Jorge A. Hollman, K.D. Srivastava, and Hugòn Juàrez. I2Sim modelling and simulation

framework for scenario development, training, and real-time decision support of multiple interdependent critical infrastructures during large emergencies. Technical report, The University of British Columbia, 2008. xi, 34, 35

[58] Vincenzo Masucci, Francesco Adinolfi, Paolo Servillo, Giovanni Dipoppa, and Alberto Tofani. Ontology-based critical infrastructure modeling and simulation. *Critical Infrastructure Protection III*, pages 229 – 242, 2009. 32

[59] D.C. Miller and J.A. Thorpe. SIMNET: the advent of simulator networking. In *Proceedings of the IEEE*, volume 83, pages 1114–1123, August 1995. 54

[60] R. Minson and G. K. Theodoropoulos. Distributing repast agent-based simulations with hla. *Concurr. Comput. : Pract. Exper.*, 20(10):1225–1256, 2008. 60

[61] John Moteff and Paul Parfomak. Critical infrastructure and key assets: Definition and identification. Technical report, CRS Report for Congress, 2004. 3

[62] Pa Muller. *The Design of Intelligent Agents: A Layered Approach*. Springer, 1996. 83

[63] Adam Nabil. Workshop on future directions in critical infrastructure modeling and simulation, final report, January 2010. 15

[64] Michael J. North and Charles M. Macal. *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*. Oxford University Press, Inc., New York, NY, USA, 2007. 39

[65] Evan W.. Patton, D. Gray, and Michael J. Schoelles. SANLab–CM – the stochastic activity network laboratory for cognitive modeling. Technical report, Rensselaer Polytechnic Institute, 2003. 16

[66] David W. Hilt P.E. August 14, 2003, northeast blackout impacts and actions and the energy policy act of 2005. Technical report, North American Electric Reliability Council. 2

[67] Paul Pederson, Danile Dudenhoeffer, Stalin Hartley, and Mark Permann. Critical infrastructure and interdependency modeling: A survey of US and international research. In *Idaho National Laboratory*, 2006. 25, 27

[68] Kalyan S. Perumalla. A systems approach to scalable transportation network modeling. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 1500–1507. Winter Simulation Conference, 2006. 103, 104

[69] K. Peters, L. Buzna, and D. Helbing. Modelling of cascading effects and efficient response to disaster spreading in complex networks. *International Journal of Critical Infrastructures*, 4:46–62(17), 5 December 2007. 17

[70] Panos D. Prevedouros, Ph. D, and Yuhao Wang. Simulation of a large freeway/arterial network with CORSIM, INTEGRATION and WATSim. *Transportation Research Record: Journal of the Transportation Research Board*, 1999. 103

[71] Venkatesh Ramaswamy, Sunil Thulasidasan, Phil Romero, Stephan Eidenbenz, and Leticia Cuellar. Simulating the national telephone network: A socio-technical approach to assessing infrastructure criticality. *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pages 1–7, Oct. 2007. 80

[72] Tom Rigole and Geert Deconinck. A survey on modeling and simulation of interdependent critical infrastructures. pages 1 –9, April 2006. 17

[73] George F. Riley, Mostafa H. Ammar, Richard M. Fujimoto, Alfred Park, Kalyan Perumalla, and Donghua Xu. A federated approach to distributed network simulation. *ACM Trans. Model. Comput. Simul.*, 14(2):116–148, 2004. 54

[74] S.M. Rinaldi. Modeling and simulating critical infrastructures and their interdependencies. page 8 pp., jan. 2004. 4

[75] S.M. Rinaldi, J.P. Peerenboom, and T.K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *Control Systems Magazine, IEEE*, 21(6):11 –25, Dec 2001. 43

[76] Luis Rocha. Complex systems modeling: Using metaphors from nature in simulation and scientific models. In Los Alamos National Laboratory, editor, *BITS: Computer and Communications News*. Computing, Information, and Communications Division, November 1999. 38

[77] Erich Rome, Sandro Bologna, Erol Gelenbe, Eric Luiijf, and Vincenzo Masucci. Design of an interoperable European federated simulation network for critical infrastructures (DIESIS). 2009. xi, 32

[78] K. Schneider, Chen-Ching Liu, and J.P. Paul. Assessment of interactions between power and telecommunications infrastructures. *Power Systems, IEEE Transactions on*, 21(3):1123 –1130, Aug. 2006. 23

[79] R. Sedgewick and J. S. Vitter. Shortest paths in euclidean graphs. In *SFCS '84: Proceedings of the 25th Annual Sympo-*

sium onFoundations of Computer Science, 1984, pages 417–424, Washington, DC, USA, 1984. IEEE Computer Society. 111

[80] Herbert A. Simon. How complex are complex systems. http://www.jstor.org/stable/192399. 38

[81] Ingve Simonsen, Lubos Buzna, Karsten Peters, Stefan Bornholdt, and Dirk Helbing. Dynamic effects increasing network vulnerability to cascading failures. 2007. 17, 18, 21

[82] Jerome Singer. Cited by (Rocha). http://informatics.indiana.edu/rocha/complex/csm.html. 38

[83] Aaron Sloman and Riccardo Poli. SIM AGENT: A toolkit for exploring agent designs. In *Intelligent Agents Vol II (ATAL–95). Springer-Verlag. 392–407*, pages 392–407. Springer-Verlag, 1996. 49

[84] Ricard V. Sole, Marti Rosas-Casals, Bernat Corominas-Murtra, and Sergi Valverde. Robustness of the European power grids under intentional attack, 2007. 20

[85] Peter Stadler. Advances in complex systems, 1999. 38

[86] John D. Sterman. System dynamics modeling: Tools for learning in a complex world. Technical report, MIT System Dynamics Group, 2002. 18

[87] Lewis Ted. *Critical Infrastructure Protection in Homeland Security.* Springer, 1996. 3

[88] Vhang-Hyeon Joh, Theo A. Arentze and Harry J.P. Timmermans. Understanding activity scheduling and rescheduling behaviour: Theory and numerical illustration. *GeoJournal*, 53(4):359–371, April 2001. 87, 91

[89] Ra Waupotitsch, Stephan Eidenbenz, Pa Smith, and Lucas Kroc. Multi-scale integrated information an telecommunications system (miits): First results from a large-scale end-to-end network simulator. In *Proceedings of the Winter Simulation Conference*, http://portal.acm.org/citation.cfm?id=1218112.1218500, 2006. 81, 82

[90] Clay Wilson. Computer attack and cyberterrorism: Vulnerabilities and policy issues for congress. Technical report, CRS Report for Congress, 2005. 2

[91] R. Zimmerman. Decision-making and the vulnerability of interdependent critical infrastructure. *IEEE International Conference on Systems, Man and Cybernetics*, 5:4059–4063, October 2004. 133, 135

[92] R. Zimmerman and C. E. Restrepo. The next step: quantifying infrastructure interdependencies to improve security. *International Journal of Critical Infrastructures (IJCIS)*, 2006. 133, 135, 137