Metrics to Evaluate Network Robustness in Telecommunication Networks

Marc Manzano Castro

Supervisor: David Harle

University of Strathclyde

May 25, 2011

Abstract

Society depends now more strongly than ever on large-scale networks. Several sizable failures have been experienced in the last years. Thus, it has become of vital importance to define robustness metrics. Classical robustness analysis has been focused on the evaluation of topological characteristics. In this project we extend this analysis introducing two new robustness metrics that include traffic service requirements: *QuaNtitative Robustness Metric* (QNRM) and *QuaLitative Robustness Metric* (QLRM). In addition, a review of some well known graph robustness metrics is provided. In order to compare these metrics with the proposals, a set of six topologies (random, small-world and scale-free) is evaluated in the Case Study. Then, it is shown that our two new metrics are able to evaluate the performance of a network under a given kind of impairment. Finally, some real tele-communication networks are analyzed.

Contents

1 Introduction **5**
- 1.1 Context and Motivation . . . 5
- 1.2 Objectives . . . 6
- 1.3 Aims . . . 7
- 1.4 Contents . . . 7

2 Theory and Concepts of Simulation **9**
- 2.1 Basic concepts . . . 10
- 2.2 Review of some network simulators . . . 12
  - 2.2.1 OPNET . . . 13
  - 2.2.2 ns-2 . . . 13
  - 2.2.3 ns-3 . . . 15
  - 2.2.4 OMNet++ . . . 16
  - 2.2.5 GTNetS . . . 17
  - 2.2.6 Shawn . . . 19
- 2.3 Path-Oriented Network Simulator . . . 20

3 Network Impairments **22**
- 3.1 Background . . . 22
- 3.2 Proposed taxonomy . . . 23
  - 3.2.1 Static . . . 23
  - 3.2.2 Dynamic . . . 24

4 Metrics of Robustness **27**
- 4.1 Background . . . 27
- 4.2 Topology characteristics . . . 27
  - 4.2.1 Average nodal degree ($\overline{k}$) . . . 27
  - 4.2.2 Node connectivity . . . 28
  - 4.2.3 Heterogeneity . . . 28
  - 4.2.4 Symmetry ratio . . . 28
  - 4.2.5 Diameter . . . 28
  - 4.2.6 Average shortest path length . . . 29
  - 4.2.7 Largest eigenvalue (λ) . . . 29

4.2.8 Second smallest Laplacian eigenvalue (λ_2) 29
4.2.9 Average neighbor connectivity 29
4.2.10 Assortativity coefficient (r) 29
4.2.11 Average two-terminal reliability (A2TR) 30
4.3 Quantitative and Qualitative Robustness Metrics 30
4.3.1 QuaNtitative Robustness Metric 30
4.3.2 QuaLitative Robustness Metric 31

5 Case Study **32**
5.1 Topologies 32
5.2 Simulation Scenario 33
5.3 Results 34

6 Robustness Analysis of Real Networks **42**
6.1 Topologies 42
6.2 Simulation Scenario 43
6.3 Results 47

7 Conclusions **55**

8 Further Work **57**

Bibliography **60**

A How to: *igraph* as an R package **65**
A.1 Introduction 65
A.2 How to install *igraph* as an R package 66
A.3 How to use igraph R package 66
A.3.1 Generating graphs 67
A.3.2 Working with graphs 68
A.4 Plotting graphs 69

List of Figures

2.1 Illustration of Event, Activity and Process 12

3.1 Examples of a SR impairment. (a) and (b) show that nodes are chosen randomly. 23
3.2 Example of a ST impairment. The element of discrimination is the *nodal degree*. 25
3.3 Example of a ST impairment. The element of discrimination is the *between-ness centrality*. 25
3.4 Example of a ST impairment. The element of discrimination is, in this case, to disconnect the network. 25
3.5 Example of a DE impairment. A failure occurs on a node, and after a period of time, it spreads to its neighbours. 25

5.1 er400d3 (left) and er400d6 (right) 33
5.2 sw400d10 (left) and sw400d20 (right) 33
5.3 sf400d2 (left) and sf400d4 (right) 34
5.4 Average Two-Terminal Reliability of the Case Study topologies 36

6.1 cogentco 43
6.2 deltacom 44
6.3 ion 44
6.4 kdl 45
6.5 uscarrier 46
6.6 Average Two-Terminal Reliability of the set of real network topologies 47

A.1 Adding *igraph R package* (1). 66
A.2 Adding *igraph R package* (2). 67
A.3 Plotting a graph with the layout of *layout.svd*. 70
A.4 Plotting a graph with the layout of *layout.fruchterman.reingold*. 71

List of Tables

5.1 Characteristics of the Case Study network topologies 38
5.2 Ranking of robustness of the Case Study network topologies, based on topology features 39
5.3 QuaNtitative Robustness Metric Results of the Case Study network topologies 40
5.4 QuaLitative Robustness Metric Results of the Case Study network topologies 41

6.1 Characteristics of the set of real tele-communication network topologies 51
6.2 Ranking of robustness of the set of real tele-communication network topologies, based on topological features 52
6.3 QuaNtitative Robustness Metric results of the set of real tele-communication network topologies 53
6.4 QuaLitative Robustness Metric results of the set of real tele-communication network topologies 54

Chapter 1

Introduction

1.1 Context and Motivation

Large-scale networks supporting the provision of tele-communication, electrical power, rail and fuel distribution services underpin and fulfill keys aspects of modern day living; often their ubiquity is taken for granted. These critical infrastructure networks essentially consist of *nodes* (railway stations, transformers, switches, etc.), *links* (tracks, pipes, cables, etc.) and *dynamic processes* that run over them (trains, oil or gas, electrical power, connections, etc.). In this project, in the Case Study, three different kind of topologies are considered, all of them related to *complex networks* (networks with non trivial topological features): *random*, *small-world* and *scale-free*. Random networks are a primitive and crude representation of such complex networks whereby nodes are randomly connected such that the variance in nodal degree is relatively small. In small-world networks, although the majority of nodes have a limited number of direct neighbours, most can be reached via only a small number of hops. In scale-free networks, the topology is such that some vertices, known as hubs, have a degree that is orders of magnitude larger than the average. Later on, some real tele-communication networks are also evaluated.

Recently, several sizable network failures have been experienced, reenforcing the need to take the possibility of such large and potentially catastrophic failures into consideration in the underlying network design. In 1996, the US General Accounting Office estimated 250,000 annual attacks on Department of Defense networks [1]. In 2003, a series of cascading failures were observed, resulting in a blackout in the Northeastern states [2]. The same year, Italy and Switzerland experienced cascading power system failures resulting in a blackout, which left 56 million inhabitants without power for nine hours [3]. Moreover, in 2004, Sassar virus disruptions accounted for the halt on maritime operations in the UK, the halt on railway operations in Australia, and interruptions in hospital facilities in Hong Kong [4]. The

largest and most widespread power outage in history happened across Java and Bali in 2005, affecting some 100 million people, again as consequence of cascade failures [5]. The 2006 earthquake in Taiwan disrupted undersea fibre optic lines and, as a result, banks from South Korea to Australia suffered significant interruptions [6]. In 2009, a major failure in the power supply network of Brazil and Paraguay, left around 87 million residents without power for almost 5 hours [7]. Finally, in 2010 a heavy snowfall in Spain caused a fault in a high tension power cable left 220,000 people in and around the Catalonian city of Girona without electricity. [8]. It is clear that sizable proportions of the world's population could be seriously damaged if a large network experiences significant failures. It is therefore crucial to be able to quantify the network robustness in a reliable manner while taking into account the dynamic processes supported by such networks. In communication networks, a dynamic process refers to a service (*connection*) provided by the network.

Because the underlying networks impact directly on the provisioning, performance and management of any given service, engineers are confronted with fundamental questions such as "how to evaluate the robustness of networks for a given service?" or "how to design a robust network appropriate to the needs of supported services?".

A well known definition for robustness is:

"A network is robust if disconnecting components is difficult."

In this work we assume the definition of robustness given in [9]:

"Is the ability of a network to maintain its total throughput under node and link removal."

The former definition comes from the *classical approach* where basic concepts from graph theory are used. The latter comes from a more *contemporary approach* that considers services running over the network in order to evaluate its robustness.

Between the classical and the contemporary, a wide range of approaches have analyzed the robustness of a network. These have evolved from the earlier approaches that focus mainly on the connectivity of a graph to more recent concepts that consider the spectrum of a graph. Generally, the metrics to compute the robustness of a network, based on graph topological features do not take into account the functioning of a service. Thus, we define two new metrics that do consider, under defined impairments or multiple failures, the impact upon individual services.

1.2 Objectives

The objectives of this project are listed below:

1. Design and program a discrete event based simulator. The simulator assumes the forwarding to be *path-oriented* and not *hop-by-hop*. Therefore it has been called *Path-Oriented Network Simulator* (PONS). PONS has three inputs: a network topology, a traffic matrix (associated with the topology) and the kind of impairment (or multiple failure) that will be caused during the simulation. *Java* has been chosen as the programming language.

2. Analysis and review of several well known graph robustness metrics.

3. Define two new metrics of robustness: QuaNtitative Robustness Metric (QNRM) and QuaLitative Robustness Metric (QLRM).

4. Provide a Case study for the use of such metrics based upon a range of topology types (of complex networks).

5. Submit an article to the 3^{rd} International Workshop on Reliable Networks Design and Modeling (RNDM).

6. Analysis of the robustness of some real tele-communication networks.

1.3 Aims

The aim of this work is to make a contribution to the scientific community, providing two new metrics of robustness. These metrics could be used either by researchers or by network providers in order to evaluate the robustness of a network in response to any kind of impairment or multiple failure.

Moreover, a detailed featuring of some real tele-communication networks is presented in order to compare them using not only our metrics, but the graph robustness metrics.

Additionally, this work also provides a useful tool (PONS) in order to carry out simulations where any kind of impairment or multiple failure is caused over a network, while there are connections running over it.

1.4 Contents

This report is structured as follows. Chapter 2, firstly defines basic simulation concepts in Section 2.1. Secondly, Section 2.2 provides a review of some well known commercial simulators. Lastly, our simulator, PONS (*Path-Oriented Network Simulator*), is presented in Section 2.3. In Chapter 3, a brief background of some taxonomies that classify attacks (or multiple failures) is provided in Section 3.1. Further, in Section 3.2 a classification of network impairments is presented. Then, Chapter 4 provides a background of some well known robustness metrics, and defines these metrics, in Section 4.1. Our two new metrics (QNRM and QLRM) are presented in Section

4.3. Furthermore, a Case Study is provided in Chapter 5 in order to demonstrate that, the two new metrics defined in this work are able to evaluate the robustness of a network in response to any kind of impairment, when the functioning of a service is taken into account. In Chapter 6 we evaluate some real tele-communication networks with our metrics of robustness (QNRM and QLRM). We provide the conclusions of this work in Chapter 7 and we give some outlines issues that could be considered as future work in Chapter 8. Finally, a brief manual in A, provides information about how to use the *igraph R package*.

Chapter 2

Theory and Concepts of Simulation

Simulators have long been used in computer networks research; it is well-established as a tool to study protocols, resource allocation problems, applications, and, in general, complex systems whose behaviour, performance or other characteristic needs to be estimated or verified. Simulators are useful when working with technologies that have not been built yet, but also when the studies carried out involve networks that are beyond of what is available as test beds, because of size, coverage or technology, and building them is not feasible due to economic, technical or other reasons.

As simulators are expected to reproduce relevant aspects of reality, they must be able to cope with the advances and transformations that networking undergo along the time. Sometimes, practical impediments may appear while using a simulation system. Examples of obstacles are:

- Excessive use of memory or CPU.
- Slowness.
- Lack of support for newer technologies or protocols.

The first two problems mentioned are usually referred to as an scalability issue, while the second is of completeness of the implementation.

A simulator is usually a complex piece of software, in many respects akin to a software development tool and, as there are many of them, choosing the right one for a project is the first difficulty, even presuming solved the scalability issue. The decision criteria may include the analysis of factors such as: model abstraction supported, simulation programming language, runtime environment and degree of dynamism (whether the programs are interpreted/script-based or compiled), integration with other tools (data processing tools or other simulators), availability of customer support and

good documentation, maturity of the implementation, availability of source code, learning effort required to use it properly, and architecture, which in turn influences other quality measures such as extensibility and adaptability [10]. Many of these factors conflict with one another, so trade-offs must be considered.

In this Chapter, some basic concepts of simulation are provided in Section 2.1. Then, a review of some well known network simulators is given in Section 2.2. Finally, Section 2.3 describes the main characteristics of the *Path-Oriented Network Simulator* (PONS), tool that has been programmed in order to carry out this project.

2.1 Basic concepts

According to [11], digital computer simulation is *"the process of designing a model of a real system and conducting experiments with this model on a digital computer for a specific purpose of experimentation"*. The taxonomy given in [12] states that digital computer simulation may be divided into three categories:

1. Monte Carlo: is a method by which an inherently non-probabilistic problem is solved by a stochastic process; the explicit representation of time is not required.

2. Continuous: the variables within the simulation are continuous functions. For example a system of differential equations.

3. Discrete event: If the variables of a program change their value at precise points in simulation time the simulation is discrete event.

.

In [12] it is specified that three related forms of simulation are commonly used in the literature:

1. Combined simulation: refers generally to a simulation that has both discrete event and continuous components[1].

2. Hybrid simulation: refers to the use of an analytical sub-model within a discrete event model.

3. Gaming simulation: may refer to discrete event, continuous, and/or Monte Carlo modeling components.

In this review we focus on discrete event simulation. A simulation involves modeling a system. A system is defined as:

[1]Typically, a discrete event sub-model is encapsulated within a continuous model.

"a part of the world which we choose to regard as a whole, separated from the rest of the world for some period of consideration, a whole which we choose to consider as containing a collection of components, each characterized by a selected set of data items and patterns, and by actions which may involve itself (a component) and other components."

The system may be real or imagined and may receive input from, and/or produce output for, its environment.

A model is an abstraction of a system intended to replicate some properties of that system. The collection of properties the model is intended to replicate (for the purpose of providing answers to specific questions about the system) must include the modeling objective. Only through the objective can meaning be assigned to any given simulation program. Since by definition a model is an abstraction, details exist in the system that do not have representation in the model. In order to justify the level of abstraction, the model assumptions must be reconciled with the modeling objective.

In [13] the author defines: "a model is comprised of objects and the relationships among them. An object is anything characterized by one or more attributes to which values are assigned. The values assigned to attributes may conform to an attribute typing similar to that of conventional high level programming languages."

In a discrete event simulation, the two concepts of *time* and *state* can not be underestimated. In [13] the following primitives which permit precise delineation of the relationship between these fundamental concepts are identified:

- An *instant* is a value of system time at which the value of at least one attribute of an object can be altered.
- An *interval* is the duration between two successive instants.
- A *span* is the contiguous succession of one or more intervals.
- The *state* of an object is the enumeration of all attribute values of that object at a particular instant.

These definitions provide the basis for some widely used simulation concepts [13]:

- An *activity* is the state of an object over an interval.
- An *event* is a change in an object state, occurring at an instant, and initiates an activity precluded prior to that instant. An event is said to be determined if the only condition on event occurrence can be expressed strictly as a function of time. Otherwise, the event is contingent.

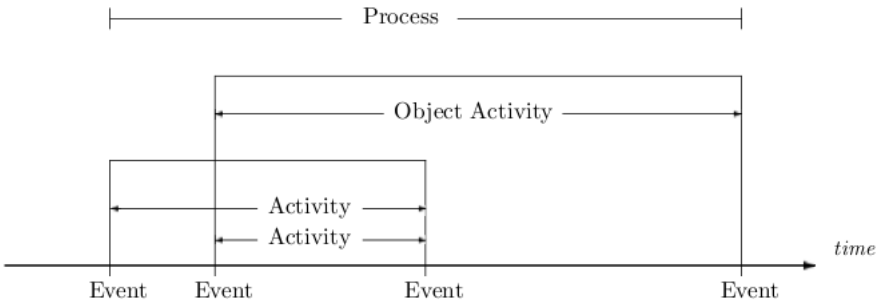

Figure 2.1: Illustration of Event, Activity and Process

- An *object activity* is the state of an object between two events describing successive state changes for that object.
- A *process* is the succession of states of an object over a span (or the contiguous succession of one or more activities).

These concepts may be viewed as illustrated in Figure 2.1. It is important to note that an activity for an object is bounded by two successive events for that object [13].

Finally, *activity* and *process* form the basis of three primary conceptual world views within discrete event simulation [14]:

- In an event scheduling world view, the modeler identifies when actions are to occur in a model.
- In an activity scanning world view, the modeler identifies why actions are to occur in a model.
- In a process interaction world view, the modeler identifies the components of a model and describes the sequence of actions of each one.

2.2 Review of some network simulators

The review provided in this section is based on analyzing the issues listed below:

- Scalability.
- Support for good software engineering practices.
- Support for data collection and aggregation.

2.2.1 OPNET

OPNET is a respected general-purpose discrete event simulator suite that comes with a graphical user interface and modules for tasks such as network modeling, planning, and analysis. It is not a freely available and open-source tool, but distributed under commercial terms. It uses a hierarchical object-based modeling aimed to match the structure of real networks, equipment, and protocols. It supports wired and wireless networks. In [15] the product is reported as being able to manage several hundred of nodes, while in the product's brochure it is claimed that its simulation engine can handle thousands [16].

Federated simulations are supported in newer versions of the product, which surely boosts its scalability. Being a mature product, it comes with extensive documentation, examples, library of protocols and network devices. The user can develop new models using a combination of programming in C/C++ with state-transition diagrams.

2.2.2 ns-2

NS is widely used in the network research community. It began in 1989 as a variation of a previous simulator called REAL. The most used version is called *ns-2* and is available at [17]. It provides substantial support for simulation of transport (e.g., TCP) and session protocols, unicast and multicast routing algorithms, and many application-level protocols, and for wired and wireless networks (ad hoc, local and satellite). *ns-2* handles arbitrary topologies, composed of routers, links and shared media and incorporates a range of link-layer topologies and scheduling and queue management algorithms.

Faithful to the ideal of being a unifying research tool for the network simulation community, it includes a large set of users' contributions. *ns-2* has served as the basis for many extensions, among them SensorSim for sensor networks and *pdns* for parallel and distributed simulations.

Network generators like Tiers and GT-ITM can be used to automate the production of arbitrary network topologies with certain user-defined properties. A network animator called NAM makes it possible to visualize the behaviour of the simulated network based on data generated while processing. *ns-2* has also an emulation interface that allows the interaction with real-world traffic, as well as traffic injection.

Networks in *ns-2* are composed of nodes, protocol agents, and links. Nodes receive packets, examine each one and decides about the appropriate outgoing interface(s). Packets are created, processed or consumed by agents, which model endpoints in the network and are used to implement protocols at various levels. Links connect two or more nodes and have a series of attributes like delay, bandwidth and queue properties [18].

Software engineering issues

ns-2 uses C++ as its implementation language, while the configuration (and scenario definition) is done in a *Tcl* derivative called *OTcl*, executed in an interpreted environment. The simulator is organized as a compiled class hierarchy of C++ objects, with the same hierarchy also written in *OTcl*, that is, written twice, but for different purposes. With the *OTcl* interpreter the user creates new objects, the network topology of nodes and links and the agents associated with the nodes, chooses scheduler, and controls the simulation (start/stop events, network failure, statistic gathering, etc.)

In [19] it is argued that this split-programming model is crucial to extensibility, one of its design goals, as it makes scripts easy to write and new protocols efficient to run. However, in [15] the author points out that this scheme complicates the addition of new components, increases the learning curve and makes debugging difficult. Also, as the documentation tends to be out of date with respect to the software, the developer has few choices: browsing the source code or asking questions on the Internet. Other authors argue that the use of class derivation as the sole mechanism for extension makes difficult reusing components developed outside the current project.

Scalability

ns-2 can handle a few thousand network elements [15] but it is not practical to work with bigger networks; it consumes too much memory and it is not fast enough. (*It is interesting to note that a single run for a simulation may be "fast enough", for example some minutes or even a few hours. The problem is that running a simulation only once has no practical scientific value.*) Higher memory consumption than other simulators is attributed in [20] to its features that provide for runtime inspection of protocol behaviour. This is also expected for it being a packet-level simulator.

The impossibility of running a simulation on a single machine because there is not enough memory may be overcome by using distributed variants like *PDNS*.

ns-2 has also been extended to perform better with mobile ad hoc networks. In [21] the authors explain their changes that achieves up to 30 times better performance in certain scenarios. Again, slow simulation and excessive memory consumption is attributed to the use of *Tcl*.

Data collection and aggregation

ns-2 offers two basic mechanisms to collect and aggregate data. The first is called trace and, in simple terms, it allows logging of events to a file or to the console while the simulation is running. As such, it is a low-level mechanism. The other is called monitor and is capable of recording counts of events, like packet arrivals and departures, packets dropped, delay experienced, etc.

With the help of classifiers, it can be applied to flows instead of packets. The output can also be directed to a file. *ns-2* also generates a trace file for NAM, the visualization tool.

In practice, and as pointed out in [22], the *ns-2* user generally has to prepare special programs or scripts to process and summarize the data obtained with the simulator's logging facility to conduct further analysis.

2.2.3 ns-3

ns-3 is, as *ns-2* is, an open sourced discrete-event network simulator available for research and educational use. *ns-3* is licensed under the GNU GPLv2 license and it can be downloaded from [23]. *ns-3* has been designed to replace the current popular *ns-2*. However, it is important to notice that *ns-3* is not an updated version of *ns-2* because it is not compatible with *ns-2*.

The basic idea of *ns-3* comes from several different network simulators including *ns-2* and *GTNetS*. In [24], the author lists the major differences that exist between *ns-3* and *ns-2*:

1. Different software core: The core of *ns-3* is written in C++ and with Python scripting interface (compared with *OTcl* in *ns-2*). Several advanced C++ design patterns are also used.

2. Attention to realism: protocol entities are designed to be closer to real computers.

3. Software integration: support the incorporation of more open-source networking software and reduce the need to rewrite models for simulation.

4. Support for virtualization: lightweight virtual machines can be used.

5. Tracing architecture: *ns-3* is developing a tracing and statistics gathering framework trying to enable customization of the output without rebuilding the simulation core.

Nonetheless, nowadays one of the biggest handicaps of *ns-3* is that it needs the research community to collaborate. Moreover, as stated in [24], the simulation credibility needs to be improved. One of the limitations of simulations, in general, is that it often suffers from lack of credibility. According to [24], there are four points that *ns-3* should address in order to find a solution to this problem:

1. Hosting *ns-3* code and scripts for published work.

2. Tutorials on how to do things right.

3. Flexible means to configure and record values.

4. Support for ported code should make model validation easier and more credible.

Finally, in order to face other simulators like *OPNET*, *ns-3* would need a lot of specialized maintainers, who could answer to user's questions, fix bugs found by users, and help to test and validating the system.

2.2.4 OMNet++

According to its website [25], OMNeT++ is a "public-source, component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel". Its primary application area is the simulation of communication networks, even though it is equally possible to use it to simulate processes and systems in other areas. Its implementation language is C++ and offers a simulation class library in that language. The product is free for academic and non-profit use; other uses require a commercial license.

Message passing is the central communication mechanism used by the simulator components, which simplifies running simulations in a parallel and distributed environment with OMNet++. Compared to the well known *ns-2*, it is a very young tool (its development started in 1998) and, because of this reason, it comes with far less number of pre-built modules and protocols.

In OMNeT++, a model of a network consists of hierarchically nested entities called modules. Modules communicate via message passing, where the messages can contain arbitrarily complex data structures. Modules can send messages either directly to their destination or along a predefined path, through gates and connections, which have assigned properties like bandwidth, delay and error rate. Modules can have parameters which are used to customize module behaviour, to create flexible model topologies and for module communication, as shared variables. The user must provide the lowest level module in the hierarchy, containing the the algorithms in the model. During simulation execution, simple modules appear to run in parallel, since they are implemented as coroutines.

OMNeT++ uses two extension languages that the user must employ to write models and control the simulation. One is C++ and the other is called NED. Files written in NED describe the topology of the network; they are translated to C++ by a tool that comes with OMNeT++, although in current versions it can be loaded dynamically and, therefore, translation is no longer obligatory. Furthermore, module functionality is written directly in C++ and defines how to process each packet that arrives to an input gate, as well as how to send it.

The files that comprise a simulation project are translated to object code and, after linking, a standard executable file is obtained. OMNeT++ uses

Tcl as a library to support graphical user interface, although for deployment a console version is recommended in the product's manual.

Software engineering issues

In [15] OMNeT++ is praised for its overall design, ease of use and flexibility, especially when comparing it with *ns-2*. This is expected, considering that it is a much younger product and its design should have benefited from past experiences with other simulators. Besides, the user base is smaller and the development process seems to happen in a more centralized way.

Scalability

In [15] a good scalability is expected because of its support for parallel and distributed simulations. OMNeT++ primarily uses Message Passing Interface (MPI) to communicate Logical Processes (LP). A complex model must be partitioned such that each submodel is run by a different LP. On shared-memory multiprocessors, named pipes can be used instead of MPI.

Data collection and aggregation

OMNeT++ logging capabilities are similar to *ns-2*, although judging from the documentation, it looks even less specialized. It supports output vectors (objects of a predefined C++ class) that programs can use to collect data and generate output to a file at simulation termination. The users' manual suggests to use external tools to automatically process these files. The documentation does not specify if these output vectors are kept in memory during simulation, which can lead to memory exhaustion, or what is the proper or easiest way to consolidate logs generated on different nodes in a distributed simulation without introducing runtime bottlenecks due to I/O.

2.2.5 GTNetS

GTNetS (Georgia Tech Network Simulator) was presented in 2003 with the stated goal of allowing much larger-scale simulations that could be created easily by existing tools at that time. To address scalability, it was designed to support parallel and distributed simulations. It is interesting to point out that the author started *GTNetS* after having implemented *pdns*, the Parallel/Distributed *ns*. He was convinced that achieving further improvements in topology scale with the baseline *ns-2* would be difficult due to basic design deficiencies.

The simulator uses C++ as the implementation and simulation language. Therefore, the simulation must be driven by a main program written in this language. This is coherent with the author's position that the use of *Tcl* in *ns-2* contributes to substantial memory consumption.

The simulator was deliberately designed to structure simulation networks like the real ones. In *GTNetS*, "there is a clear distinction between nodes, interfaces, links, and protocols. Node objects represent the basic functionality of a network node (either a router or end-user system), and contain one or more Interface objects. Each interface object has an IP Address and an associated network mask, as well as a Link object encapsulates the behaviour of the transmission medium. Packets in *GTNetS* consist of a list of Protocol Data Unit objects (PDUs). This list is created and extended while a packet moves down the protocol stack through the various layers. When moving up the stack, each protocol layer removes and processes the corresponding protocol header, as its done in a real protocol stack. Each protocol layer communicates with the layer below it by invoking a DataRequest method, specifying the packet (and current state of the PDU stack), and any protocol specific information required by the next lower layer. Similarly, protocols accept up-calls from the layer below using a DataIndication method. Layer 4 endpoints are bound to port numbers, either well-known fixed values or transient ports, just like real layer 4 endpoints. Connections between layer 4 endpoints are by IP Address and Port Number, in a fashion nearly identical to actual protocols." [22].

GTNetS comes with a number of well-known protocols at all the supported layers. For example, at the application layer there are models for a web browser and the Gnutella, and even for distributed denial-of-service attacks. At the transport layer, it has models for TCP Reno, TCP NewReno, TCP Tahoe and TCP SACK. A difference with ns-2 is that the end-points of a connection need not be created manually because "applications" are bound to ports in the transport layer and start listening automatically, just like they do under the familiar TCP/IP Socket interface. Supported link-layer protocols include IEEE 802.3 and IEEE 802.11, for wired and wireless networks respectively. An extension called GTSNetS (note the additional "S" between T and N) is presented in [26], to specifically target sensor networks. No further information is presented here about it.

Software engineering issues

The simulator is still very young; the author himself points out features that are still immature or missing. It was not possible to find other author's experiences with the software.

Scalability

The author reports preliminary results in [15] in which networks with more than 480,000 nodes are simulated in a distributed environment consisting of 32 systems with a total of 128 processors. The total running time has an almost-linear growth. In a related paper [27], the scalability of *pdns* and

GTNetS is studied, concluding that both have very similar performance when run on workstation-class machines and on a cluster at the Pittsburgh Supercomputing Center.

Data collection and aggregation

It has a number of data summarization primitives to assist the user in gathering network performance statistics during the simulation execution. The author does not provide much detail about mechanisms, but mentions a couple of examples. One is about the Web Browser object, that has an optional histogram object that can be used to trace the response time for each requested web object. The other refers to keeping track of TCP sequence numbers sent and acknowledged as a function of the simulated time. Logs can be written selectively to files. As with other simulators, external programs must be used for processing and analysis.

2.2.6 Shawn

Shawn is a new simulator designed to work with huge wireless sensor networks, e.g., those with hundreds of thousands of nodes. The increased scalability is achieved by using more abstract models: instead of fully simulating communication media according to its real-world characteristics, or lower-level networking protocols, it is better to use a well-chosen random distribution on message delay and loss [28].

Shawn deliberately departs in its approach from other network simulators. Its authors believe that simulation of network stacks is not a fruitful approach for the evaluation of protocols and algorithms for wireless sensor networks. Shawn simulates the effect caused by a phenomenon, not the phenomenon itself. For example, instead of simulating a complete MAC layer including the radio propagation model, its effects are modeled, i.e., packet loss and corruption. The simulations run much faster, but details about the performance and behaviour of the physical layer or the packets are impossible to obtain, as it is with more traditional simulators like *ns-2*.

Software engineering issues

Shawn is implemented in C++ and this language must also be used to implement the models. In [29] there is a document intended for developers, but it is still very incomplete.

Scalability

Being scalability its main selling point, it is not surprising that, according to its authors, its performance is excellent. A comparison with *ns-2* is presented in [28]. For the example, a network with 2,000 nodes simulated in

ns-2 requires more than 25 hours, while in *Shawn* it takes only 19 seconds. Memory consumption is equally impressive: for the same network, *ns-2* uses more than 220 MB of RAM, while *Shawn* uses less than nine.

Data collection and aggregation

Both [28] and the developers' guide omit any reference to logging or any data collection mechanism. In the latter, however, appears mentioned the Apache-sponsored library *log4cplus*.

2.3 Path-Oriented Network Simulator

The author of this project had to deal, two years ago, with OMNet++ in his bachelor's degree of Computer Science. In order to carry out that project in the BCDS (*Broadband Communications and Distributed Systems*) research group, some well known network simulators were reviewed (some of them have been reviewed in Section 2.2). Therefore, OMNet++ was chosen to be used. Although the experience was totally enriching, too much effort and time had to be dedicated to learn how to use the simulator, and how to program new modules that were not able on the framework that OMNet++ was offering in 2008.

One may ask *"why do you need to program your own simulator, if there are a lot of them?"*. The answer is easy: the level of abstraction that is needed in order to carry out this project (and some others of the BCDS research group) can not be found in any of the simulators presented. For example, OMNet++ is a powerful network simulator but it is too detailed for what this project requires. It needs to program modules, configure them, etc. Basically, it is a matter of the time that has to be dedicated to both, learn how to use the simulator and program new modules for it.

The initial version of the *Path-Oriented Network Simulator* (PONS) was programmed by Juan Segovia, member of the BCDS research group, at the end of 2008. PONS has been programmed with Java and its main characteristic is that, as its name indicates, it is a *path-oriented* simulator. It means that a connection is affected if one of the components of the path, where the connection is going through, fails. It is also interesting to note that it was initially programmed to be easily extended. Thus, since the earliest version of PONS (2008), several modules have been added in order to extend its functionality.

It is important to note that the last module that has been added to PONS is one that, depending on several parameters, can cause different kind of impairments on a network. The impairments considered in PONS are extensively defined in Chapter 3.

Therefore, PONS has three main inputs, in order to carry out simulations:

1. A network topology: the topology must be provided in either *.NET* or *.SGF* format.

2. A traffic matrix: the traffic file must be a *.TRF* file, generated with the *gentraf* [30] tool. This traffic file must have a relation with the network topology.

3. A kind of impairment or multiple failure.

Moreover, PONS has a set of tools that are able to calculate several topological features, given a network topology. For example, the *average shortest path length*, the *largest eigenvalue*, the *assortativity coefficient* or the *joint degree distribution*, are some of the features that PONS' tools are able to calculate. These features and some more are defined in Chapter 4.

Chapter 3

Network Impairments

In this Chapter, a brief background of some well known taxonomies of attacks is provided. Then, the taxonomy proposed in this project is presented.

3.1 Background

Assuming that a network is more robust if the service on the network performs better, where performance of the service is assessed when the network is either (a) in a conventional state or (b) under perturbations (failures, virus spreadings, etc.), the robustness does depend on the type of impairment that occurs. From here on, the term *impairment* refers to any kind of attack, multiple or cascading failure that can occur within a network.

Attacks over the years have ranged from throwing a glass of water over a computer, to more developed techniques. Several taxonomies have been proposed in order to classify network attacks; specifically within communication networks. Some of the references provided in this Section, are not only focused on *network attacks*, but on *computer* ones. With respect to a *network attack*, a network could be used in several ways (such as a worm) in order to propagate any kind of attack or multiple failure.

In [31], many network attacks regarding to a TCP/IP based network are considered. In [32] a general overview of the types of attacks that are related to Internet's security is given. In [33], the author presents an exhaustive review of several taxonomies, analyses in detail each kind of attack that is considered and, finally, presents one of the most complete taxonomies that can be found in the literature. The proposed taxonomy in [33] consists of four dimensions: *"The first dimension covers the attack vector and the main behaviour of the attack. The second dimension allows for classification of the attack targets. Vulnerabilities are classified in the third dimension and payloads in the fourth"*. As a novelty, in [34] a taxonomy of attacks on 3G networks is provided. One of the latest taxonomies presented can be found in [35], called AVOIDIT (Attack Vector, Operational Impact, Defense,

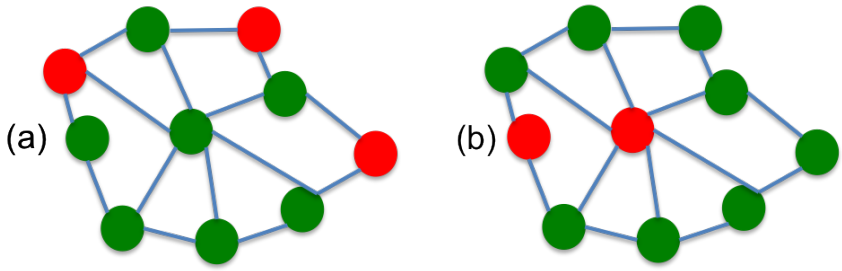

Figure 3.1: Examples of a SR impairment. (a) and (b) show that nodes are chosen randomly.

Information Impact, and Target). Although it is a cyber attack taxonomy, it presents some interesting features, because five major classifiers (as the taxonomy's name indicates) characterize the nature of an attack.

3.2 Proposed taxonomy

After the background provided in the previous Section, the discussion presented here simplifies such previous categorizations and focuses on classifying the types of impairments that can occur on the *nodes* of a network. The taxonomy presented below has been defined in order to fulfill the actual requirements of this project. Nonetheless, it has also been defined in order to be easy extended, so as to consider other components of a network, such as *links*.

Therefore, impairments or multiple failures are basically divided into two groups: *statics* and *dynamics*. The former is related to the idea of affecting a network permanently and just once, while the latter is related to an impairment that has a temporal dimension.

3.2.1 Static

Static impairments are essentially one-off attacks that affect one or more nodes at any given point. There are, in essence, two forms of static impairments:

Random (SR (*Static Random*))

In the SR case, nodal attacks occur indiscriminately selecting nodes at random. Fig. 3.1 shows this kind of impairments.

Target (ST (*Static Target*))

Nodes in an ST attack are chosen in order to maximize the effect of that attack; there is an element of discrimination in the impairment. The choice of attack target may be a function of network-defined features such as nodal degree, between-ness centrality or clustering, as well as other "real-world" features, such as the number of users potentially affected and socio-political and economic considerations. Fig. 3.2, Fig. 3.3 and Fig. 3.4 show some examples of ST attacks, considering different elements of discrimination.

3.2.2 Dynamic

This second type of failures (commonly related to multiple failures such as cascading failures) has a temporal dimension. Two types are defined:

Epidemical (DE (*Dynamic Epidemical*))

Considering a DE, a failure occurs in a node (or a set of nodes of the network) and the failure can spread through the network (becoming an epidemic) or not. The rise and decline in epidemic prevalence of an infectious disease (or failure) is a probability phenomenon dependent upon the transfer of an effective dose of the infectious agent from an infected individual to a susceptible one [36]. Fig. 3.5 shows an example of how an epidemic can affect a network.

This type of failures is based on epidemic models (EM) and there are several forms of them. The first type, called the *Susceptible-Infected* (SI) considers nodes as being either susceptible (S) or infected (I). This type assumes that the infected nodes will remain infected forever and, so, can be used for "worst case propagation". Another type is the *Susceptible-Infected-Susceptible* (SIS), which considers that a susceptible node can become infected on contact with another infected node, then recovers with some likelihood of becoming susceptible again. Therefore, nodes will change their state from susceptible to infected, and vice versa, several times. The third kind is the *Susceptible-Infected-Removed* (SIR), which extends the SI model to take into account the *removed* state. In the SIR group, a node can be infected just once because when the infected nodes recover, they become immune and will no longer pass the infection onto others. Finally there are two models that extend the SIR one: SIDR (*Susceptible Infected Detected Removed*) and SIRS (*Susceptible Infected Removed Susceptible*). The first one adds a Detected (D) state, and is used to study the *virus throttling*, which is an automatic mechanism for restraining or slowing down the spread of diseases. The second one considers that after a node becomes removed, they remain in that state for a specific period and then go back to the susceptible state.

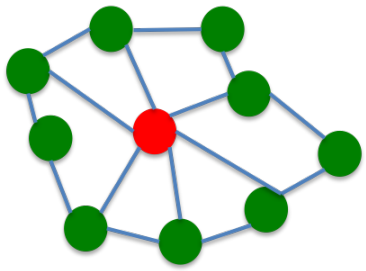

Figure 3.2: Example of a ST impairment. The element of discrimination is the *nodal degree*.

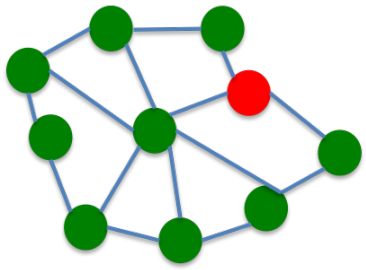

Figure 3.3: Example of a ST impairment. The element of discrimination is the *between-ness centrality*.

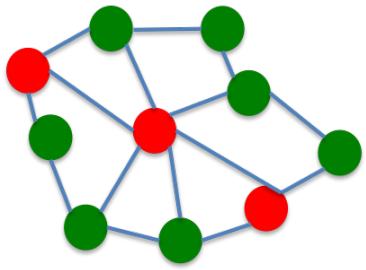

Figure 3.4: Example of a ST impairment. The element of discrimination is, in this case, to disconnect the network.

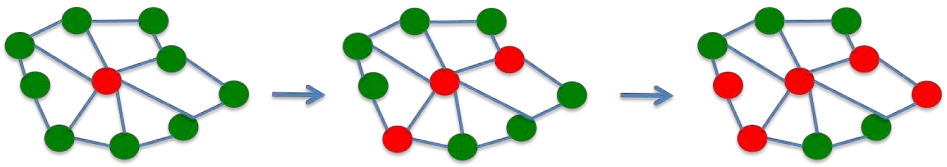

Figure 3.5: Example of a DE impairment. A failure occurs on a node, and after a period of time, it spreads to its neighbours.

Periodical (DP (*Dynamic Periodical*))

A DP is, simply, any kind of impairment that occurs periodically following its characteristic cycle.

Chapter 4

Metrics of Robustness

In this Chapter, firstly, a background of some well known graph robustness metrics is provided. Secondly, these metrics are defined. Lastly, the two metrics proposed in this project are presented.

4.1 Background

Several topology features are considered in the classical approach which is based upon basic concepts of graph theory: these include average nodal degree, node connectivity [37], heterogeneity [38], symmetry ratio [39], diameter, average shortest path length [40], average neighbor connectivity [41] and the assortativity coefficient [41].

Moreover, in a more contemporary approach, other metrics in networking literature were introduced; including the largest eigenvalue [41][42], the second smallest laplacian eigenvalue [43] and the average two-terminal reliability [44].

The set of metrics presented in this section is defined in more detail in 4.2. None of these metrics matches completely with the advanced concept of robustness that is considered in this project.

4.2 Topology characteristics

4.2.1 Average nodal degree ($\overline{k}$)

This is the coarsest connectivity feature of any topology. Networks with higher $\overline{k}$ are "better-connected" on average, and, consequently, are likely to be more robust. On one hand, "more robust" means that there are more chances to establish new connections. However, if a node with a high nodal degree fails, potential higher numbers of connections are also prone to be affected. Thus, this metric by itself provides only a limited measure of the

robustness of a network which is likely to vary depending on how the nodal degree is actually distributed over the graph.

4.2.2 Node connectivity

This metric represents the smallest number of nodes whose removal results in a disconnected or single-node graph. Moreover, connectivity can also be defined as the smallest number of node-distinct paths between any two nodes [37]. This metric gives a crude indication of the robustness of a network in response to any of the impairments defined in Section 3.2.

4.2.3 Heterogeneity

Heterogeneity is the standard deviation of the average nodal degree divided by the average nodal degree [38]. In Sydney et al. [45] a range of different attacks (SR and ST) are invoked over a variety of networks, and it can be observed that heterogeneous networks are likely to be more robust. The lower the magnitude of its heterogeneity, the greater the robustness of the topology.

4.2.4 Symmetry ratio

This ratio is essentially the quotient between the number of distinct eigenvalues (obtained from the adjacency matrix of the network) of the network and the network diameter. Therefore, on high-symmetry networks, with symmetry values between 1 and 3, the impact of losing a node does not depend on which node is lost, what means that networks perform equally in response to a random (SR) or a target attack (ST) [39]. Random networks do not have, in general, high symmetry values. However, for random graphs, where nodes are of equal importance in a statistical sense: since links are placed randomly, no node is privileged by design. This condition can not be applied to small-world or scale-free networks.

4.2.5 Diameter

The diameter is, like the average nodal degree, another coarse robustness metric of a network. It is the longest of all the shortest paths between pairs of nodes. In general, one would wish the diameter of networks to be low. Scale-free networks generally have small diameters, but are not particularly robust in response to deliberate attacks (ST), due to their relatively low value of node connectivity. Nonetheless, small-world networks represent a combination of the advantages of the properties of random networks (where no node is privileged by design) and scale-free networks (where there is a low diameter). We also note that *expansion*, the diameter of a network

normalized by its size, could be also used in order to carry a comparison analysis [46].

4.2.6 Average shortest path length

Average shortest path length (ASPL) is calculated as an average of all the shortest paths between all the possible origin-destination node pairs of the network. Networks with small ASPL are more robust because, in response to any kind of impairment (SR, ST, DE or DP), they are likely to lose fewer connections.

4.2.7 Largest eigenvalue (λ)

Most networks with high values for the largest eigenvalue have a small diameter and are more robust. In general, networks with larger eigenvalues have more node and link disjoint paths to choose from. Therefore, this metric provides bounds on network robustness with respect to both link and node removals [41]. This metric is also associated in defining the *epidemic threshold* of a network, which correlates with the severity of an epidemic failure (DE) on a network [42].

4.2.8 Second smallest Laplacian eigenvalue (λ_2)

This metric, also known as *algebraic connectivity*, measures how difficult it is to break the network into islands or individual components. The larger the λ_2, the greater the robustness of a topology against both node and link removal [43].

4.2.9 Average neighbor connectivity

This metric provides information about 1-hop neighborhoods around a node. It is a summary statistic of the *Joint degree distribution (JDD)* and it is simply calculated as the average neighbor degree of the average k-degree node [41].

4.2.10 Assortativity coefficient (r)

The assortativity coefficient r, can take values between $-1 \leq r \leq 1$. When $r < 0$ the network is called to be *dissassortative*, which means that has an excess of links connecting nodes of dissimilar degrees. Such networks are vulnerable to both static random and targeted attacks (SR and ST). The opposite properties apply to *assortative* networks with $r > 0$ that have an excess of links connecting nodes of similar degrees [41].

4.2.11 Average two-terminal reliability (A2TR)

This metric is the probability that a randomly chosen pair of nodes is connected. If the network is fully connected the value of A2TR is 1. Otherwise, it is the sum over the number of node pairs in every connected component divided by the total number of node pairs in the network. This ratio gives the fraction of node pairs that are connected to each other. Therefore, the higher the value (for a given number of nodes removed), the more robust the network is in response to an static random attack (SR) that affects the same number of nodes [44].

4.3 Quantitative and Qualitative Robustness Metrics

Two new robustness metrics are now proposed, which capture the definition given in this project for robustness and both define key aspects of the services (in our case *connections*) that run over a network. Services can be classified according to different Quality of Service (QoS) parameters, such as: delay, jitter, packet loss, etc. Network failures some, if not all, of these parameters resulting in revised QoS levels. Moreover, from the network operator perspective, failures also affect the number of established and future connection demands. Considering both aspects (quantity and quality) two new metrics are proposed to evaluate how network services could be affected in response to different multiple failure scenarios. On one hand, and in order to simplify, the *QuaLitative Robustness Metric* (QLRM) quantifies variations in the *average shortest path length* of established connections, reflecting that the path length is a function of some key QoS parameters (delays, packet loss, etc.). On the other hand, the *QuaNtitative Robustness Metric* (QNRM) evaluates the number of *blocked connections*.

4.3.1 QuaNtitative Robustness Metric

The *QuaNtitative Robustness Metric* or QNRM analyses how an impairment of any kind (SR, ST, DE or DP) affects the number of connections established on a network. In this metric, the number of *Blocked Connections (BC)* in each time step are analyzed. We define a BC as a connection that should have been established at time t but could not be established as a consequence of nodal failures.

Define $BC(t)$ as the number of BC in a given time step, $TTC(t)$ as the number of connections that should have been established in the same time step and $Total$ as the maximum number of time steps. In order to compare different topologies that may not have the same number of $TTC(t)$ in each time step, we compute our metric, in each time step t, with the quotient shown in the following equation:

$$QNRM[t] = \frac{BC(t)}{TTC(t)} \tag{4.1}$$

Finally, we calculate the average of all values obtained during the interval of interest:

$$QNRM = \frac{\sum_{t=1}^{Total} QNRM[t]}{Total} \tag{4.2}$$

4.3.2 QuaLitative Robustness Metric

The *QuaLitative Robustness Metric* or QLRM analyses how the quality of service on a network varies, when any kind of impairment (SR, ST, DE or DP) occurs. This metric measures the *average shortest path length (ASPL)* in each time step. In contrast to QNRM, QLRM evaluates the *Established Connections (EC)*.

In order to compare the QLRM for different topologies the values obtained from the *average shortest path length (ASPL)* are normalized. Define $U(ASPL)$ as the quotient of the *standard deviation* of the ASPL of the topology and its ASPL. Also define $U(KoI)$ as the same quotient, but calculated when any *Kind of Impairment* has occurred in the network (and the ASPL has been affected). Then, this metric is defined as follows:

$$QLRM = \frac{U(ASPL)}{U(KoI)} \tag{4.3}$$

The metric is calculated by normalizing the values with the standard deviations obtained because, the magnitude of increase or decrease of the ASPL of a topology in response to an impairment is not of prime importance for this metric, but rather its variation. With this value we are able to determine the deterioration of the QoS of a network when an impairment occurs, compared to the QoS that the unimpaired network should normally provide.

Chapter 5

Case Study

In this Chapter, we choose six different topologies as exemplars to review the metrics defined in Section 4.3 and to compare the previously defined metrics from the literature with the two proposed in this project. The aim of this Chapter is not to try to assert which kind of the topologies presented is more robust, but to demonstrate that the proposed metrics in this project (QNRM and QLRM) are able to evaluate the robustness of a network, when services are running over it.

The rest of this Chapter is structured as follows: in Section 5.1, we characterize a set of six exemplar topologies with the metrics described in Section 4.2. Furthermore, in Section 5.2 we define a simulation scenario in order to calculate the two new metrics of robustness presented in this work. Finally, in Section 5.3, first we provide a robustness ranking of the six networks regarding to the graph robustness metrics. Secondly, we show and analyze the results obtained from the simulations.

5.1 Topologies

The six topologies analyzed in this Chapter are presented below and their key characteristics are listed in Table 5.1. Topologies are all related to *complex networks*. They have been proposed to show different characteristics in terms of diameter and average node degree.

The library that has been used in order to generate them is called *igraph* [47]. *igraph* is is a free software package for creating and manipulating undirected and directed graphs. It runs on most modern machines and operating systems, and it is tested on MS Windows, Mac OSX and various Linux versions. *igraph* is available as a C library, as a Python extension, as a Ruby extension and also as a *R* (a free software environment for statistical computing and graphics) package. In this project, the *igraph*'s R package has been used. In A, a tutorial of how to use *igraph* on R is provided.

The random networks have been obtained using the Erdős–Rényi model

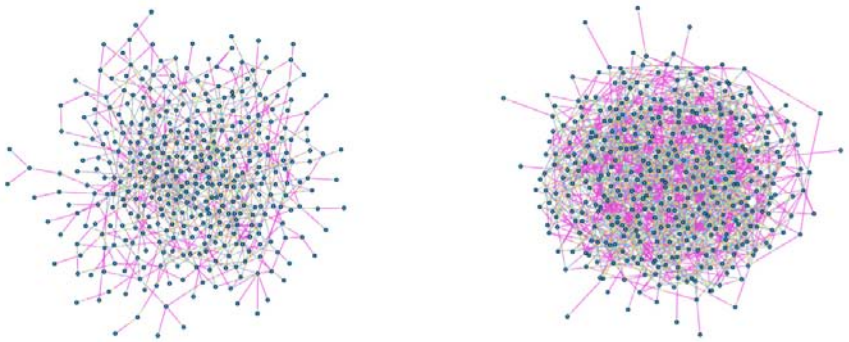

Figure 5.1: er400d3 (left) and er400d6 (right)

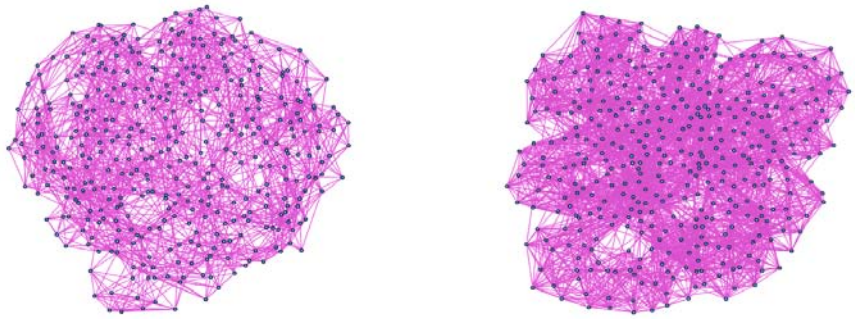

Figure 5.2: sw400d10 (left) and sw400d20 (right)

[48], small-world using the Watts and Strogatz model [49] and scale-free ones using the Barabási–Albert (BA) model [50]. In Table 5.1, the two random networks are the ones that start with *er-*, while the small-world and scale-free networks are indicated by h the *sw-* and *sf-* prefixes respectively. The two random networks can be observed in Fig. 5.1, the two small-world in Fig. 5.2 and the two scale-free in Fig. 5.3.

5.2 Simulation Scenario

In order to calculate our metrics of robustness, the simulation scenario must be detailed. All the simulations last for 10000 time steps with a traffic load of 80000 connections in total. Source and destination of the connections have been selected randomly and with the restriction that they cannot be adjacent (connections are a minimum of two hops). There is no restriction in link capacity so if there are no failures, all the connections are accepted. The generation of the connections and their duration follows negative exponential

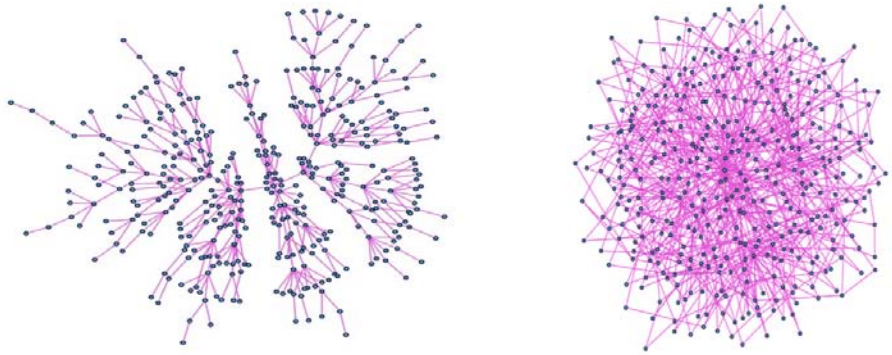

Figure 5.3: sf400d2 (left) and sf400d4 (right)

distributions with average inter-arrival and holding times of 0.12 and 100 time steps respectively.

Simulations causing the following impairments are carried out:

- SR: A static random attack that affects the 20% of nodes of the network is activated at the start of the simulation.
- DE: The *Susceptible-Infected-Disabled* (SID) epidemic model [51], previously presented by the authors of this work, will be used in this case study. A dynamic epidemic failure that initially affects the 3% of nodes of the network is activated at the start of the simulation, reaching a total of 20% of nodes affected after a period of time (this period is different for each topology and depends upon its specific topological features).

Then, we are able to obtain results showing how the set of exemplar topologies performs in response to either a static random attack (SR) or a dynamic epidemic failure (DE), when both affect the same number of nodes.

5.3 Results

The results of the Case Study are presented over. First, a ranking of the topologies based around the traditional robustness metrics is listed. Secondly, the simulation results are provided in order to analyze and compare them with the grades obtained from the graph robustness metrics.

In Table 5.2 the classification based on the features of the topologies of 5.1 can be observed. In this classification, *1* represents the most robust with increasing rank representing successively reduced robustness. The last row indicates the global ranking of the topologies and is the simple unweighted average of the positions of the previous rankings of each topology. This

average could be calculated using different weights for each kind of metric, depending on the specific necessity of the network designer. However, in the first instance, we consider all the metrics to be equally weighted and will contemplate the option of a weighted average in future work.

As it can be seen in Table 5.2, the ranking of the *average two-terminal reliability* is provided. This ranking has been obtained from Fig. 5.4, which shows how the reliability of the set of topologies evolves when nodes are removed from the network. It is important to note that the *average nodal degree* ranks the topologies as A2TR does. Furthermore, the rest of rankings in Table 5.2 are based on topological features of the networks.

On the third row, topologies are ranked by their *node connectivity*. As expected, scale-free networks can be disconnected by removal of just one node. It is interesting to note that, according to the *node connectivity*, the random topologies have the same performance as the scale-free networks. Regarding the *largest eigenvalue*, *sw400d20* is the most robust topology, indicating that, under an epidemic failure (DE), this network performs better than the others.

Thereafter, rankings of the *average shortest path length* and the *second smallest laplacian eigenvalue* show interesting results: although *sw400d20* is the most robust, there is no match in the 2^{nd}, 3^{rd} and 4^{th} most robust topologies. The following two rows show the classification based on the *average network connectivity* and on the *assortativity coefficient*. The small-world pair are the most robust, implying that these two networks are less vulnerable under any kind of static impairments (SR or ST).

Finally, the classification regarding to the *symmetry ratio* is shown. In this case, it can observed that the two topologies with less *average nodal degree* are the ones that are more symmetric (and more robust). However, later on we show that, contrarily, these two topologies are highly affected by any kind of attack (SR and DE) in comparison with the other four exemplars. Thus, *symmetry ratio* would not be a suitable metric in relation to *low average nodal degree* networks.

To summarize the ranking provided in Table 5.2, a global ranking has been calculated and listed in the last row. This final and summary ranking gives an approximation to the robustness of the networks considered in this case study considering the traditional robustness metrics which omit considerations about any connections on the network. Here, the two small-world are the most robust, followed by the random network *er400d6* and the scale-free *sf400d4*. Although the small-world *sw400d20* is ranked as the most robust, there is one metric that considers that this network would be less robust than the others (the *symmetry ratio*). In addition, some metrics differ in identifying the 2^{nd} and 3^{rd} most robust topologies. This means that one should really use a group of metrics to define the robustness rather than rely on any single graph robustness metric. Thus, considering several graph based robustness metrics becomes necessary when robustness of a network is

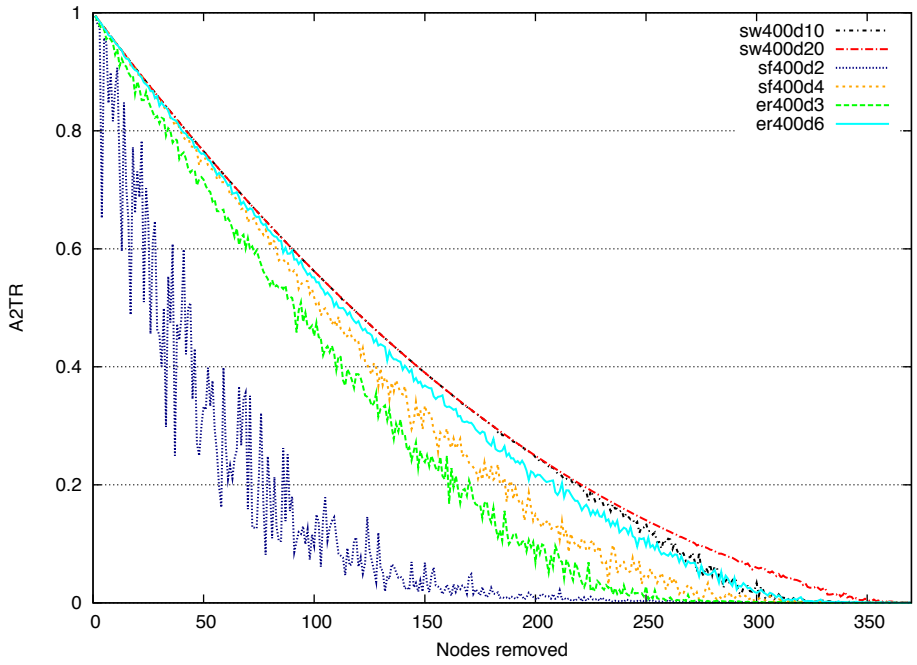

Figure 5.4: Average Two-Terminal Reliability of the Case Study topologies

to be analyzed, but such an approach would not be sufficient for a network provider, because it does not take into account the connections that run over a network and does not give any information about the service performance of a network under any kind of impairment.

The results of the simulations are presented further down. In Table 5.3 results associated with the QNRM metric can be observed. Table 5.3 is divided as follows: rows from 1 until 3 pertain to the behavior of the network in response to a SR impairment while from 4 until 6 pertain to the metric's value in response to an DE failure. The last two rows show the relation between the DE and the SR in order to facilitate a comparison between the robustness of the networks when either a SR or a DE failure occurs.

As can be observed in Table 5.3, regarding the SR impairment, the most robust topologies are the two small-world networks, followed by *sf400d4* and *er400d6*. The worst network in this scenario is *sf400d2* because it blocks almost the 80% of the connections that should be established. Nevertheless, it is interesting to note that, in response to a DE failure, the third most robust network is the *er400d6*, followed by *sf400d4*.

The last row shows a classification of the topologies sorted by the ratio calculated in the row above it. Then, *sw400d20* is the topology that shows the most improvement in its performance when comparing a SR and a DE failure; the number of blocked connections reducing to almost 50% when

an epidemic failure (DE) occurs. Second (*er400d6*) and third (*sw400d10*) position networks have a ratio value extremely close to unity which means that they perform equitably in response to both kinds of impairments. The network where the number of blocked connections in response to an epidemic rises most, compared with a static random impairment, is *sf400d4*. This means that *sf400d4* is a network where the performance varies according to the kind of attack; a characteristic that may not be desirable for a telecommunications network supporting high levels of dynamic connections.

Furthermore, in Table 5.4 results obtained for the QLRM metric are shown. Table 5.4 is structured as follows: first two rows display the average shortest path length (ASPL) feature of each topology with their standard deviation, rows 3 until 6 are related to the robustness metric in response to a SR impairment. The four following rows are associated with the behaviour under a DE failure while the last two rows of the table shows the relation between the DE and the SR to allow a comparison of the robustness of the networks under either SR or DE failures.

The fifth row of Table 5.4 shows the value of QLRM in response to a SR impairment. The ranking provided in the following row reveals that, when the quality of the service is assessed, *sw400d20* is the most robust network of the exemplar set. Networks *er400d6* and *sw400d10* have similar behavior, although the random network is slightly more robust than the small-world examplar. The worst network in terms of QLRM is *sf400d2* because it has the largest variation of the ASPL. It is interesting to note that *sf400d2* decreases its ASPL when a SR impairment occurs because, as can be seen in Table 5.3, this network blocks almost the 80% of connections and establishes only *short-path-length* connections. The robustness characteristics of the exemplar set described in terms of QLRM is similar to the DE failure case (ninth row). The last two rows of Table 5.4 show that (as with QNRM), *sf400d4* is the network that performs poorest when considering DE and SR impairments. It can also be observed that *sw400d20* is, again, the most robust network, and that *er400d6* or *sw400d10* have similar behavior.

Finally, it can be observed that the metrics shown in Table 5.2 represent a relatively simplistic approach to define the robustness of a network because the metrics do not take into account the connections that are running over the network. Therefore, the results shown in Table 5.3 and Table 5.4 demonstrate that our metrics are able to define the robustness of a network, capturing the definition of robustness assumed in this paper. QNRM and QLRM are able to inform the network designer how the performance of the service would degrade in response to a particular type of impairment. Furthermore, in order to choose the topology when it is known how the functioning of a service would be affected, graph robustness metrics could be considered. Moreover, other kind of topological features (such as the number of links) could also be helpful for the network designer when cost of the network is of key importance.

Table 5.1: Characteristics of the Case Study network topologies

Characteristic	er400d3	er400d6	sw400d10	sw400d20	sf400d2	sf400d4
Number of nodes	400	400	400	400	400	400
Number of links	618	1205	1996	3975	399	789
Average nodal degree (AND)	3.00	6.03	9.98	19.88	2	3.95
Stdev	1.67748	2.43705	0.97569	1.34867	1.87584	3.69135
Minimum nodal degree	1	1	6	15	1	1
Node connectivity (NC)	1	1	6	15	1	1
Heterogeneity	0.5591	0.4041	0.0977	0.0678	0.9379	0.9345
Symmetry ratio	30.7692	50	57.1428	66.6666	20.6842	50
Diameter	12	7	6	5	18	7
Average shortest path length	5.48	3.56	3.75	2.79	7.28	3.91
Stdev	1.5046	0.86358	0.97115	0.67285	2.38783	0.94979
Largest eigenvalue	4.1158	7.1677	10.10778	20.002101	4.513222	8.343718
Second smallest Laplacian eigenvalue	0.11025	0.518127	0.635512	1.657585	0.003921	0.527415
Clustering coefficient	0.2000	0.0512	0.48678	0.53384	0.615	0.03033
Assortativity coefficient	-0.14419	-0.00079	0.00888	0.01585	-0.08945	-0.03827
Average neighbor connectivity	0.0100210072	0.0175646585	0.025251003	0.0500408253	0.0094094886	0.018522224444
Degree distribution $P(k)$	-	-	-	-	$\sim k^{-3.006345}$	$\sim k^{-2.949767}$

Table 5.2: Ranking of robustness of the Case Study network topologies, based on topology features

	er400d3	er400d6	sw400d10	sw400d20	sf400d2	sf400d4
Average two-terminal reliability (Fig. 5.4)	5	3	2	1	6	4
Average nodal degree	5	3	2	1	6	4
Node connectivity	3	3	2	1	3	3
Heterogeneity	4	3	2	1	6	5
Largest eigenvalue	6	4	2	1	5	3
Average shortest path length	5	2	3	1	6	4
Second smallest Laplacian Eigenvalue	5	4	2	1	6	3
Average neighbor connectivity	5	4	2	1	6	3
Assortativity coefficient	6	3	2	1	5	4
Symmetry ratio	2	3	4	5	1	3
Global Ranking	(4.6) 5	(3.2) 3	(2.3) 2	(1.4) 1	(5) 6	(3.6) 4

Table 5.3: QuaNtitative Robustness Metric Results of the Case Study network topologies

Impairment		er400d3	er400d6	sw400d10	sw400d20	sf400d2	sf400d4
Static Random (SR)	QNRM	0.4482	0.3682	0.3453	0.3437	0.7974	0.3592
	Standard Deviation	0.0002	0.0002	0.0004	0.0005	0.0004	0.0005
	Ranking	5	4	2	1	6	3
Dynamic Epidemic (DE)	QNRM	0.4863	0.3687	0.3502	0.1750	0.9148	0.4175
	Standard Deviation	0.0010	0.0007	0.0019	0.0004	0.0009	0.0014
	Ranking	5	3	2	1	6	4
	$\frac{QNRM_{DE}}{QNRM_{SR}}$	1.085	1.0011	1.0142	0.5090	1.1472	1.1622
	Ratio Ranking	4	2	3	1	5	6

Table 5.4: QuaLitative Robustness Metric Results of the Case Study network topologies

Impairment		er400d3	er400d6	sw400d10	sw400d20	sf400d2	sf400d4
Static Random (SR)	ASPL	5.48	3.56	3.75	2.79	7.28	3.91
	Standard Deviation	1.5046	0.86358	0.97115	0.67285	2.38783	0.94979
	ASPL	6.283	3.818	4.191	3.021	5.818	4.251
	Standard Deviation	0.00032	0.00004	0.0001	0.00002	0.0018	0.00006
	QLRM	0.00019	0.00004	0.00009	0.00003	0.00094	0.00006
	Ranking	5	2	4	1	6	3
Dynamic Epidemic (DE)	ASPL	6.932	3.981	4.153	2.956	5.006	5.224
	Standard Deviation	0.0107	0.0004	0.001	0.00002	0.0532	0.0035
	QLRM	0.00562	0.00041	0.00093	0.00003	0.03240	0.00276
	Ranking	5	2	3	1	6	4
	Total ratio: $\frac{QLRM_{DE}}{QLRM_{SR}}$	30.3069	9.5905	10.0915	1.0219	34.3496	47.4684
	Ratio Ranking	4	2	3	1	5	6

Chapter 6

Robustness Analysis of Real Networks

In this Chapter a set of real tele-communication networks is evaluated. Following the same methodology than in Chapter 5, in Section 6.1 the set of topologies is introduced. Further, Section 6.2 presents the simulation scenario that has been defined in order to carry out the analysis. Finally, in Section 6.3, firstly, a robustness ranking of the set of networks regarding to the graph robustness metrics is provided. Secondly, the results obtained from the simulations are shown and analyzed.

6.1 Topologies

The five topologies considered in the analysis carried out in this Chapter are presented below. All of them have been obtained from [52], a repository of well known real tele-communication networks. It is interesting to note that, almost with all the topologies provided in [52], an image with the geographical position of them is attached. The five topologies are:

1. *cogentco*: *cogentco*'s worldwide Tier-1 optical IP network is one of the largest of its kind, with direct IP connectivity to more than 3,500 AS (Autonomous System) networks around the world and over 11,000 Gbps inter-networking capacity. The network map can be observed in Fig. 6.1.

2. *deltacom*: *deltacom* is a network that primarily provides services to 8 states of USA. The network map can be observed in Fig. 6.2.

3. *ion*: *ion* is a fiber network connecting over 60 rural New York State communities and their surrounding areas to the Information Superhighway. The network contains over 2,200 route miles of fiber, and

Figure 6.1: cogentco

consists of four diverse SONET rings. This network can be observed in Fig. 6.3.

4. *kdl*: *kentucky datalink (kdl)* is a fiber network that spans nearly 30,000 miles and reaches into 26 states. The network map is shown in Fig. 6.4.

5. *uscarrier*: *uscarrier* is a network that spans over 20,000 fiber miles. It can be observed in Fig. 6.5.

Table 6.1 shows the key characteristics of the topologies described above. It can be observed that, while four of them have a *number of nodes* that ranges between 100 and 200, one of them (*kdl*) has been chosen in order to have one topology with a higher value of it.

6.2 Simulation Scenario

In this Chapter, the simulation scenario considered is almost the same than the one considered in 5.2. However, it differs in some aspects, which are described below:

- SR: In the analysis carried out in this Chapter, the static attack affects the 10% of nodes of the network, and as described in 5.2, it is activated at the start of the simulation.
- DE: The dynamic epidemic failure, as described in 5.2, is activated at the start of the simulation affecting the 3% of nodes, but this time, reaching a total of 10% of nodes affected after a period of time.

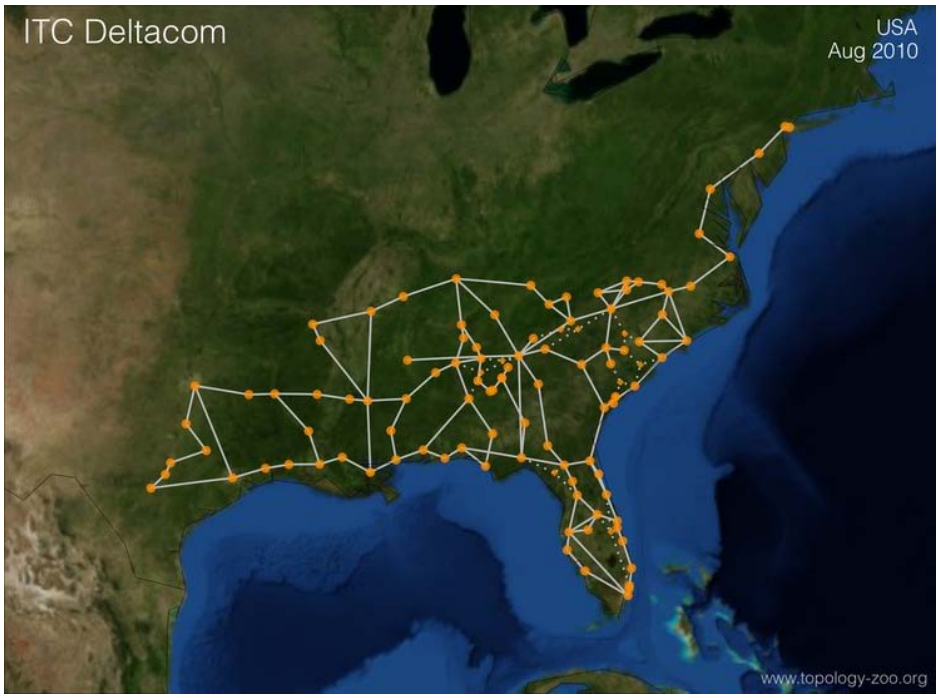

Figure 6.2: deltacom

Figure 6.3: ion

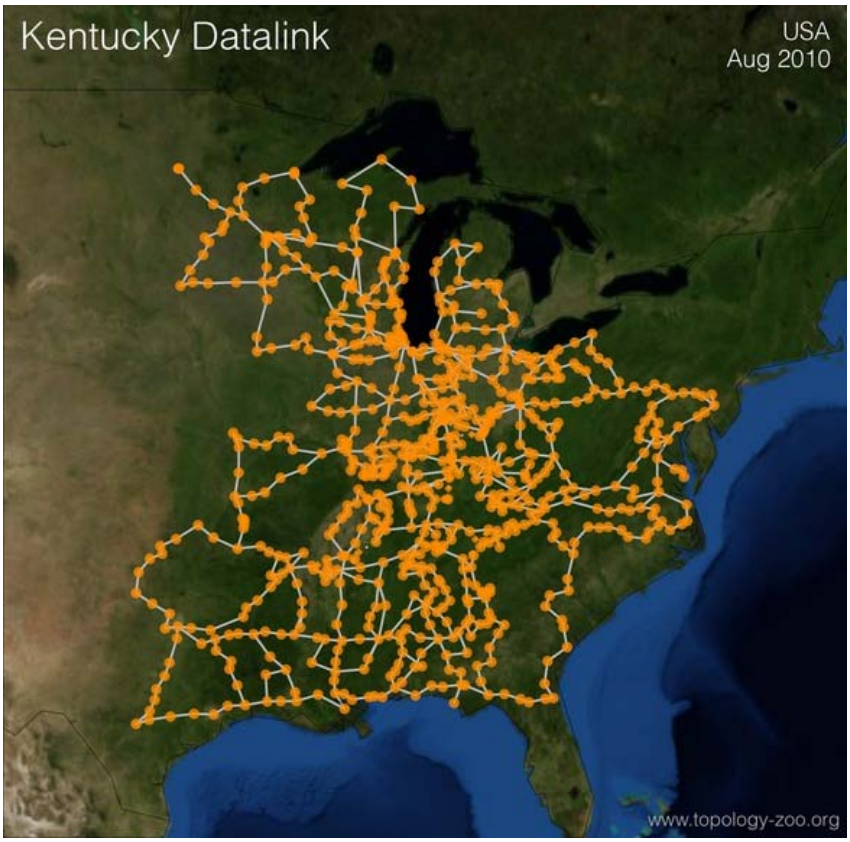

Figure 6.4: kdl

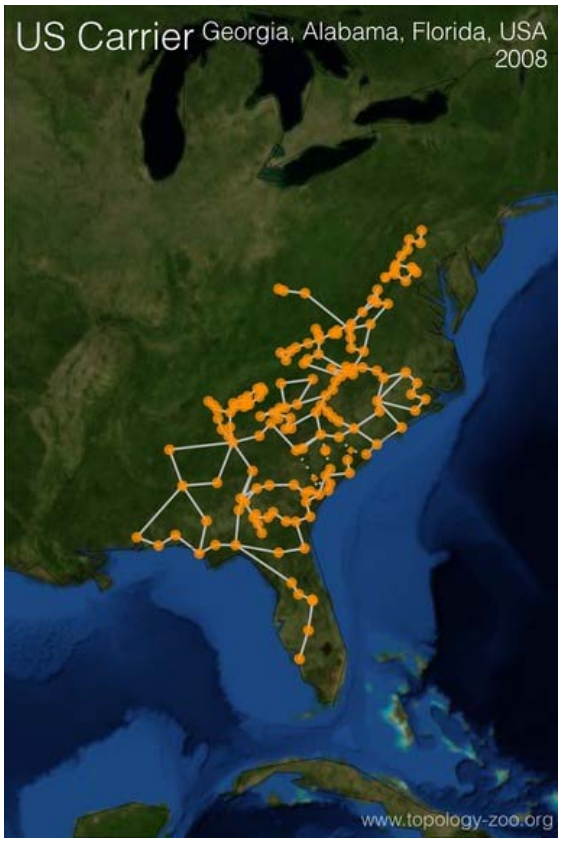

Figure 6.5: uscarrier

Then, we are able to obtain results showing how the set of real network topologies performs in response to either a static random attack (SR) or a dynamic epidemic failure (DE), when both affect the same number of nodes.

6.3 Results

The results of this Chapter are presented over. First, the *Average Two-Terminal Reliability* of the networks is shown in Fig. 6.6. Secondly, a ranking of the topologies based around the traditional robustness metrics is listed. Lastly, the simulation results are provided in order to analyze and compare them with the grades obtained from the graph robustness metrics.

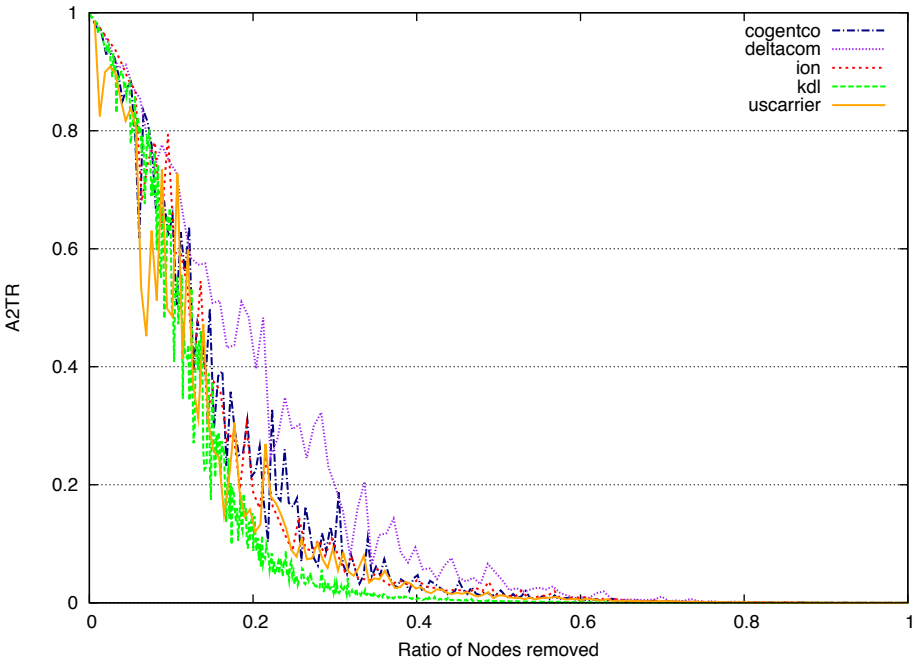

Figure 6.6: Average Two-Terminal Reliability of the set of real network topologies

Fig. 6.6 shows the *Average Two-Terminal Reliability* of the five networks. Because not all of them have the same number of nodes, the number of nodes removed has been uniformed for each one of the networks, in order to plot them all in one graphic. One can notice that the networks considered in the Case Study, could be easily ranked regarding to their *A2TR*. However, as it can be observed in Fig. 6.6, the real networks considered in this Chapter would be difficult to rank, because all of them have a similar *average nodal degree*. However, it is clear that the *deltacom* network is the most robust

one, regarding to the *A2TR*. The rest of them have a roughly similar curve, although the *kdl* network is the first one that reaches values near 0.

In Table 6.2 the classification based on the features of the topologies of 6.1 can be observed. In this classification, as it was described in Chapter 5, *1* represents the most robust with increasing rank representing successively reduced robustness. The last row indicates the global ranking of the topologies and is the simple unweighted average of the positions of the previous rankings of each topology. This average could be calculated using different weights for each kind of metric, depending on the specific necessity of the network service provider. However, in the first instance, we consider all the metrics to be equally weighted and will contemplate the option of a weighted average in future work.

As it can be seen in Table 6.2, on the first row the ranking regarding to the *average nodal degree (AND)* is provided. The most robust network is *deltacom*, followed by *cogencto*, while the less robust one (that has the slowest value of *AND*) is *ion*. It is important to note that, the five networks considered in this Chapter have a slightly similar *average nodal degree*, ranging from 2 to 3. This is an important difference between the set of networks used in the Case Study 5 and the set of real networks used in this Chapter. Real tele-communication networks do not have high values of *AND*.

On the second row, the five networks are ranked by their *node connectivity* and, because all of them have the same value of it (1), all of them are ranked equally. It was not expected that real networks could be disconnected by removal of just one node. Regarding the *heterogeneity*, *ion* is the most robust topology, while *deltacom* and *cogentco* are the worst ones.

Thereafter, rankings of the *largest eigenvalue* and the *second smallest laplacian eigenvalue* show interesting results. Although both of them rank *deltacom* as the most robust network, there is no match in the 2^{nd} and 3^{rd} most robust topologies. As the second most robust, while the former ranks *cogentco*, the latter ranks *ion*, which results to be the worst ranked by the former.

The following two rows show the classification based on the *average network connectivity* and on the *assortativity coefficient*. *deltacom* is the most robust, implying that this network is less vulnerable under any kind of static impairments (SR or ST). Finally, the classification regarding to the *symmetry* ratio is shown. In this case, it can be observed that, just like the ranking of *heterogeneity*, *deltacom* is not the most robust one.

To summarize the ranking provided in Table 6.2, a global ranking has been calculated and listed in the last row. This final summary ranking gives an approximation to the robustness of the networks considered in this Chapter considering the traditional robustness metrics, which omit considerations about any connections on the network. Here, *deltacom* is the most robust, followed by *uscarrier* in second place. *cogentco* and *ion* have both the same average value, so they both are in third place. It is interesting to observe

that, if the global ranking had not been calculated with the same weights for all the metrics, *cogentco* and *ion* would not have had the same ranking.

As observed in Chapter 5, some metrics differ in identifying the 2^{nd} and 3^{rd} most robust topologies. This means that one should really use a group of metrics to define the robustness rather than rely on any single graph robustness metric. Thus, considering several graph based robustness metrics becomes necessary when robustness of a network is to be analyzed, but such an approach would not be sufficient for a network provider, because it does not take into account the connections that run over a network and does not give any information about the service performance of a network under any kind of impairment.

The results of the simulations are presented further down. In Table 6.3 results associated with the QNRM metric can be observed. Table 6.3 is divided as follows: rows from 1 until 3 pertain to the behavior of the network in response to a SR impairment while from 4 until 6 pertain to the metric's value in response to an DE failure. The last two rows show the relation between the DE and the SR in order to facilitate a comparison between the robustness of the networks when either a SR or a DE failure occurs.

As can be observed in Table 6.3, regarding to the SR impairment, the most robust topology is *deltacom*, blocking almost the 35% of the connections that should be established, when a SR impairment affects the 10% of the nodes. Further, the 2^{nd} most robust topology is *cogentco*, blocking around 36% of the connections, while the 3^{rd} most robust is *kdl*, blocking almost the 37% of them. Therefore, the difference between these networks is not significant and the three of them can be considered equally regarding to a SR impairment. Then, *ion* is in 4^{th} position, blocking almost the 50% of the connections. *uscarrier* is the less robust one because it blocks almost the 68% of the connections that should be established.

Moreover, in response to a DE failure, the ranking is sorted completely the other way round. Here, *uscarrier* is the most robust topology, blocking around the 20% of the connections. *ion* is the 2^{nd} most robust blocking around the 27%. It is interesting to note that, *deltacom*, that was the most robust in response to a SR impairment, is the less robust in response to a DE failure, blocking almost the 75% of the connections.

The last row of Table 6.3 shows a classification of the topologies sorted by the ratio calculated in the row above it. Then, the *uscarrier* is the topology that shows the most improvement in its performance when comparing a SR and a DE failure; the number of blocked connections reducing to almost 30% when an epidemic failure (DE) occurs. Second (*ion*) and third (*cogentco*) position networks have a ratio value under the unity, which means that they perform better in response to a DE failure than to a SR impairment. *deltacom* is the topology that shows the least improvement in its performance when comparing a SR and a DE failure.

Furthermore, in Table 6.4 results obtained for the QLRM metric are

shown. Table 6.4 is structured as follows: first two rows display the average shortest path length (ASPL) feature of each topology with their standard deviation, rows 3 until 6 are related to the robustness metric in response to a SR impairment. The four following rows are associated with the behaviour under a DE failure while the last two rows of the table shows the relation between the DE and the SR to allow a comparison of the robustness of the networks under either SR or DE failures.

The fifth row of Table 6.4 shows the value of QLRM in response to a SR impairment. The ranking provided in the following row reveals that, when the quality of the service is assessed, *deltacom* is the most robust network, which means that the *average shortest path length* does not varies significantly when a SR impairment is caused. The worst network in terms of QLRM is *kdl* because it has the largest variation of the ASPL. It is interesting to note that *uscarrier* decreases its ASPL when a SR impairment occurs because, as can be seen in Table 6.3, this network blocks almost the 68% of connections and establishes only short-path-length connections.

Regarding to the DE failure case (ninth row), the most robust topology is *cogentco*, while *kdl* continues being the less robust network. The last two rows of Table 6.4 show that, *kdl* is the network that performs poorest when considering DE and SR impairments. It can also be observed that *cogentco* is the most robust one.

Finally, as observerd in Chapter 5, it can be observed that the metrics shown in Table 6.2 represent a relatively simplistic approach to define the robustness of a network because the metrics do not take into account the connections that are running over the network. Comparing the results shown in Table 6.2 with the ones shown in Table 6.3 and Table 6.4 one may notice that just few positions of the rankings match. For example, it is interesting to observe that, while in Table 6.2 *deltacom* appears to be the most robust one, in Table 6.3 it appears to be the less robust in response to a DE failure. These kind of details must be taken into account when a robustness analysis is going to be carried out. QNRM and QLRM have shown that they are a useful tools for a network service provider. Nonetheless, in order to choose the topology when it is known how the functioning of a service would be affected, graph robustness metrics could also be considered. Moreover, other kind of topological features (such as the number of links) could also be helpful for the network designer when cost of the network is of key importance.

Table 6.1: Characteristics of the set of real tele-communication network topologies

Characteristic	cogentco	deltacom	ion	kdl	uscarrier
Number of nodes	197	113	125	754	158
Number of links	242	161	146	895	189
Average nodal degree (AND)	2.46	2.85	2.34	2.37	2.39
Stdev	1.04706	1.21171	0.082251	0.84254	0.8204
Minimum nodal degree	1	1	1	1	1
Node connectivity	1	1	1	1	1
Heterogeneity	0.425634146	0.425161404	0.03515	0.35550211	0.343263598
Symmetry ratio	6.793103	4.708333	4.807692	12.779661	4.361111
Diameter	28	23	25	58	35
Average shortest path length	10.52	7.16	10.14	22.73	12.09
Stdev	5.09079	3.79633	4.78563	10.64351	6.45623
Largest eigenvalue	3.77828	3.889182	2.955117	3.168197	2.984177
Second smallest Laplacian eigenvalue	0.008573	0.022354	0.013318	0.001943	0.005653
Clustering coefficient	0.12884	0.14197	0.0992	0.08404	0.10886
Assortativity coefficient	0.01956	0.03832	-0.2797	-0.10462	-0.09518
Average neighbor connectivity	0.014800135	0.030002218	0.021155546	0.003549304	0.017018839

Table 6.2: Ranking of robustness of the set of real tele-communication network topologies, based on topological features

	cogentco	deltacom	ion	kdl	uscarrier
Average nodal degree	2	1	5	4	3
Node connectivity	1	1	1	1	1
Heterogeneity	5	4	1	3	2
Largest eigenvalue	2	1	5	3	4
Average shortest path length	3	1	2	5	4
Second smallest Laplacian Eigenvalue	3	1	2	5	4
Average neighbor connectivity	4	1	2	5	3
Assortativity coefficient	2	1	5	4	3
Symmetry ratio	4	2	3	5	1
Global Ranking	(2.88) 3	(1.44) 1	(2.88) 3	(3.88) 4	(2.77) 2

Table 6.3: QuaNtitative Robustness Metric results of the set of real tele-communication network topologies

Impairment		cogentco	deltacom	ion	kdl	uscarrier
Static Random (SR)	QNRM	0.3634	0.3477	0.4881	0.3678	0.6797
	Standard Deviation	0.00052	0.0005	0.0005	0.00053	0.00072
	Ranking	2	1	4	3	5
Dynamic Epidemic (DE)	QNRM	0.2826	0.7478	0.2624	0.4257	0.2039
	Standard Deviation	0.00196	0.010833	0.04608	0.00738	0.01498
	Ranking	3	5	2	4	1
	$\frac{QNRM_{DE}}{QNRM_{SR}}$	0.7776	2.1507	0.5375	1.1574	0.2999
	Ratio Ranking	3	5	2	4	1

Table 6.4: QuaLitative Robustness Metric results of the set of real tele-communication network topologies

Impairment		cogentco	deltacom	ion	kdl	uscarrier
Static Random (SR)	ASPL	10.52	7.16	10.14	22.73	12.09
	Standard Deviation	5.09079	3.79633	4.78563	10.64351	6.45623
	ASPL	10.995	7.394	12.755	30.741	8.652
	Standard Deviation	0.00135	0.00043	0.00907	0.02938	0.00361
	QLRM	0.00025	0.000109	0.001506	0.00204	0.00078
	Ranking	2	1	4	5	3
Dynamic Epidemic (DE)	ASPL	10.709	7.389	10.4878	32.956	11.444
	Standard Deviation	0.00162	0.00454	0.01497	0.69544	0.07916
	QLRM	0.00031	0.0011	0.00302	0.04506	0.0129
	Ranking	1	2	3	5	4
	Total ratio: $\frac{QLRM_{DE}}{QLRM_{SR}}$	1.232	10.5652	2.0072	22.0796	16.5781
	Ratio Ranking	1	3	2	5	4

Chapter 7

Conclusions

In this Chapter the conclusions of the project are exposed. First, some general project's conclusions are presented. Then, more specific conclusions about the Case Study of Chapter 5 and the Robustness Analysis of Real Networks of Chapter 6 are given.

In this project, some basic concepts of simulation theory are provided. Moreover, some well-known network simulators are reviewed. Because all these simulators require a detailed simulation scenario specification, the *Path-Oriented Network Simulator (PONS)* has been presented. Some features of PONS are then provided.

Thereafter, because the robustness of a network when services are taken into account, depends on the kind of impairment that is caused on the network, a brief overview of some previous works based on classifying different types of attacks (or failures) is provided. Then, in order to simplify previous classifications, a taxonomy focused in network impairments is presented.

Following that, some well-known traditional graph robustness metrics are reviewed. As such metrics do not take into account services that run over a network, two new robustness metrics have been presented: *QuaNtitative Robustness Metric* or QNRM and *QuaLitative Robustness Metric* or QLRM. Moreover, it has been shown that our proposals are able to quantify the performance of a network under particular types of impairment.

In the Case study, a set of topologies has been defined: two random networks, two small-world and two scale-free. In order to generate the set of topologies, the *igraph R package* has been used. A ranking has been provided in order to classify this set of networks by their features related to robustness metrics. Nevertheless, although these metrics could be used to describe the robustness of a network, they are not capable of indicating how service operation can be affected under any kind of multiple failure scenario. Then, the set of topologies has been evaluated with the two proposed metrics (QNRM and QLRM), under static random attacks (SR) and dynamic epidemic failures (DE), both of them affecting the 20% of nodes.

Results of the Case Study have shown that the proposed metrics are able to describe the robustness of a network under different impairments (in this case study: SR and DE). Further, results have also shown which topologies, from the set considered in the Case Study, are more robust under a specific attack: a network can perform differently depending on the impairment form. In summary, the two small-world network are the most robust. Nonetheless, *er400d6* and *sf400d4* could also be considered on specific cases. As it can be observed, the rankings provided by our metrics and the rankings of the graph robustness metrics are different. Therefore, the choice of most robust topologies should be done complementing the information provided by QNRM and QLRM with those provided by the graph robustness metrics. Furthermore, if the cost is taken into account when designing a network, some other features (such as the number of links) of the topologies could be considered.

In the Robustness Analysis of Real Networks, a set of five real networks has been chosen. Four of these networks have a number of nodes that is between 100 and 200 nodes, while one network has been chosen with more than 700 nodes. The network maps of the five networks are provided.

Results of the Robustness Analysis of Real Networks have shown that, according to the ranking provided by the graph robustness metrics, *deltacom* is the most robust network, *uscarrier* is the second most robust and *kdl* (the network with a higher number of nodes) is the least robust. However, if the information provided by this ranking is complemented with the results given by our metrics (QNRM and QLRM), a network provider is able to know how the services of the set of five real networks will be affected in response to a given type of impairment. For example, QNRM shows that *deltacom* (the one that, according to the classical metrics, is the most robust) is the worst network (least robust) in response to a DE failure affecting the 10% of nodes, because it blocks almost the 75% of the connections that should be established. Additionally, QNRM shows that *uscarrier* is the worst network when a SR impairment affecting the 10% of nodes is caused, because it blocks almost the 68% of the connections. Then, QLRM shows that *cogentco*, the network that the graph robustness metrics rank in the 3^{rd} position, is the second most robust topology in response to a SR impairment. Furthermore, *cogentco* is the most robust topology in response to a DE failure. This means that, regarding to the variation of the quality of service, *cogentco* is better than *uscarrier* (which is ranked in 2^{nd} position by the classical metrics). Therefore, this information would not be known if only graph robustness metrics were taken into account, what demonstrates that our metrics are two useful tools that can be strongly considered by network providers. Moreover, regarding to real tele-communication networks, it can be observed that some networks have been (casually or not) designed to be more robust in response to a specific kind of impairment (for example, *deltacom*) than to another one.

Chapter 8

Further Work

In this Chapter some outlines issues that could be considered as a future work are defined.

First of all, regarding to the *Path-Oriented Network Simulator (PONS)*, there are some aspects that could be considered to improve the simulator:

1. The current version of PONS has not got yet a module in order to process the results. Actually, this step is carried out with a set of scripts programmed in Perl. As future work it could be interesting to add a module that could carry out this process.

2. Impairments, in the current versions of PONS, are focused on nodes. As a future work, impairments could be applied also to links, or clusters, etc.

3. When a simulation has to be carried out, it has to be done within a command-line environment. Adding a Graphical User Interface (GUI) in order to execute simulations would be helpful.

4. It could be interesting to write an extensive documentation of PONS. Then, the simulator could be published on Internet and the research community could participate providing PONS of new modules.

Regarding to the quantification of the robustness of a network:

1. As a first step, it could be also interesting to evaluate the set of networks considered in this project (the ones of the Case Study, and the real networks) under static target (ST) and dynamic periodical (DP) failures.

2. More specific QoS parameters could be included in the QLRM metric, not just the *average shortest path length*, giving an improved evaluation of the QoS for the different network services.

3. It could be interesting to evaluate a wider set of topologies in order to be able to further define what type of topology (random, small-world or scale-free) is more robust in response to a given kind of impairment. To do that, several topologies of each kind could be considered, differing in the *average nodal degree*. For example, a set of topologies with *average nodal degree* of 2, 3, 4, 5, 6, 7, 8, 9, 10, 15 and 20 could be considered for each kind of network.

4. Taking into consideration that all the results provided in this project have been obtained assuming an infinite capacity of links and a unique traffic pattern, it could be interesting to carry out the same analysis that has been presented in this project, but within a simulation scenario of finite capacity. Comparing both results (the ones obtained from infinite capacity simulations with the ones from finite capacity) it could be possible to know how many connections are lost due to the capacity, and not only how the service is affected by a failure or an impairment, but also by the capacity of the network.

5. Moreover, it could also be interesting to carry out analysis (with both finite and infinite capacity) but with different traffic patterns, in order to know what kind of topology (random, small-world and scale-free) or what real network is more suitable for a specific pattern of traffic.

Finally, it could be interesting to find correlations between our metrics and both graph robustness metrics and some other metrics that consider connections running over a network [9] [53].

Acknowledgements

This report presents the work that I have performed during my Erasmus scholarship at University of Strathclyde. Here, I would like to thank many people, without whom this report would not be possible. In the first place, I would like to thank my two supervisors and promotors: David Harle and Eusebi Calle. I feel lucky to be their student. I am thankful to them for their guidance, patience, for sharing ideas and knowledge in countless inspiring and open discussions. Their enthusiasm makes my research much more enjoyable.

I would like to express my gratitude to the members of the *Broadband Communications and Distributed Systems* research group, specially to Juan Segovia and Pere Vilà, for their time and effort spent guiding me and for the nice conversations we had. It is my great honor to have them involved in my research.

Working on the 3^{rd} floor of the Royal College building has been very enjoyable.

Finally, I would like to thank my parents for their unlimited love and support, and to my girlfriend for her care and humor, and for her inspiring and unlimited ideas, as well as for her encouraging attitude towards work and life. My sincere gratitude goes to my good friends, for the time we have enjoyed together.

Bibliography

[1] "Computer security," 1996.

[2] U. S.-Canada Power System Outage Task Force, *Interim Report: Causes of the August 14, 2003 Blackout in the United States and Canada*, November 2003.

[3] C. W. Johnson, "Analysing the causes of the italian and swiss blackout, 28th september 2003," in *Proceedings of the twelfth Australian workshop on Safety critical systems and software and safety-related programmable systems - Volume 86*, ser. SCS '07. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2007, pp. 21–30.

[4] BBC, "Sasser net worm affects millions," 2004.

[5] Donnan and Shawn, "Indonesian outage leaves 100m without electricity," *Financial Times*, 2005.

[6] ITPRO. http://www.itpro.co.uk/, 2006.

[7] A. Downie, "Brazil blackout raises more questions for the olympics," *Time*, 2009.

[8] Guardian. http://www.guardian.co.uk/, 2010.

[9] A. Sydney, C. M. Scoglio, P. Schumm, and R. E. Kooij, "Elasticity: Topological characterization of robustness in complex networks," *CoRR*, vol. abs/0811.4040, 2008.

[10] D. A. Cavalcanti, J. Kelner, P. R. Cunha, and D. H. Sadok, "A simulation environment for analysis of quality of service in mobile cellular networks."

[11] R. E. Shannon, "Introduction to the art and science of simulation," in *Proceedings of the 30th conference on Winter simulation*, ser. WSC '98. Los Alamitos, CA, USA: IEEE Computer Society Press, 1998, pp. 7–14.

[12] R. E. Nance, "A history of discrete event simulation programming languages," in *The second ACM SIGPLAN conference on History of programming languages*, ser. HOPL-II. New York, NY, USA: ACM, 1993, pp. 149–175.

[13] ——, "The time and state relationships in simulation modeling," *Commun. ACM*, vol. 24, pp. 173–179, April 1981.

[14] E. H. Page, S. P. Grin, and S. L. Rother, "Providing conceptual framework support for distributed web-based simulation within the high level architecture," 1998.

[15] G. A. D. Caro, "Analysis of simulation environments for mobile ad hoc networks," Tech. Rep., 2003.

[16] OPNET. http://www.opnet.com.

[17] ns 2. http://www.isi.edu/nsnam/ns.

[18] K. Fall and K. Varadhan, *The ns Manual (formerly ns Notes and Documentation)*, Jan. 2009.

[19] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," 2000.

[20] D. M. Nicol, "Scalability of network simulators revisited," in *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, Orlando, FL, Feb. 2003.

[21] V. Naoumov, T. Gross, and D. Informatik, "Simulation of large ad hoc networks," 2003.

[22] G. F. Riley, "The georgia tech network simulator," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, ser. MoMeTools '03. New York, NY, USA: ACM, 2003, pp. 5–12.

[23] ns 3. http://www.nsnam.org/download.html.

[24] J. Pan, "A survey of network simulation tools: Current status and future developments," Tech. Rep., Nov. 2008.

[25] OMNet++. http://www.omnetpp.org/.

[26] E. Ould-Ahmed-Vall, G. F. Riley, B. S. Heck, and D. Reddy, "Simulation of large-scale sensor networks using gtsnets," in *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 211–218.

[27] R. M. Fujimoto, K. Perumalla, A. Park, H. Wu, M. H. Ammar, and G. F. Riley, "Large-scale network simulation – how big? how fast," in *In Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems (MASCOTS*, 2003.

[28] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer, "Shawn: A new approach to simulating wireless sensor networks," 2005.

[29] Shawn. http://www.swarmnet.de/shawn/.

[30] gentraf. www.vtpi.org/gentraf.pdf.

[31] C. Rodgers, "Threats to tcp/ip network security," 2001.

[32] W. John and S. Tafvelin, "Analysis of internet backbone traffic and header anomalies observed," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 111–116.

[33] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Computers & Security (COMPSEC)*, vol. 24, no. 1, pp. 31–43, 2005.

[34] K. Kotapati, P. Liu, Y. Sun, and T. F. La Porta, "A Taxonomy of Cyber Attacks on 3G Networks," Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, Tech. Rep. NAS-TR-0021-2005, January 2005.

[35] S. Shiva, C. Simmons, C. Ellis, D. Dasgupta, S. Roy, and Wu, "AVOIDIT: A cyber attack taxonomy." University of Memphis, Tech. Rep., August, 2009.

[36] "Enc. britannica," *http://dictionary.reference.com/browse/epidemic*.

[37] A. H. Dekker and B. D. Colbert, "Network robustness and graph topology," in *Proceedings of the 27th Australasian conference on Computer science - Volume 26*, ser. ACSC '04. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2004, pp. 359–368.

[38] J. Dong and S. Horvath, "Understanding Network Concepts in Modules," *BMC Systems Biology*, vol. 1, no. 1, 2007.

[39] A. H. Dekker and B. Colbert, "The symmetry ratio of a network," in *Proceedings of the 2005 Australasian symposium on Theory of computing - Volume 41*, ser. CATS '05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 13–20.

[40] C. Shannon and D. Moore, "The spread of the witty worm," *IEEE Security and Privacy*, vol. 2, pp. 46–50, 2004.

[41] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, k. c. claffy, and A. Vahdat, "The internet as-level topology: three data sources and one definitive metric," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 17–26, January 2006.

[42] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, "Epidemic thresholds in real networks," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 4, pp. 1–26, 2008.

[43] A. Jamakovic and S. Uhlig, "Influence of the network structure on robustness," in *ICON*, 2007, pp. 278–283.

[44] S. Neumayer and E. Modiano, "Network reliability with geographically correlated failures," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 1658–1666.

[45] A. Sydney, C. Scoglio, M. Youssef, and P. Schumm, "Characterising the robustness of complex networks," *Int. J. Internet Technol. Secur. Syst.*, vol. 2, pp. 291–320, December 2010.

[46] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topology generators: degree-based vs. structural," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM, 2002, pp. 147–159.

[47] igraph. http://igraph.sourceforge.net/.

[48] B. Bollobas, *Random Graphs*, W. Fulton, A. Katok, F. Kirwan, P. Sarnak, B. Simon, and B. Totaro, Eds. Cambridge University Press, 2001.

[49] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.

[50] A. L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, October 1999.

[51] E. Calle, J. Ripoll, J. Segovia, P. Vilà, and M. Manzano, "A multiple failure propagation model in gmpls-based networks," *IEEE Network*, vol. 24, pp. 17–22, November 2010.

[52] topology zoo. http://www.topology-zoo.org/.

[53] W. Molisz and J. Rak, "Impact of wdm network topology characteristics on the extent of failure losses," *13th International Conference on Transparent Optical Networks (ICTON)*, 2010.

[54] T. R. P. for Statistical Computing. http://www.r-project.org/.

Appendix A

How to: *igraph* as an R package

A.1 Introduction

In this Section, the following information regarding to *igraph* has been taken from [47]. However, Section A.2 and Section A.3 have been written based on the personal experience of the author of this project.

igraph is a free software package, written in ANSI C, for creating and manipulating undirected and directed graphs. It includes implementations for classic graph theory problems like minimum spanning trees and network flow, and also implements algorithms for some recent network analysis methods, like community structure search.

In the introduction of *igraph* in [47], one can observe the following assertion: *"The efficient implementation of igraph allows it to handle graphs with millions of vertices and edges. The rule of thumb is that if your graph fits into the physical memory then igraph can handle it."* Therefore, it is clear that *igraph* is a useful tool to take into account when it is mandatory to work with graphs.

The current version of *igraph*, at the moment of writing this report, is *0.5.4*. It can be installed in several forms:

- as a C library.
- as an R package.
- as a Python extension module.
- as a Ruby extension.

In *igraph*'s website [47] a detailed *Documentation* section can be found. In my opinion, there are some algorithms that could be explained in more detail, what could help to understand what kind of implementation did the

authors choose. Moreover, a *Mailing list* section is also available, as well as *Wiki* one, which provides some tutorials.

In this project, *igraph* as an R package has been chosen, and the topologies analyzed in the Case Study 5 have been generated using it.

A.2 How to install *igraph* as an R package

It is possible to use *igraph* as an extension package to *The GNU R project for Statistical Computing*. The flexibility of the R language and its richness in statistical methods add a great deal of productivity to *igraph*, with a very small speed penalty.

First of all, it is necessary to download and install R from [54]. Then, the simplest way to install the *igraph* R package is typing the following command in your R session.

```
> install.packages("igraph")
```

For this project, the R MAC OS X version has been used.

A.3 How to use igraph R package

Once everything has been installed, the following step is to add the *igraph* package to R, as shown in Fig. A.1 and Fig. A.2.

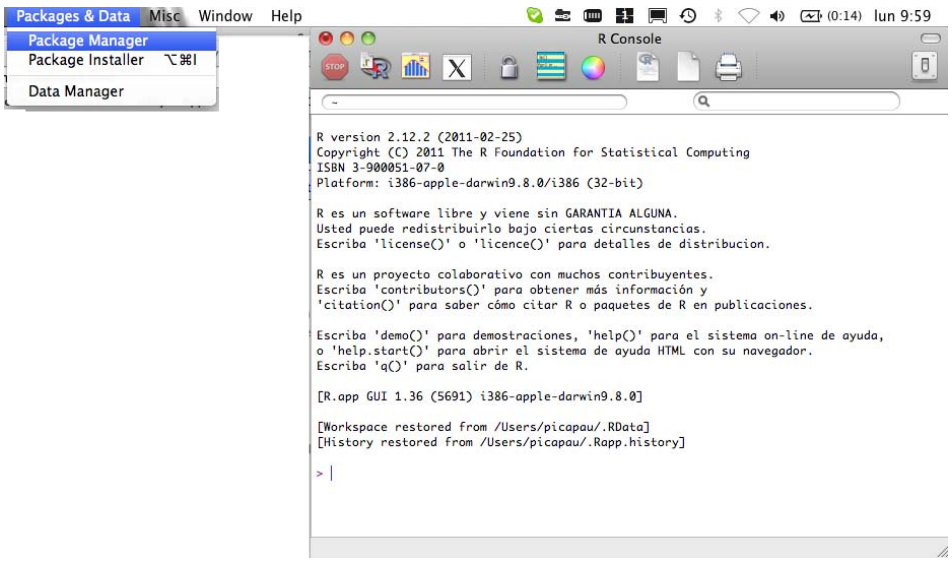

Figure A.1: Adding *igraph R package* (1).

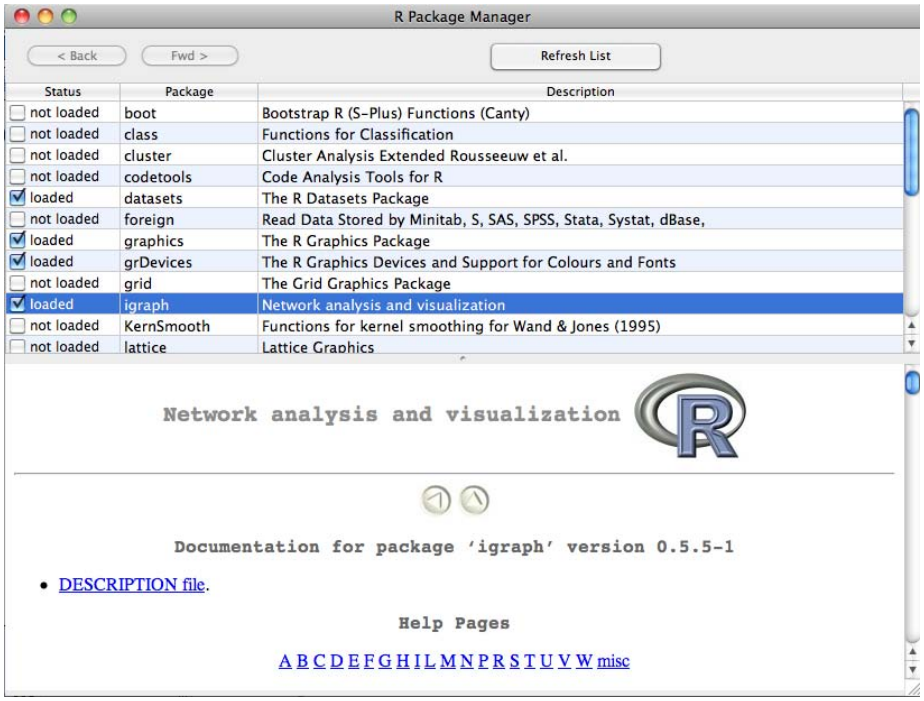

Figure A.2: Adding *igraph R package* (2).

A.3.1 Generating graphs

In order to generate a random graph, the following command must be written on the R's command-line prompt:

```
> g <- erdos.renyi.game(10, 1/5)
```

The first parameter is the number of vertices of the graph, while the second one is either the probability for drawing an edge between two arbitrary vertices or the number of edges in the graph. *erdos.renyi.game(n, p or m)* generates random graphs according to the Erdős–Rényi model [48]. By default undirected graphs are generated.

To generate a small-world graph, the following command is used:

```
> g <- watts.strogatz.game(1, 10, 2, 0.05)
```

The second parameter indicates de number of nodes, while the third parameter indicates the neighborhood degree of nodes. *watts.strogatz.game(dim, size, nei, p)* generates small-world graphs according to the Watts and Strogatz model [49]. By default undirected graphs are generated.

Finally, to generate a scale-free network, the following command is used:

```
> g <- barabasi.game(1000, 1,  directed=FALSE)
```

The first parameter is the number of vertices, while the second one is the power of preferential attachment. *barabasi.game(n, power = 1, directed = TRUE)* generate scale-free graphs according to the Barabási–Albert (BA) model [50]. By default directed graphs are generated.

A.3.2 Working with graphs

igraph incorporates a wide battery of functions in order to work with graphs. Some useful commands are commented below.

The following commands calculate the *average nodal degree*:

```
> d<-c(1:400)
> for (i in 0:399){ d[i+1]<-degree(g,i)}
> sum(d)/400
```

Where 400 is the number of nodes of the graph. To plot the *degree distribution*:

```
> plot(degree.distribution(g), type="o", col="blue")
```

Moreover, the following command calculates the *average shortest path length*:

```
> average.path.length(g)
```

Then, graphs generated with *watts.strogatz.game* or *barabasi.game* may have loops or multiple edges. The following command must be applied in order to simplify them:

```
> g <- simplify(g)
```

Also, graphs generated with *erdos.renyi.game* may be disconnected. The following commands help to check it:

```
> is.connected(g)
[1] TRUE
> no.clusters(g)
[1] 1
```

The first command returns a boolean indicating if the graph is disconnected. The second one, the number of islands or parts of the graph (1 c̄onnected).

In order to know the power of the Power Law distribution that a specific scale-free topology has, the following commands must be used:

```
> d <-degree(g)
> power.law.fit(d+1,xmin=2)
```

Finally, it is possible to read graphs from files. The next example shows the command that must be used in order to read a graph in a *graphml* format.

```
> g <- read.graph(file="PATH/sf400d4.graphml",format="graphml")
```

A.4 Plotting graphs

igraph offers a wide variety of layouts in order to plot a graph. The command used must be the following one:

```
> plot(g,vertex.size=1,vertex.label="",layout=layout.svd)
```

The parameter *layout* is the one that must be modified in order to plot the graph in different ways. For example Fig. A.3 shows the layout of *layout.svd*, while Fig. A.4 shows the layout of *layout.fruchterman.reingold*.

Figure A.3: Plotting a graph with the layout of *layout.svd*.

Figure A.4: Plotting a graph with the layout of *layout.fruchterman.reingold*.