Project no.:	**IST-FP6-STREP - 027513**
Project full title:	**Critical Utility InfrastructurAL Resilience**
Project Acronym:	**CRUTIAL**
Start date of the project:	**01/01/2006 Duration: 39 months**
Deliverable no.:	**D19**
Title of the deliverable:	**Model-based evaluation of the middleware services and protocols & architectural patterns**

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Contractual Date of Delivery to the CEC:	31/03/2009
Actual Date of Delivery to the CEC:	16/03/2009
Organisation name of lead contractor for this deliverable	CNIT

Editor(s): Susanna Donatelli[6]
Author(s): Susanna Donatelli[6]; Eric Alata[4]; Andrea Bondavalli[3]; Marco Beccuti[6]; Davide Cerotti[6]; Silvano Chiaradonna[3]; Alessandro Daidone[3]; Giovanna Dondossola[1]; Felicita Di Giandomenico[3]; Giuliana Franceschinis[6]; Fabrizio Garrone[1]; Oussama Hamouda[4]; Mohamed Kaaniche[4]; Paolo Lollini[3]; Vincent Nicomette[4];
Participant(s): (1) CESI-R (3); CNR-ISTI; (4) LAAS-CNRS; (6) CNIT.

Work package contributing to the deliverable:	WP5
Nature:	R
Dissemination level:	PU
Version:	004
Total number of pages:	89

Abstract

This deliverable is the final report on the model-based evaluation and validation activity in CRUTIAL. It builds on the interim-report deliverable D25. Please note that the material included goes beyond the scope identified by the deliverable title, since it includes all the model-based evaluations that were included in the goals of WP5. Indeed this deliverable contains the model based evaluation of the architectural solutions, but also an evaluation of the interdependencies between the Electrical and the Information Infrastructures through the application of (part of) the CRUTIAL modelling framework devised in WP2, as well as a model based characterization of the attacks, built on the data that the project partners have gathered through honeypot.

Contents

List of Figures

1 Introduction

This deliverable reports on a number of evaluation and validation activities that have been carried on in the project through the use of models using the formalisms, methodologies and tools introduced in deliverables D8 [Kaniche et al. 2008] and D11[Donatelli et al. 2008b]. They are quite diverse activities, but highly representative of the various roles that model-based approaches can play in the analysis of critical infrastructure. In particular this deliverable concentrate on the modelling of

- attacks to the ICT
- the architectural solutions devised by the project, and
- the (inter)dependencies between EI and II.

Chapter 2 reports on the work that has been done to estimate, in a statistically proper manner, the time between attacks. The result of this work is a set of distribution estimated from the time series registrated by a set of deployed honeypots as described in D20. Both low-level interactions and high-level interaction honeypots have been considered. The fitting of distributions has required a careful and non trivial polishing of the collected data.

Chapter 3 reports instead on the work that has ben conducted to validate and evaluate CIS-PS (the CIS protection service) Section 3.1 describes the SWN models of the basic CIS-PS (called CIS-PS-IT - Intrusion Tolerant): the main objective of the analysis, conducted with the GreatSPN tool, has been the validation of the CIS algorithm. The model construction and analysis has enlighted some delicate points to be carefully considered in the implementation. No performance analysis has been performed on this basic model (although the SWN are suitable for a quantitative evaluation), which is instead extensively attacked in Section 3.2 that deals with the more complex CIS-PS-SH (Self Healing). Deterministic Stochastic Petri Nets models have been constructued and analyzed through the tool DEEM. The analysis has allowed a throughout comparison of various strategies (including some variation of the one actually implemented in CIS) for varying sets of input parameters that take into account the CIS components failure rate.

Chapter 4 exercise the SAN models devised in the CRUTIAL modelling framework of D16 onto some reference electrical grids defined in the literature, experimenting how the EPS react, and eventually loose some of its ability to supply power, upon an electrical failure that cannot be dealt with correctly by the ICT control since it is under a DoS attack that makes a number of substations unreachable. The model is highly parametric and thus constitute a flexible way to evaluate the EPS behaviour under various (electrical and ICT) failure scenarios.

Chapter 5 introduces first a detailed model of scenario 2, the same scenario that has been implemented in the testbed of the teleoperation in WP3. The analysis of the model reveals the need for an interaction with the SAN model, since it is only in the SAN model that the effect on the electrical state can be evaluated realistically. The interaction between the two models is then identified, with the SWN model contributing in having a more faithful representation of the scenario and with the SAN model contributing in having a faithful representation of the electrical state evolution. This interaction has been applied on a case study grid to obtain a first set of results on the EPS behaviour in the case of scenario 2.

2 Statistical Models of Attacks

This section deals with the elaboration of statistical models that are representative of malicious traffic observed on the Internet using data collected from honeypots. Such models are very relevant to establish realistic assumptions about the distribution and the intensity of Internet attacks targeting systems and infrastructures connected through the Internet, as in CRUTIAL. Also, they constitute a first step towards the elaboration of stochastic models aimed at evaluating quantitative measures characterizing the impact of malicious threats on the target systems, as discussed in the context of CRUTIAL modelling methodology addressed in Workpackage 2 and presented in deliverable D16 [20].

As detailed in deliverable D20 [16], honeypots have been increasingly used in the recent years to collect real data about malicious traffic on the Internet. A honeypot is a machine connected to the Internet that no one is supposed to use and whose value lies in being probed, attacked or compromised . The statistical models of attacks discussed in this section are based on the data collected from two types of honeypots:

1. Low interaction honeypots deployed in the context of the Leurr.com environment, which is a cooperative attack data collection initiative set up by Eurecom to which LAAS contributes, based on distributed honeypot platforms. This environment integrates up to eighty identically configured low-interaction honeypots that have been deployed progressively since 2003.
2. The high-interaction honeypot developed and deployed by LAAS to analyse the activities performed by attackers once they gain access to a victim and try to progress in their intrusion process.

2.1 *Attack models based on low interaction honeypot data*

Deploying honeypots at a distributed and large scale is interesting to collect a large volume of data characterizing malicious activities observed at various locations of the Internet. One of the questions that can be raised is whether data collected by honeypots deployed at different locations exhibit similar or different phenomena and whether the attack processes observed show different or similar statistical distributions.

The results presented in this section are aimed at addressing these questions. The objective is to elaborate analytical statistical models that faithfully reflect the distribution of the interarrival time between attacks observed at various honeypot platforms. Such models provide useful insights about the statistical characteristics of malicious traffic observed on the Internet. They can be used to generate synthetic workloads that are representative of malicious traffic. The statistical distributions presented in this section are also useful to support the definition of quantitative evaluation models based on realistic assumptions.

This section is organized as follows. Section 2.1.1 gives an overview of the collected data and of some of the problems that need to be addressed to exploit the data for building models. Section 2.1.2 presents the proposed methodology. Section 2.1.3 deals with the statistical modelling of the times between attacks based on the data collected from the deployed honeypots and presents some examples of results.

2.1.1 Overview of the Leurre.com environment and the collected data

The Leurre.com data collection environment is aimed at deploying at various geographical locations on the Internet a large set of identically configured low interaction honeypot platforms using the freely available software called honeyd. The objective is to collect a large volume of data that can be used to carry out representative and non biased analyses of attack processes. Each platform emulates three computers running Linux RedHat, Windows 98 and Windows NT, respectively, and various services such as ftp, web, etc. A firewall ensures that connections cannot be initiated from the computers, only replies to external solicitations are allowed. All the honeypot platforms are centrally managed to ensure that they have exactly the same configuration. The data gathered by each platform are securely uploaded to a centralized database with the complete content, including payload of all packets sent to or from these honeypots, and additional information to facilitate its analysis, such as the IP geographical localization of packets' source addresses, the OS of the attacking machine, the local time of the source, etc.

The data recorded in the database can be analyzed at various levels of granularities. Indeed, the packets received at each platform can be grouped e.g. according to the source address, the target virtual machine, the time between the arrival of consecutive packets received from the same source, etc.

The concepts of "source" and "attack" used in this section are defined as follows:

- A source corresponds to an IP address observed on one or many platforms, for which the inter-arrival time between two consecutive packets does not exceed a given threshold (25 hours). The time difference is computed by converting all times to GMT (Greenwich Mean Time).
- An attack is composed by the set of packets exchanged between a source and a particular honeypot platform.

The deployment of the honeypots has been carried out progressively starting in 2003. To date, up to 80 honeypot platforms have been deployed at various locations in academia and industry, in 30 countries, covering the five continents. The total number of attacks recorded in the Leurre.com database between February 2003 and August 2007 is 4 873 564 attacks issued from 3 026 972 different IP addresses. This constitutes a significantly large sample on which statistical analyses can be performed.

Table 2.1 gives some statistics summarizing the number of attacks observed on each

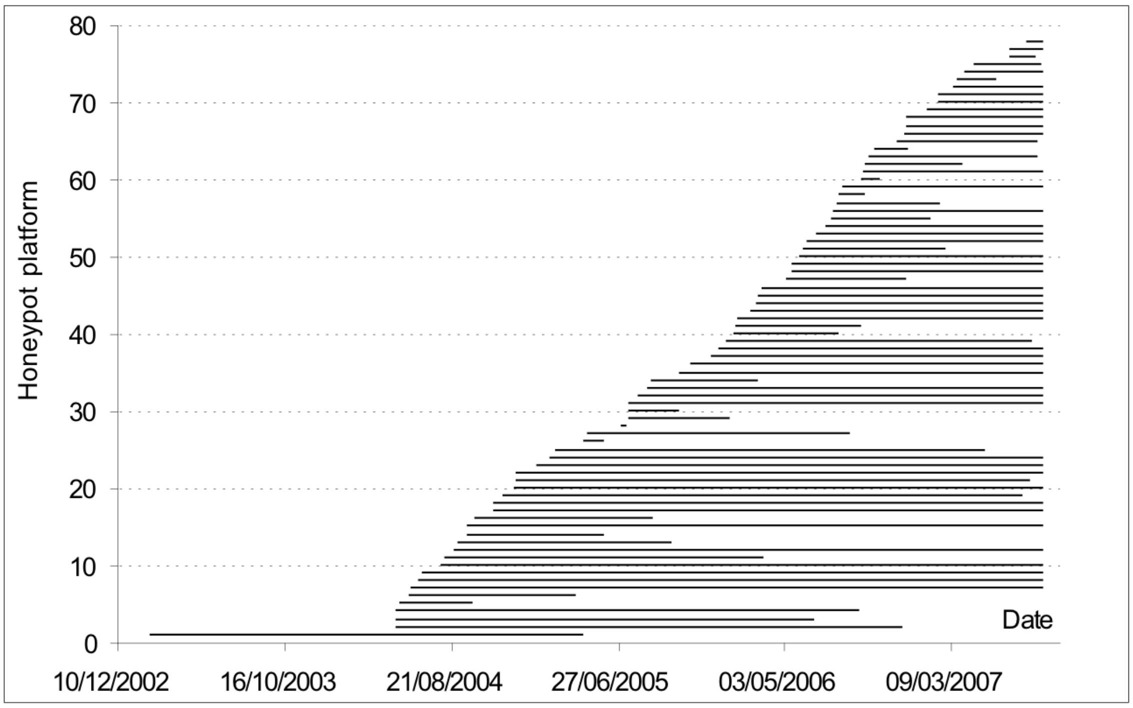


Figure 2.1: Number of deployed honeypots evolution.

platform. It can be seen that the level of malicious activity recorded on the different platforms was not uniform. This can be explained to some extent by the fact that the platforms have been deployed progressively as illustrated in Figure 2.1.

Min	Max	Average	Median	Std. deviation
3	504651	62480.59	39594.5	81140.93

Table 2.1: Statistics on the number of attacks recorded on the honeypot platforms

Figure 2.1 gives an overview of the deployment where each bar associated with a given platform indicates the time interval between the first packet and the last packet recorded on the platform. As can be seen from the figure, the observation period of the different platforms was not uniform. Some of them have been operational only for a short period of time, compared to others for which we have data covering 4 years. It is important when performing comparative analysis of attack processes observed on several platforms that all the platforms have been observed during the same sufficiently long period of time.

Considering Figure 2.1 again, the observation period indicated by the bar associated with each platform does not mean that the platform was active all the time during this period. Indeed, for several reasons, some of the honeypots exhibited many times, silence periods during which no activity was recorded. These silence periods are more likely due to the unavailability or the unreachability of the honeypot from the Internet as a consequence of power failures, network failures or simply due to the disconnection of the honeypot itself for administration and maintenance activities. Two examples are presented in Figure 2.2 and Figure 2.3 which plot the evolution of the number of attacks per day recorded on

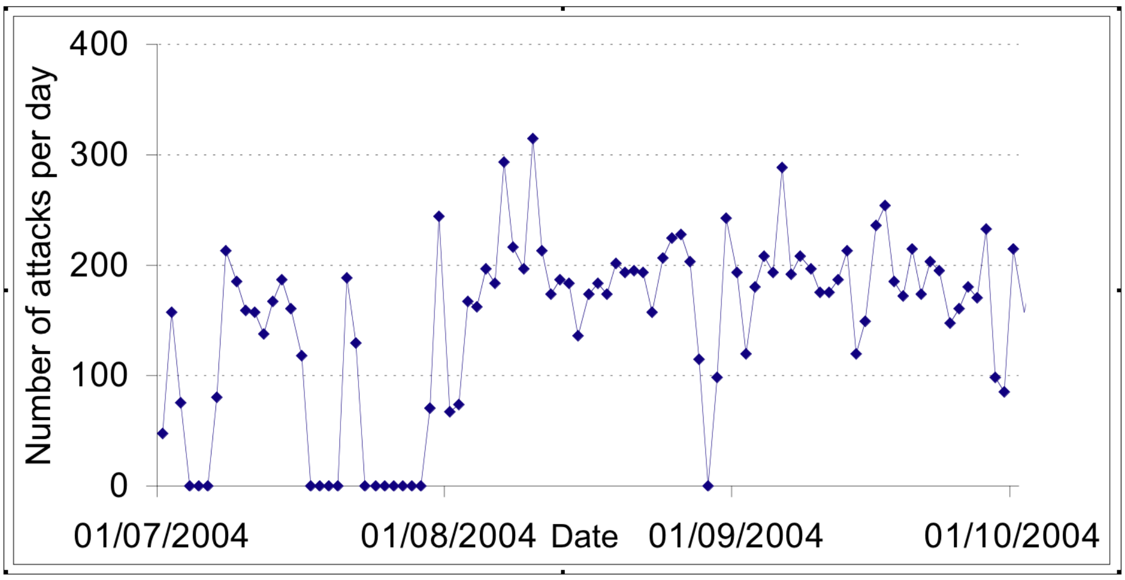


Figure 2.2: Evolution of the number of attacks per day observed on platform 9.

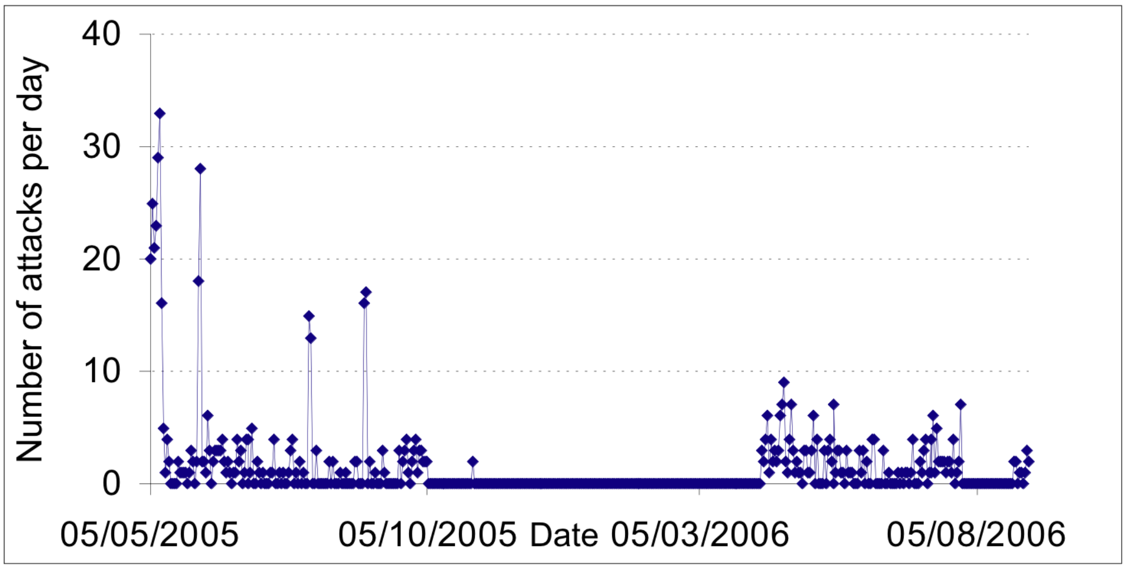


Figure 2.3: Evolution of the number of attacks per day observed on platform 37.

Platforms 9 and 37 respectively.

Considering the intensity of attacks observed on average per day on each platform, it is more likely that the silence periods are more related to unreachability problems than to the absence of activities from the attackers. Thus, it is important to identify and process such periods before building models characterizing the occurrence of attacks, otherwise the results will be biased. In the following section, we present the methodology that we have developed to address this problem.

2.1.2 Methodology

The methodology that we have set up to deal with silence periods consists of two main steps:

1. identification of the silence periods,
2. selection of the data observation period and the platforms to be included in the modeling of the distribution of times between attacks based on the results obtained in step 1.

2.1.2.1 Identification of silence periods

In our data, the silence periods generally correspond to atypical and infrequent intervals of time between attacks that are significantly separated in value from the rest of the other observations recorded on the honeypot platform. Accordingly, they can be considered as ”outliers”.

Various statistical tests exist for the identification of outliers, e.g., Nixon, Grubbs or boxplot tests . In our methodology, we used the modified boxplot test defined in , which is well suited when the distribution of the data is skewed, which is the case of our honeypot data. This test proceeds in two steps.

At a first step, this test computes for the considered data set D a metric denoted as `MC(D)`, taking values in the interval $[-1, 1]$, that measures the skewness of the distribution. Positive (respectively, negative) values correspond to positively (respectively, negatively) skewed distributions, and when `MC(D)` is null the distribution is not skewed.

At a second step, the test computes a critical interval that depends on the sign of the skewness metric `MC(D)`, such that any value outside this interval is considered as an outlier.

Let us denote by `Q1, Q2`, and `Q3` the first, second and third quartiles of the considered data sample D, and let `IQR = Q3-Q1`. The test identifies outliers as follows:

$$\begin{gathered}\text{Si } MC(D) \geq 0, \text{ and}\\ x \notin [\mathcal{Q}_1 - 1,5 \cdot e^{-4 \cdot MC(D)} \cdot IQR; \mathcal{Q}_3 + 1,5 \cdot e^{3 \cdot MC(D)} \cdot IQR]\\ \text{Si } MC(D) < 0, \text{ and}\\ x \notin [\mathcal{Q}_1 - 1,5 \cdot e^{-3 \cdot MC(D)} \cdot IQR; \mathcal{Q}_3 + 1,5 \cdot e^{4 \cdot MC(D)} \cdot IQR]\end{gathered}$$

We have applied this test to the data collected from each honeypot platform. The percentage of identified outliers for each platform is generally less than 1%. However, we have observed a large variation of the magnitude of the intervals of time considered as outliers. The average value is around 6 hours and the standard deviation is about 78 hours, considering the values of the outliers identified for the 80 platforms.

2.1.2.2 Data selection for the modelling of inter-arrival times between attacks

The outliers identified in the first step of our methodology correspond to suspicious periods of silence. In our context, we make the assumption that they most likely correspond to unavailability periods of the corresponding platform, than to periods of deliberate inactivity of the attackers.

Then, the question is: what should we do with these outliers? Usually, two solutions are investigated:

1. Remove the outliers from the data set or substitute them by synthetic values generated based on the general characteristics of the sample distribution.
2. Select a subset of the initial data such that the impact of the outliers is reduced.

The first solution is not acceptable in our context as it might lead to biased results. Also it makes the comparison of the attack processes observed on different platforms considering the same period of time, more difficult in particular, when the outliers correspond to long periods of time. Thus, the second solution is more suitable to our context.

In our methodology, we have considered three main criteria to select the period of time and the subset of data to be used for the statistical modelling of the times between attacks on the different honeypot platforms.

1. The length of the observation period.
2. The number of platforms included in the analysis.
3. The minimum level of average availability estimated for each platform.

The average availability of each platform is estimated based on the assumption that the silence periods correspond to unavailability periods as explained in the beginning of this section.

We have developed an iterative algorithm based on a sliding window that starts first by considering the whole data collection period and estimates the availability of each platform. If the number of platforms satisfying the minimum availability per platform criterion is higher than a predefined threshold, the algorithm stops. Otherwise, we consider a shorter period of 1 hour less and run the algorithm again until it converges.

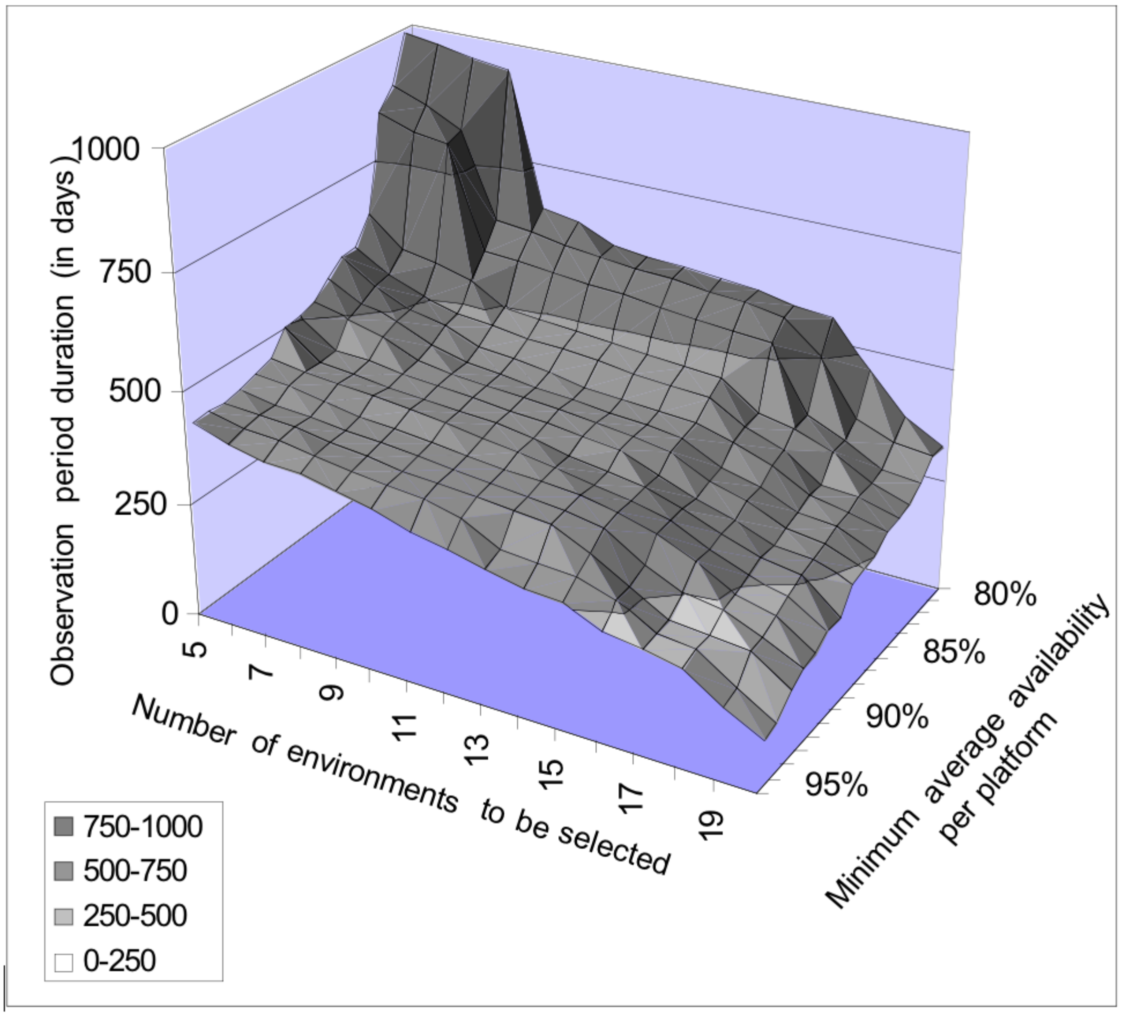


Figure 2.4: Relationship between the duration of the selected period, the minimum average availability per platform and the number of platforms satisfying the a vailability requirement.

Figure 2.4 presents graphically the results obtained from the algorithm. Some numerical examples extracted from the figure are reported in Table 2.2 . As expected, if one sets a predefined number of platforms to be selected, increasing the minimum availability requirement to be satisfied by each platform, will lead to a shorter observation period, and vice-versa.

		Number of selected platforms		
		8	**15**	**20**
Minimum availability per platform	**80%**	637	448	420
	85%	490	413	343
	90%	455	350	259
	95%	287	189	89

Table 2.2: Examples of results extracted from Figure 1-4

In our study, we have set as an objective to have the longest possible observation period with a reasonable number of platforms to enable comparative analyses of attack processes observed on various platforms. Accordingly, we have selected 8 platforms with a minimum availability requirement of 80% corresponding to an observation period of 637 days. The number of platforms selected is sufficient to make significant comparative analyses.

Table 2.3 reports some statistics characterizing the activities observed on the selected platforms. The first column identifies the platform, the second column gives the number of intervals between attacks `(#ti)` observed for this platform. The following columns indicate the values of `Q1`, `Q2` and `Q3` quartiles, the maximum, the mean, and the standard deviation of the interarrival times between attacks. Finally, the last column gives the average availability of the corresponding platform. These platforms are geographically located in six different European countries: France, Italy, Belgium, Poland, Germany and UK.

Honeypot	#ti	Q1(ti) (sec)	Q2(ti) (min)	Q3(ti) (min)	Max (ti) (min)	Mean (ti) (min)	Std. Deviation (min)	Average Availability (%)
9	134161	56	3	8	53	6	7	93
13	15742	538	32	73	382	52	59	9
14	42670	107	7	24	158	18	25	85
28	10200	578	35	96	650	73	100	82
31	90580	76	4	11	52	8	9	81
32	65962	161	7	16	76	11	12	84
42	38826	102	7	25	278	19	29	84
62	25042	435	19	40	199	29	30	80

Table 2.3: Statistics on the activities observed on the selected platforms during the observation period of 637 days

2.1.3 Time between attacks distribution

Considering the selected platforms and the data collected during the selected period of time identified by the algorithm presented in the previous section, we have investigated candidate statistical models that are representative of the distribution of times between attacks observed on the different platforms.

Finding tractable analytical models that faithfully reflect the observed times between attacks is useful to characterize the observed attack processes and to find appropriate indicators that can be used for prediction purposes. We have investigated several candidate distributions, including Weibull, Lognormal, Pareto, and the Exponential distribution, which are traditionally used in reliability studies. The best fit for each platform has been obtained using a mixture model combining a generalized Pareto and a Weibull distribution.

Let us denote by `T` the random variable corresponding to the time between the occurrence of two consecutive attacks at a given platform, and `t` a realization of `T`. Assuming that the probability density function `f(t)` asssociated to `T` is characterized by a mixture distribution combining a generalized Pareto distribution and a Weibull distribution, then `f(t)` is defined as follows.

$$f(t) = \Pi \cdot \frac{1}{\sigma} \cdot (1 - \frac{\epsilon t}{\sigma})^{\frac{1}{\epsilon}-1} + (1 - \Pi) \cdot \frac{k}{\lambda} \cdot (\frac{t}{\lambda})^{k-1} \cdot e^{-(\frac{t}{\lambda})^k}$$

σ and ϵ the parameters of the generalized Pareto distribution, k and λ are the parameters associated to the Weibull distribution and Π is a mixture probability.

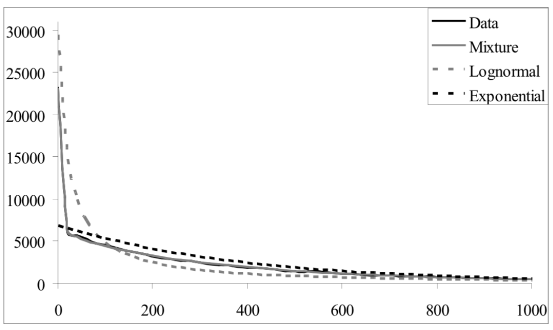
We have used the R statistical package to estimate the parameters that provide the best fit to the collected data. The quality of fit is assessed by applying the Kolmogorov-Smirnov and the Chi-Squared statistical tests. The results obtained for four of the eight platforms are presented in Figure 1-5. Similar conclusions have been observed for the other platforms as well. It can be noticed that for all the platforms, the mixed distribution provides a good fit to the observed data whereas the exponential and lognormal distributions are not suitable to describe the observed attack processes. Thus, the traditional assumption considered in reliability evaluation studies assuming that failures occur according to a Poisson process does not seem to be satisfactory when considering the data observed from our honeypots.

As regards the interpretation of the mixture distribution, the Pareto part models the bursty arrival of attacks (correlated and intensive attacks targeting one IP address) whereas the Weibull part describes background uncorrelated attacks that occur less frequently in time. It is noteworthy that this result confirms the preliminary investigations derived in [19] based on a small subset of the data presented in this section.

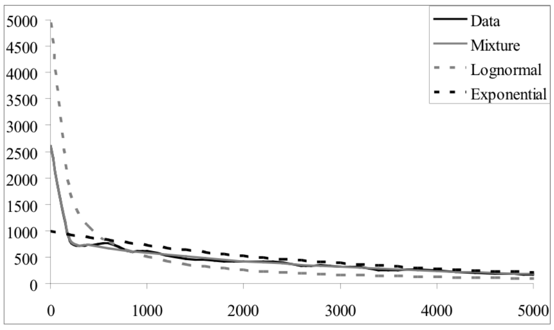
2.2 Attack models based on high interaction honeypot data

High-interaction honeypots are complementary to low-interaction honeypots as they allow the observation of attackers behaviour once they manage to compromise a victim and try to progress in their intrusion, while this is not possible with the latter. Thus it is also relevant to study the statistical distribution of attacks based on real data collected from such honeypots and analyse how they compare with the distributions presented in Section 2.1. This is the aim of this section considering the attack data collected from the high-interaction honeypot deployed in the context of project, that is described in Deliverable D20 [16].

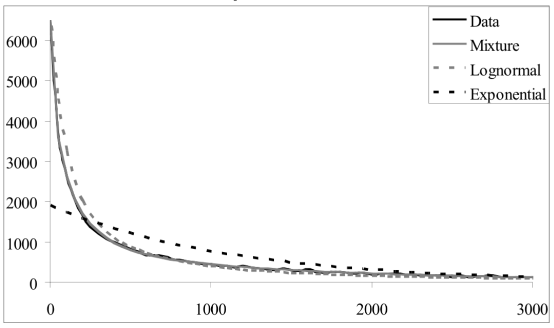
This section is organized as follows. Section 2.2.1 gives an overview of the high-interaction honeypot and the collected data and Section 2.2.2 deals with the statistical modelling of the times between attacks based on the data collected from the honeypot.



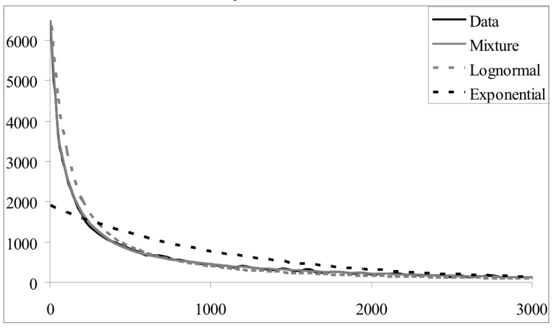
(a) environnement 9



(b) environnement 13



(c) environnement 14



(d) environnement 28

Figure 2.5: Observed and estimated times between attacks probability density functions for four low interaction honeypot platforms.

2.2.1 Overview of the high-interaction honeypot and the collected data

The deployed configuration is based on VmWare and the GNU/Linux operating system. It includes three virtual machines; only two of them are directly accessible from the Internet by the attackers (see Figure 2.6)).The experiments that we have carried out focus on the attacks performed via the SSH service. Each virtual machine has been set up initially with 17 user accounts, with only port 22 corresponding to the SSH service open.

The honeypot has been deployed during more than one year (419 days). The data collected include: 1) the pairs (username and password) tested by the attackers to gain access to the system; 2) the data exchanged inside the SSH connection, and 3) the system calls generated by the activity of the attacker.

The analysis of the attack processes observed on the honeypot requires the development of a rigorous methodology allowing the identification of attack sessions and their characteristics from the captured raw data. An attack session corresponds to a sequence of SSH connections in a short time, carried out by the same IP address and targeting the same virtual machine. As described in [2], the identification of attack sessions can be done using a sliding window algorithm and considering a time threshold defining the maximum duration separating the reception of two consecutive packets belonging to the same session.

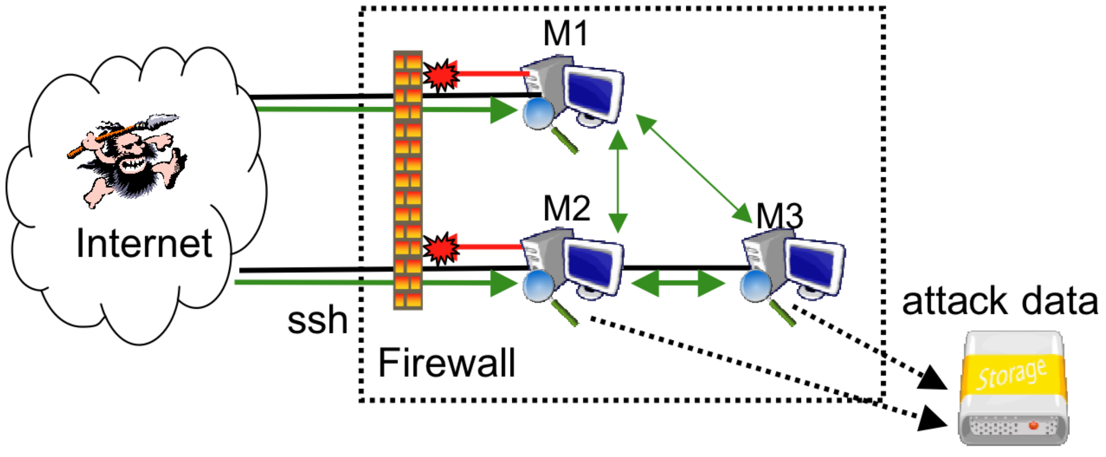


Figure 2.6: Simplified view of the deployed honeypot architecture.

During the experiment, 552 362 connection attempts have been recorded. The application of the clustering algorithm led to the identification of 1940 attack sessions. For each user account, two distinct steps have be observed in the attack process targeting the corresponding account:

1. The first step, generally carried out by means of automatic tools, concerns brute-force dictionary attacks aimed at guessing the valid user account to gain access to the system.
2. The second step corresponds to the activities performed by the attackers, once the valid user account has been found.

In the following sections, the attacks corresponding to the first step are called "dictionary attacks" and those corresponding to the second step are called "Intrusions".

The methodology that allowed us to identify these two categories of attacks is summarized in deliverable D20 and detailed in [2, 24]. In particular, we have observed that the IP addresses associated to the dictionary attacks and intrusions do not overlap. Thus, it is likely that these attacks are carried out by different communities, using different sets of machines.

2.2.2 Data selection for the modelling of times between attacks

In the following, we focus on the statistical modelling of the distributions characterizing the times between attacks observed on each of the virtual machines connected to the Internet: M1 and M2. For each of them, we first consider the set composed of all the dictionary attacks and the intrusions observed, then we analyze the distribution characterizing the times between intrusions only. Table 2.4 presents global statistics characterizing the times between attacks observed on M1 and M2 corresponding to each of these cases.

		Number of Attacks	Min	Max	Average	Median	Std. deviation
M1	All	883	0 sec	339771 sec (94.4 hrs)	33201 sec (9.2 hrs)	11153 sec (3.1 hrs)	49039 sec (12.6 hrs)
	Intrusions	152	2 sec	1701296 sec (472.6 hrs)	131861 sec (36.6 hrs)	17434 sec (4.8 hrs)	285956 sec (79.4 hrs)
M2	All	567	0 sec	321832 sec (89.4 hrs)	32377 sec (9.0 hrs)	10920 sec (3.0 hrs)	46456 sec (12.9 hrs)
	Intrusions	51	35 sec	4162909 sec (1156 hrs)	50044 sec (13.9 hrs)	32330 sec (9.0 hrs)	56969 sec (15.8 hrs)

Table 2.4: Statistics on the times between attacks recorded on the honeypot

As performed in Section 2.1.3, we have investigated candidate statistical models that are representative of the distribution of times between attacks observed on M1 and M2, considering first all attacks and then intrusions only.

We have investigated several candidate distributions, including Weibull, Lognormal, Pareto, Beta, Generalized Gamma, the Exponential distribution, etc.. We have used the EasyFit statistical package to estimate the parameters of the distributions. The quality of fit is assessed by applying the Kolmogorov-Smirnov, and the Chi-Squared statistical tests.

For both M1 and M2 data sets, we have observed that the Beta distribution faithfully describes the distribution of the times between attacks when considering all attacks, while the Generalized Gamma distribution provides better results when considering the intrusions only.

The probability density function of the Beta and Generalized Gamma distributions are defined as follows:

Beta distribution:

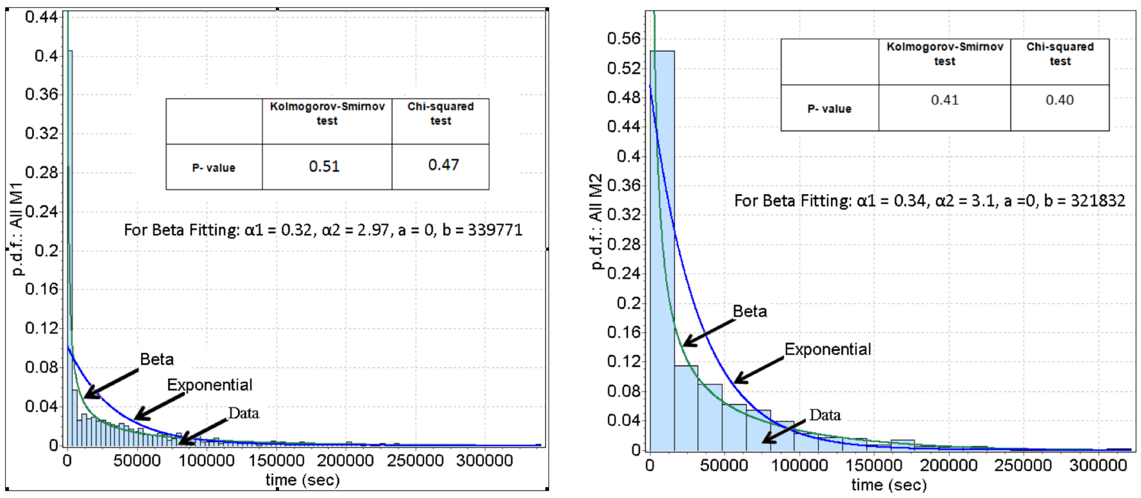
$$pdf(t) = \frac{1}{B(\alpha_1, \alpha_2)} \cdot \frac{(t-a)^{\alpha_1 - 1}(b-t)^{\alpha_2 - 1}}{(b-a)^{\alpha_1 + \alpha_2 - 1}}$$

Generalized Gamma distribution:

$$pdf(t) = \frac{k(t-\gamma)^{k\alpha - 1}}{\beta^{k\alpha}\Gamma(\alpha)} exp(-(t-\gamma)/\beta)^k$$

Figure 2.7 and Figure 2.8 plot the observed and the estimated probability density functions considering all attacks and intrusions, respectively. Also, the p-values corresponding to the Kolmogorov-Smirnov, and the Chi-Squared statistical tests are provided for the

Beta and generalized gamma distributions. The p-values show that the corresponding distributions provide a good fit considering a 5% significance level for the tests. For the sake of comparison, we also plot the estimated pdf when assuming an exponential distribution. This distribution is rejected by the Kolmogorov-Smirnov, and the Chi-Squared statistical tests for all the cases presented in Figure 2.7 and Figure 2.8. It is noteworthy also that the mixed distribution that proved to be well suited to describe the distribution of the times between attacks for the data collected from the low interaction honeypots was not suitable for describing the data collected from the high interaction honeypots.



	Kolmogorov-Smirnov test	Chi-squared test
P- value	0.51	0.47

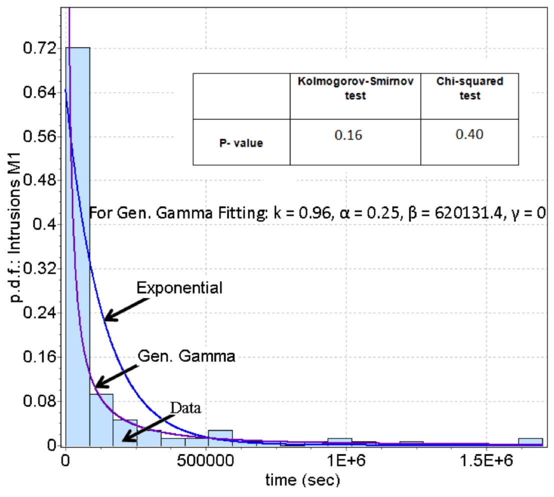
	Kolmogorov-Smirnov test	Chi-squared test
P- value	0.41	0.40

(a) M1 (b) M2

Figure 2.7: Observed and estimated times between attacks probability density functions for M1 and M2 considering All attacks

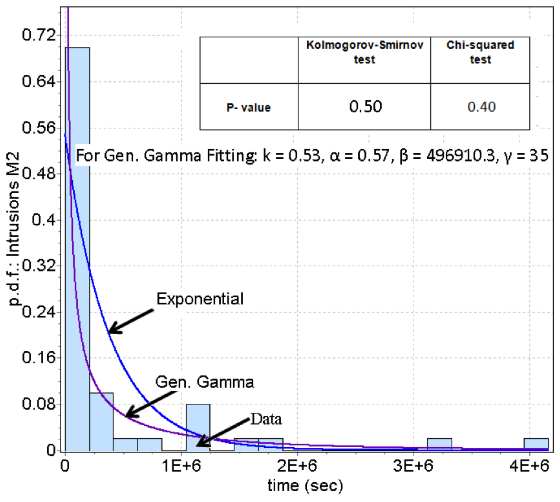
2.3 Discussion

This section focussed on the identification of statistical distributions that are representative of the attack processes observed on two types of honeypots providing different levels of interaction with the attackers. Such distributions are needed to support the elaboration of quantitative security assessment models that are based on realistic assumptions. The analysis of the data collected from a large set of identically distributed low interaction honeypots revealed that a mixture Pareto and Weibull distribution is well suited to describe the attack processes observed on several honeypot platforms. However, this distribution was not suitable for describing the times between attacks observed on the high interaction honeypots. For the latter, a beta and generalized gamma distributions have proved to be more suitable. This result can be explained by the fact that the attack activities recorded by the two types of honeypots are different. In particular, the high interaction honeypot has been designed to observe manual attacks and intrusions, that are less frequent than the automatic attacks usually captured by low interaction honeypots. Clearly, the results presented in this section provide preliminary indications about the rate of occurrence of attacks



	Kolmogorov-Smirnov test	Chi-squared test
P- value	0.16	0.40

(a) M1



	Kolmogorov-Smirnov test	Chi-squared test
P- value	0.50	0.40

(b) M2

Figure 2.8: Observed and estimated times between attacks probability density functions for M1 and M2 considering Intrusions only

and the type of distribution that can be used to reflect the observed behavior. However, more data would be needed to generalize these results, by considering the deployment of the honeypot in other locations and more importantly by extending the honeypot to cover other types of attacks, e.g. DoS attacks, web-based attacks, etc.

3 Evaluation of Architectural Solutions

In this chapter we propose the evaluation of the CIS solution(s) devised in Workpackage 4. The first approach is reviewed in Section 3.1: it is based on an SWN analysis of the basic CIS algorithm using the GreatSPN tool [11]. This analysis has pointed out some delicate points and difficulties in the implementation of the CIS, problems that have been solved in the enriched version of CIS that includes proactive and reactive recovery strategy. The performance of CIS under the various recovery strategy has been conducted using Deterministic Stochastic Petri Nets under the tool DEEM [8], and it is reported in Section 3.2.

3.1 SWN models of basic CIS

This section describes how (Stochastic) Well-formed Nets (WN) [18] have been used to model CIS and to analyze its correctness. The model is non trivial, and therefore asignificant amount of effort has been devoted to build the model. We have used compositionality and colour limitation to validate the model incrementally, and, in general, to get a deeper understanding in the system. While validating the model we have found a few "delicate" points in the design that had to be carefully taken into account in the implementation.

The contributionn is organized as follows. Section 3.1.1 reviews the CIS algorithm that we have considered, we then proceed in Section 3.1.2 with the description of the model (model components and their composition). Section 3.1.3 reports on the process that has been followed for system validation.

3.1.1 The system under study

The Crutial Information Switch(CIS) device realizes the protection of LANs from the WAN or from other LANs, acting like a firewall: it captures packets that pass through it, checks if they satisfy the security policy and either forwards the packets or discards those that do not satisfy the policy; messages that satisfy the policy are defined as *legal*, *illegal* otherwise. In CRUTIAL the access policy is supported by the OrBAC environment [1], a role-based system which defines the rules for information exchange and collaboration between different facilities of the critical infrastructures.

The CIS system implements a distributed replicated firewall between a non trusted WAN and the trusted LAN that we want to protect, as shown in Figure 3.1. To increase the resilience of the system to malicious faults and accidental faults each CIS is replicated, and we say that CIS is composed of n CIS replicas. A replication device located at the end of the WAN multicasts the incoming messages to the n CIS replicas. Each CIS replica verifies whether the message is in accordance with the predefined security policy, if it is the case, it

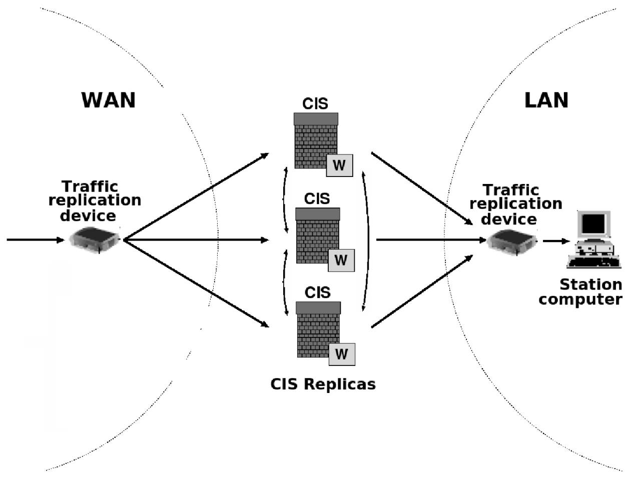


Figure 3.1: The CIS architecture.

votes the message. A message that satisfies the policy is called **legal**, (**illegal** otherwise).

All CIS replicas exchange with each others their vote through point-to-point reliable channels, using a cryptographic key K_W, shared between CIS replicas. Only messages positively voted by at least $f+1$ replicas are signed with a shared key K_{LAN}, known to the LAN, and then forwarded to the LAN; to avoid traffic multiplication only one randomly selected replica forwards the approved message to the LAN. It is assumed that computers located in the LAN accept only messages with a valid signature.

The objective of the CIS is to limit as much as possible the portion of the system that has to be guaranted reliable, or that can fail only in predefined ways, so each CIS replica is composed of two parts: the payload and the wormhole. By assumption each part can be affected by different types of failure, as follows:

- Payload: asynchronous system with $n = 2 * f + 1$ replicas in which at most f can be subject to *Byzantine failures*. Fault independence is assumed for the replicas, i.e., the probability of a replica to be compromised is independent of another replica failure. (this hypothesis is not unrealistic if the replicas have been separately develped).
- Wormhole: secure tamperproof subsystem $W = \{W_1, \ldots, W_n\}$ in which at most $f_c = f$ local wormholes can fail by crash. It is assumed that when a wormhole W_i crashes, the corresponding payload replica CIS_i crashes together.

A system is subject to Byzantine failure if there are no constraints on the kind of faults that can occur in the system. According to [14] byzantine failures are the most general type of failures: a Byzantine component is allowed any arbitrary behavior, for instance, a faulty process may change the content of messages, duplicate messages, send unsolicited messages, or even maliciously try to break down the whole system.

Messages arriving at CIS replicas both from the WAN and the LAN have unreliable fair multicast semantics: if a message is multicasted infinitely many times it will be received by all its receivers infinitely many times.

A more detailed description of the CIS and its characteristics can be found in [29, 7]. The aim of the model is to verify two basic properties of validity and integrity [29]; the satisfaction of these properties entails that only legal messages will be forwarded to their destination.

Theorem 1: Validity *A legal message received by at least one correct replica is forwarded to its destination.*

Theorem 2: Integrity *An illegal message is never processed by its destination machine.*

The algorithm for processing incoming messages by each replica is shown below (Algorithm 1)

Algorithm 1 CIS payload (replica CIS_i), from [7].

```
{Parameters}
integer $T_{vote}$ {Expected time to vote a message}
{Variables}
set $Voting = \varnothing$ {Messages being voted}
set $Pending = \varnothing$ {Not yet forwarded messages}
set $TooEarly = \varnothing$ {Messages forwarded before their arrival}
{Code for WAN message reception and processing}
upon $U\text{-}receive(WAN, m)$
 1: if $m_\sigma \in TooEarly$ then
 2:    $TooEarly \leftarrow TooEarly \setminus \{m_\sigma\}$
 3: else
 4:    if $PolEng\_verify(m)$ then
 5:       $Voting \leftarrow Voting \cup \{m\}$
 6:       $m_\sigma \leftarrow approve(m)$
 7:       $Voting \leftarrow Voting \setminus \{m\}$
 8:       $Pending \leftarrow Pending \cup \{m_\sigma\}$
 9:       $waitRandom()$
10:       if $m_\sigma \in Pending$ then
11:          $U\text{-}multicast(LAN, m_\sigma)$
12:       end if
13:    end if
14: end if
{Code for LAN message reception and processing}
upon $U\text{-}receive(LAN, m_\sigma)$
15: if $m_\sigma \in Pending$ then
16:    $Pending \leftarrow Pending \setminus \{m_\sigma\}$
17: else if $W\_verify(m_\sigma)$ then
18:    $TooEarly \leftarrow TooEarly \cup \{m_\sigma\}$
19: end if
function $approve(m)$
20: $vote_i \leftarrow W\_create\_vote(m)$
21: $\forall CIS_j \in CIS$ $send(j, \langle \text{VOTE}, \mathsf{H}(m), vote_i \rangle)$
22: $C_m \leftarrow \varnothing$
23: repeat
24:    wait until $receive(j, \langle \text{VOTE}, \mathsf{H}(m), vote_j \rangle)$
25:    $C_m \leftarrow C_m \cup \{vote_j\}$
26:    $\sigma \leftarrow W\_sign(m, C_m)$
27: until $\sigma \neq \perp$
28: return $m_\sigma$
{Periodic task for message retransmission}
for each $T_{vote}$ that $Voting \neq \varnothing$
29: $\forall m \in Voting$ : $U\text{-}multicast(WAN, m)$
```

The algorithm for a single replica consists of three components: Upon_ReceiveWAN, Upon_ReceiveLAN, and Retransmission. The three components of a *single* replica share three variables: *Voting*, the set of messages being voted by the replica, *Pending*, the set of messages that have been signed by the wormhole of the replica, but that have not yet been forwarded to the LAN and *TooEarly*, the set of correctly signed messages forwarded to the LAN by some other replica but that are currently not being treated, or even not yet received (from the WAN) by the replica. The interaction between replicas is either direct, through the reliable channels, or indirect, through the LAN, as we shall explain later on. T_{vote} is the single configuration parameter of the payload protocol and defines the expected time required to receive, vote and sign a legal message.

Primitives used by the payload are:

U-multicast(G,m), U-receive(G,m) : the former is invoked to multicast a message m to the group G, the latter is used to receive message m from G, where G can be either WAN or LAN;

PolEngineVerified: verify if a message is *legal*according to the deployed policy.

The wormhole offers the following primitives:

W_create_vote(m) authenticates vote message m with a key K_W shared between the wormholes; this key allows payload to distinguish whether a messege has been voted or not by a correct replica;

W_Sign(m,C_m) signs with K_{LAN} message m if and only if the replica payload presents a certificate set C_m containing at least $f+1$ valid votes produced by different wormholes and correctly signed with key K_W;

W_verify(m) tests if message m is correctly signed with K_{LAN}.

For each replica r, the algorithm works as follows:

Upon_ReceiveWAN (lines 1 to 14): when a message m is received by r from the WAN it can be forwarded to the LAN only if the wormhole of the replica signs it, and if no other replica has already sent this message to the LAN; if the message is in the set TooEarly the message is discarded, if the message is not in TooEarly it is processed. If it is a *legal* message r calls the Approve function. When the message is finally approved it is not sent immediately to the LAN, but only if, after a random delay, no other replica has forwarded it (to avoid more than one replica sending the same message to the LAN).

Approve (lines 20 to 28): create a vote by calling the wormhole and sends this vote to all replicas, through the direct channels. As soon as $f+1$ votes are receives the wormhole of r signs the message.

Upon_ReceiveLAN (lines 16 to 19): all replicas listen to the LAN, so when a replica sends an approved message to the LAN, all other replicas are aware of it. When a message m is received by r from the LAN, r removes it from the set Pending, thus allowing component Upon_ReceiveWAN of the same replica r to know that the message has been already forwarded to the LAN. If instead r has not seen the message at all, it verifies if the message was correctly signed (with the W_verify(m) function), and it includes the message in the TooEarly set, to avoid the Upon_ReceiveWAN component to take care of it at later stages.

Retransmission (line 29): this component is needed since there is no guarantee that all replicas (or at least the minimum number required for voting) receive the message from the WAN. If a message m sits in the Voting set of a replica for too long (according to the T_{vote} variable), then the replica assumes that the other replicas have not received m from the WAN and multicasts m over the WAN.

3.1.2 Model description

From the description in the previous section, it should be clear that the correctness of the algorithm, i.e. that only messages in accordance with the security policy will be forwarded to their destination, is the result of the interaction between different non-trivial sequences of events. This motivates the need for the construction and analysis of a formal model. The model represents only synchronization aspects (message approving) and concurrency, it does not represent cryptographic aspects in detail.

To model a CIS we need to understand some additional aspects of the algorithm, like:

1. level of concurrency: if a thread is the minimum unit of concurrency, how many threads do we have, and therefore what can be concurrently executed in the algorithm?
2. how are variables shared between threads?
3. how can we model fair multicast in a bounded model?
4. how can we model byzantine behaviour in a bounded model?

We shall consider the above points one by one in the following.

Level of concurrency. The algorithm is composed of three components per replica: Upon_ReceiveWAN, Upon_ReceiveLAN, and Retransmission. The first two are activated when a U-receive(WAN,m), (U-receive(LAN,m)) event happens. Message retransmission is instead a periodic task. For the first two components a new thread is created when their activation event happens. For example, whenever a message m originating from the WAN is received by a CIS replica, a new thread will be generated that processes message m.

How are variables shared between replicas and threads? Variables are shared only between threads generated by the same CIS replica, not between threads of different CIS replicas, so *variables are "local to replicas"*. CIS needs synchronization mechanisms that manage the concurrent access to variables by threads of the same replica, and mutual exclusive access is enough for the algortihm.

What is a fair multicast in practice? Fair multicast means that if a message is sent infinitely often, it will be infinitely often received. This hypothesis ensures that by adding to CIS the Retransmission task sooner or later all replicas will receive the message.

What are the implication of a byzantine behaviour? Byzantine means any behaviour, but for the algorithm only the actions of a replica that are visible (either directly or indirectly) from other replicas have to be taken into account. We have therefore analyzed the code, from which it results that a replica r can influence another replicas s in the following ways:

- r can keep retransmitting a *legal* or *illegal* message over the WAN, so that s receives the same message many times;
- r can send a *legal* or *illegal* message m, many times over the reliable direct link that connects r to s;
- r can send a message m many times over the LAN;

We have modelled the CIS using Stochastic Well-formed Nets (SWN) [18], a stochastic extension of Well-formed nets (WN). WN are an extension of the basic P/T nets [25] in which tokens can be identified (have "colours"), while SWN is an extension of WN that distinguishes two types of transitions: transitions that have an associated non-zero delay, whose values are specified through a stochastic distribution (Markovian distribution for exact solution, but also non-Markovian if simulation is used) or a zero delay (immediate transitions) that fire with priority over delay transitions. The state space of SWN is partitioned into tangible states (whose that enable delay transitions) and vanishing states (whose that enable immediate transitions). The choice of (S)WN allows to take advantage of the possibility of identifying components (typical of any type of colored nets), performing stochastic analysis (peculiar of any stochastic extension of colored nets), and of the symbolic solver, that allows a reduced state space made of symbolic markings that are eqivalence classes of markings. From the reachability analysis point of view SWN and WN are equivalent. All the models of CIS have been developed inside GreatSPN [11], for which SWN are a native language. The choice of (S)WN and of GreatSPN allows to take advantage of the possibility of identifying components (typical of any type of colored nets), composing subnets (using the GreatSPN subtool algebra [4]) and of the symbolic solver, that allows a reduced state space made of symbolic markings that are eqivalence classes of markings (called SRG). The use of SWN may allow, at later stages, a performance evaluation analysis of the efficacy of CIS in timely coping with intrusions.

In the following we list the choices used in building our model. Let us describe the basic model (the starting point of the validation activity).

- Colours: we identify messages and replicas, moreover messages can be *legal* or *illegal*, replicas can be *correct* or *faulty*. Consequentely, we have defined three colour classes: M, identifier of a message, T the type of a message, *legal* or *illegal*, R the identifier of the replicas, split into two subclasses to distinguish *correct* from *faulty*. A message will be represented by pairs m, t, so the same message can be *legal* for a replica and *illegal* for another one (to account for byzantine behaviour of *faulty* replicas), while a replica is a single variable r, and it will be either *correct* or *faulty* (which implies that the model does not allow a replica to change behaviour: it is either *correct* or *faulty*).

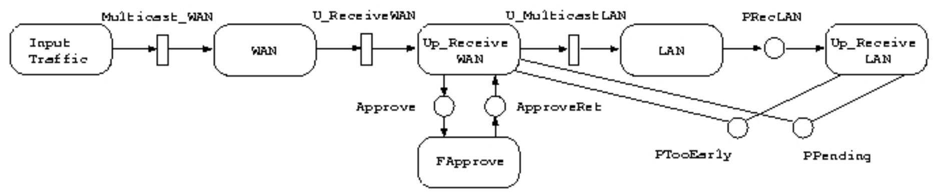


Figure 3.2: The composition schema.

- Variables Pending, Voting, TooEarly: since they are local to replicas, they are colored with the identity of the replicas (the R colour).
- Messages signed with K_W or K_{LAN}, are modeled as normal messages, which means that in different portions of the net colour T, type of message, has different meaning. Indeed, in the execution of the system after the Approve, what is relevant is not if the message is *legal* or *illegal*, but if it is signed or not.
- The number of parallel threads for a single component of a single replica is limited. Note that this limitation is not present in the algorithm, but it must be present in an implementation, and in this work we have "lifted" this aspect to the specification level.
- WAN and LAN can have reliable or unreliable behaviour (where reliable means that all messages sent are received as they are in finite time, and unreliable means messages can be dropped).
- Payload and Wormhole of a given replica fail by fail-crash (they stop working); note that this is weaker than the hypothesis of the algorithm that allows also Payload to change from *correct* to *faulty*.

We have built the model as composition over transitions and places of common labels of six submodels: Input Traffic, WAN, Up_ReceiveWan, LAN, Approve, UpReceiveLAN, as shown in Figure 3.2, where common places and transitions are shown, for clarity, outside of each component. This specific choice of modularity was done so as to minimize the number of components to be changed while considering different steps in the model validation, as explained in the analysis part, or to limit the size of the submodel (this is the reason why a separate Approve submodel has been designed).

Our starting point is the *Basic Model*, which is built with the following, simplifying, hypothesis: LAN and WAN are reliable (so that there is no need for Retransmission), LAN, WAN and replicas do not fail, Byzantine behaviour of *faulty* replicas is limited to changing the legality of messages that are distributed for voting to the other replicas.

We now describe the components of the *Basic Model* one by one, recalling that immediate transitions are represented as thin bars and have priority over non-zero delay (timed) transitions, that are depicted as boxes, function on arcs are either identity functions (variables x, y, etc.), the S function or a linear combination. Function S stays for "one token per colour", and function S on an input arc to transition t means that t is a synchronization

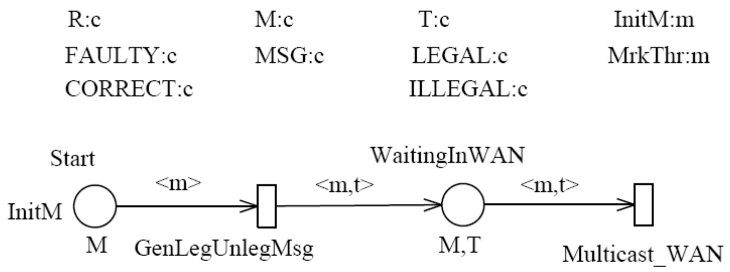


Figure 3.3: The SWN that generates the WAN incoming (legal and illegal) traffic

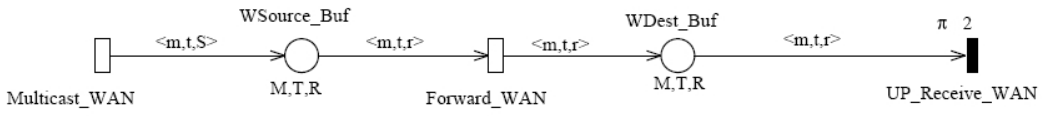


Figure 3.4: The SWN of the WAN transmission, including multicast

requiring one token per color, while S on an output arc means that t is a fork that puts one token per color in the output place.

Figure 3.3 is the SWN of the traffic generator (subnet 'input Traffic"): the *Start* place is initialized with one element per message, legality of the message is defined nondeterministically/probabilistically (by transition *GenLegUnlegMsg*, that takes t as free variable), and a message, with its type, is then multicasted by transition *Multicast_WAN*. The subnet of the WAN, shown in Figure 3.4, includes the multicast (realized by the function S on the arc out of *Multicast_WAN*) that puts messages in the input buffers from the WAN of the replicas. The two previous models are rather simple because we assume that the WAN is reliable.

Figure 3.5 shows the SWN for the Up_ReceiveWAN task (line 1 to 14 of the algorithm). The subnet is initialized with one token per replica (function S applied to colour R) in place *PThreadW* that limits the number of concurrent threads for the Up_ReceiveWAN task. The net is highly connected with the other components, through transitions *UP_Receive_WAN* *U_Multicast* and places *Approve* and *ApproveRet* (to call the Approve function) and places *PTooEarly*, *PVoting*, *PPending*, that represent the corresponding variables. Informations in these variables accounts for the replica identity, as well as for the message identity. Note that a CORRECT replica only forwards to Approve the *legal* messages (*illegal* ones are dropped), while FAULTY replicas forward any type of message, simply complementing its type. Once a replica r gets a messages signed by the Approve, it does not multicast it immediately to the LAN, but it puts it in *PPending*, it waits for a random delay, and only if at the end of the delay the message is still in *PPending*, it will be multicasted on the LAN (through transition *U_Multicast*), if instead the message is not in *PPending*, it means that some other replica has forwarded it to the LAN, and the thread terminates.

Figure 3.6 depicts the SWN of the LAN on which an approved message is multicasted

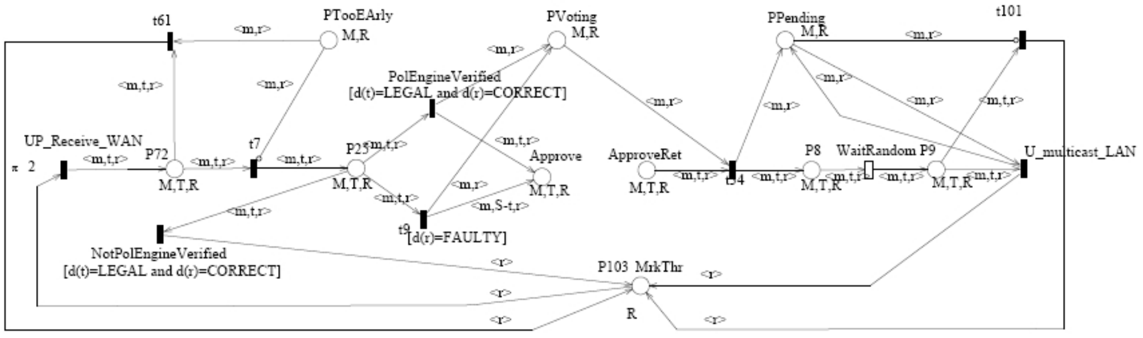


Figure 3.5: The SWN of the CIS behaviour upon receiving from WAN

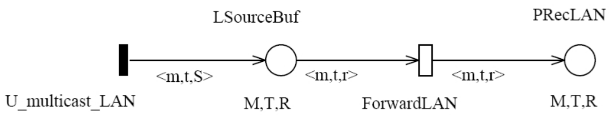


Figure 3.6: The SWN of the reliable LAN behaviour

by the replica that has not found the message still sitting in Pending after the random delay, while Figure 3.7 describes the thread Up_ReceiveLAN (lines 15 to 19 of the algorithm). When a replica r receives a signed message from the LAN it means that some other replica has forwarded it to the LAN. If the message is in *PPending*, it removes it, and if it was not in *PPending* and it is correctly signed with the K_{LAN} key, it puts the message in *PTooEarly*, to avoid treating this message at later stages. Place *PThreadL* contains the available trheads: as we shall discuss later, it is important that there are enough threads for each replica.

Finally, the Approve function (lines 20 to 28 of the algorithm) is modelled by the SWN of Figure 3.8, that interacts with the other subnets only through the two places *Approve* and *ApproveRet*. An important role for understanding the net is played by place *BufFromDirectLink*, that represents the input buffer from the reliable direct channels that connect the replicas. When a replica r calls the Approve, its vote is broadcasted to all replicas, including r itself, by transition *t10* but the same vote may reach different replicas at different times, since the transmission is modelled by the stochastic transition *DirectLink*. The Approve function for replica r and message m terminates when either two or three replicas have voted m as legal. Note the use of two immediate transitions (*t5* and *t6*) to clean the buffer (place *BufFromDirectLink*) upon entering Approve (this correspond to line 22 of the algorithm). This is a portion of net that *is not parametric in the number of distinct replicas*, so to increase the number of replica it is necessary to manually modify the Approve subnet. Although the model is not parametric, the extension to larger numbers of replicas is quite straightforward: the case of $2 * k + 1$ replicas requires $k + 1$ transitions: k of them replace *T6*, to account for $k + 1, k + 2, \ldots, 2 * k$ replicas having already voted, while *T7* stays unchanged.

The component models have been defined using the tool GreatSPN [11], and the composition was done using the tool algebra [4] that allows to compose SWN using transition and place superposition over places. In the figures the labels have substituted the name of

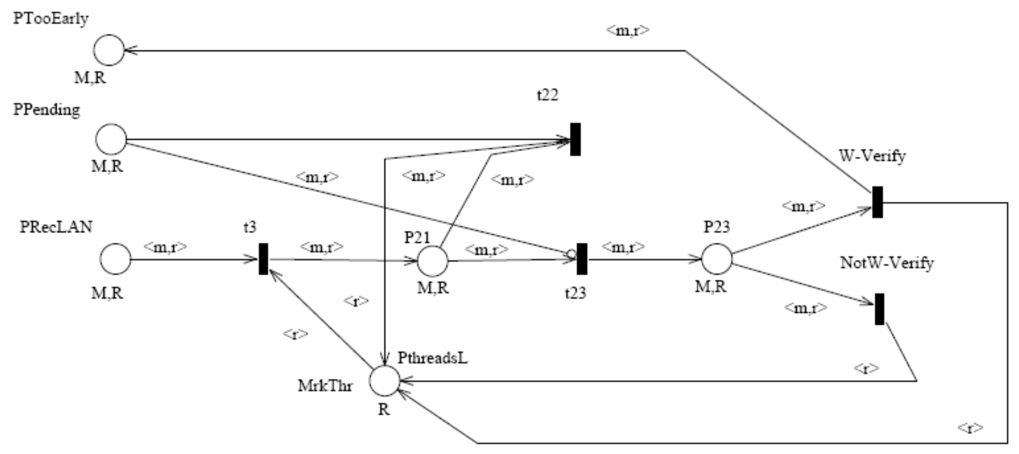


Figure 3.7: The SWN of the behaviour upon receiving from the LAN.

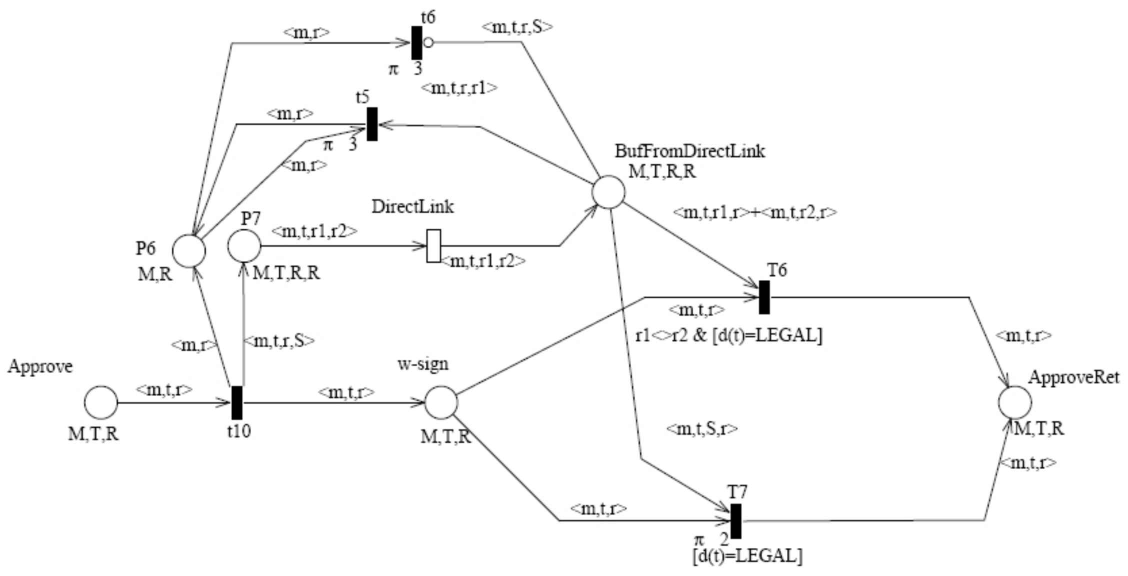


Figure 3.8: The SWN of the Approve function - a CIS votes according to the messages received.

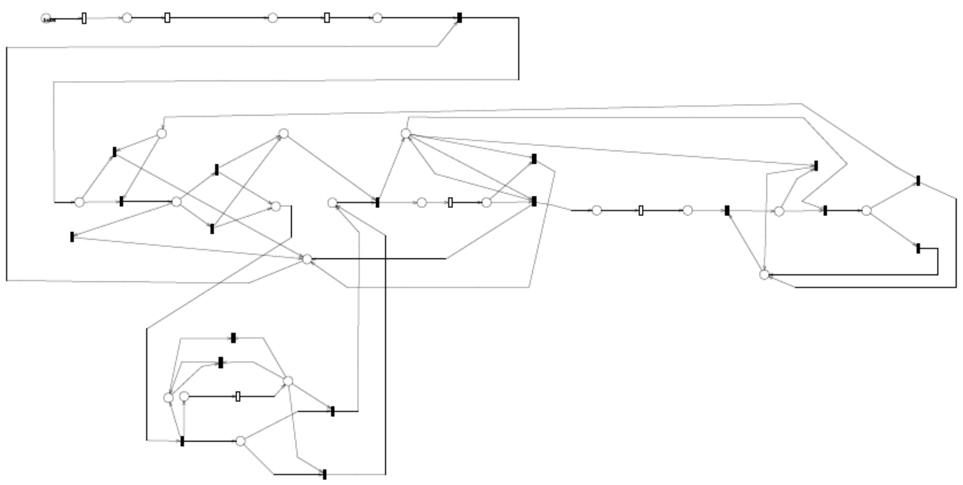


Figure 3.9: The composed SWN for the Basic Model

places and transitions for simplicity. The result of the composition is shown in Figure 3.1.2. Even from this schematic SWN it is clear that the model does not exhibit cyclic behaviour.

3.1.3 Incremental validation

We have built the model and we have validated it with the following modelling process:

- Choose the level of abstraction of the representation, adequate for the analysis purposes.
- Model each component separately as a subnet and compose the subnets into a complete net. A large number of deadlocks are obtained: is this a correct behaviour?
- To check the deadlocks we consider the simple case of only correct replica and legal messages: results are not as expected.
- A detailed analysis is done in the case of 2 replicas only, and a malfuctioning on the use of the variable TooEarly has been revealed. The case of 3 replicas, all correct, produces an unexpected large state space.
- Single components are analysed in isolation: this allow to reveal a malfunctioning of the Approve function; a reduced model is built in which the Approve component is abstracted away by a single transition.
- The reduced model is used to analyze the behaviour in presence of more than a message.

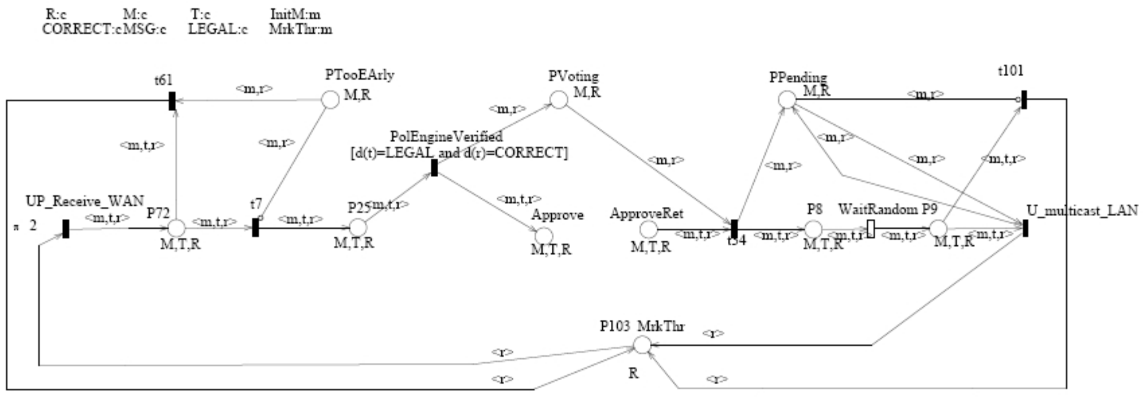


Figure 3.10: The SWN of the CIS behaviour upon receiving from WAN without *illegal* messages and faulty behaviour.

The first two points have already been treated in the previous section on model construction.

The analysis of the Basic Model of Figure 3.1.2 may not be trivial for large values of the color class R of replicas and M of messages, so we started with $|R| = 3$ and $|M| = 1$, with two replicas in subclass CORRECT and one in FAULTY, and one single message, that can arrive to a replica as either *legal* ot *illegal*, independently from the replica.

For this simple case we have 42.876 tangible states, 36 deadlocks and 211.187 vanishing states (that we shall indicate as $42.876T+36D+211.187V$ for short). The symbolic reachability graph has instead the following number of symbolic states $21.864T + 25D + 106.429V$ with a reduction factor of about 2 (which means that, on the average, each symbolic marking is an equivalence class with two ordinary markings), smaller for deadlock states. The analysis of the 25 deadlocks reveals some unexplained behaviour, and to get a better understanding we decided to backtrack to a simpler model, that we call Model Zero.

Model Zero: no faulty replicas, no illegal messages, two replicas. We have built a model of CIS under "perfect behaviour", that is too say messages are always legal and all replicas are CORRECT. We have checked all subnets and observed that we need to eliminate the two color subclasses ILLEGAL from M and FAULTY from class R, and to change only the subnet of Figure 3.5 to remove all transitions that check if messages are ILLEGAL and if replicas are FAULTY. The resulting net is shown in Figure 3.10, where an additional output place *Count* has been added for transition *U_Multicast_LAN* to count and to identify the messages that are sent to the LAN. Moreover we have decreased the number of replicas from 3 to 2, by changing the cardinality of R. The resulting net has $107T + 4D + 302V$ ordinary states and $58T + 3D + 151V$ symbolic ones.

The analysis of the four deadlocks is now feasible, and reveals that the net ends either in a single good state, where the only places marked are *PThreadW* and *PThreadL*, that represent the pool of threads, or in three states in which place *PTooEarly* is marked, in particular there is one state in which place *PTooEarly* is marked for both replicas, meaning that both replicas have received a message from the LAN while their Pending variable was empty. How could this happen? After some thought we realized that there is a misfunctioning of the algorithm if more than one replica does a multicast on the LAN: the first message cleans the Pending variable, and the second one puts it in TooEarly. This behaviour is indeed possible, since it can happen that two replicas ends the random delay one shortly after the other, before they have been able to receive the message from the other replica. This is a subtle behaviour that it is likely to manifest itself only in presence of slowdowns in the LAN, so it may be a difficult error to catch by testing. Indeed, some time after we found this malfunctioning, it was confirmed also by the authors of the algorithm, while testing it under a denial of service attack, a situation that can significantly increase the LAN and WAN time to delivery. The error never manifested itself instead in all tested cases in which the LAN was working normally.

Model Zero-1: no faulty replicas, no illegal messages, three replicas. We have run the same model with three replicas and we have been very surprised to observe a state space of more than half a million states (after which the state space generation has been blocked). What makes a model with 2 replicas to stay within a few hundred states and a model with three correct replicas to explode and even exceed the case with two CORRECT replicas and one FAULTY? We have suspected some serious malfunctioning and we have proceeded in analyzing the components in isolation.

Models Zero-2: analysis of the isolated components of Model Zero-1. We have modified the subnet components so as to be able to generate the state space in isolation. Input transitions have been connectd to an appropriately initialized place and input places have been appropriately initialized. The analysis was particularly interesting for the Approve component: if two replicas are put in the *PApprove* place, then the net ends in a single deadlock in which the two replicas are in place *PApproveRet*, as expected. If instead we put three replicas, all correct, the state space increases from $16T + 1D + 15V$ to $974T + 27D + 919V$, so that we are now faced with 27 deadlocks, moreover this is the same size that we get if one of the trhee replicas if FAULTY. The analysis of the state space reveals that transition $t7$ never fires, meaning that the Approve always uses exactly two votes, and the rest are left in place *BufDromDirectLink*. The first observation correspond to a correct behaviour (there is no point in waiting for a third vote if two are enough, and this justify also why there is no difference if one replica is FAULTY), but the second one should have been taken care by transitions *t5* and *t6*, that model statement 22 of the algorithm. Indeed statement 22 takes care of cleaning the input buffer for subsequent runs of the approve *on the same message m*, and not for different m, which means that the input buffers from the direct links are likely to exceed their capacity (especially if one replica behaves in a byzantine manner). Again, this appear to be as a malfunctioning of the algorithm, or, more probably, as an underspecification of the algorithm, that delegates to the implementation an adequate management of the buffers. Up to know the problem is solved through the use of timeouts that regulate the reset of the buffers.

	1	2	3
Ordinary Tangible: O_TRS	53	3.127	175.589
Ordinary Deadlocks: O_D	3	9	27
Ordinary Vanishing: O_VRS	186	20.832	1.749.888
Ordinary Total : O_VRS	242	23.968	1.925.504
Symbolic Tangible: O_TRS	29	817	16.193
Symbolic Deadlocks: O_D	2	4	6
Symbolic Vanishing: O_VRS	93	5.208	152.070
Symbolic Total : O_VRS	124	6.029	168.269

Table 3.1: Number of states for the case of $|R| = 2$ and for varying number $|M|$ of messages

Model Zero-3: a reduced model Having observed that the Approve does vote correctly if the buffers are managed correctly, but that left-over messages in the buffers highly increase the state space, we have produced a reduced model, in which the Approve subnet is substituted by a single transition that directly connects place *PApprove* and *PApproveRet* in the subnet of Figure 3.5, to investigate the behaviour of the model in presence of more messages, more replicas and more threads. Each of the three aspects are considered in separated paragraphs that follow.

Model Zero-3: more messages The state space sizes for the reduced model with two replicas ($|R| = 2$) and 1, 2, and 3 messages is shown on the columns of Table 3.1. In the table each column represents a model for which we list, on the rows, the sizes of the ordinary tangible, deadlock and vanishing states, the total number of ordinary states, and the corresponding symbolic ones in the last four rows. The initial marking of the available threads has been set to allow for as many theads as there are replicas and messages. A first observation, somewhat independent on the fact that we are concentrating over varying number of messages, is that there is a significant saving in using the symbolic state space generation, as evident when comparing the two rows that report the total number of ordinary and symbolic state: the symbolic state space is 12 times smaller than the ordinary one for the case of trhee messages.

Considering that messages are dealt with almost separately by the algorithm, we would have expected the state space for $|M| = 2$ to be the Cartesian product of the state spaces for the $|M| = 1$ case. But this is indeed not the case, it is significantly smaller. Actually, the total size of the state space can be determined from the case of $|M| = 1$ through two simple formulas:

$$|O\_T_{m+1}| = |O\_T_m| * |O\_T_1|$$

$$|O\_VRS_{m+1}| = |O\_T_m| * |O\_VRS_1| \quad + \quad |O\_VRS_m| * |O\_T_1|$$

where we have indicated with the index m the case of m messages, and with O_T the sum of tangible and deadlock states. When we apply the formula to the case of $m = 1$, it says that the cross-product of tangible states is a tangible state (not surprisingly), that the cross-product of a tangible for a vanishing (and viceversa) is a vanishing. What is surprising is that “vanishing states do not multiply”: but this has a clear explanation, since starting from a tangible initial state, there is no timed transition that, once fired, enables an immediate transition for the first message and one for the second, since messages are dealt with separately in the model.

Another observation from the table is about the number of deadlocks: we have 3, 9, and 27 ordinary deadlocks, that may not be easy to map onto a general algorithm behaviour, since ordinary markings contain too much detail, that is instead abstracted in the symbolic markings, that are equivalence classes of ordinary markings. To show the difference between the ordinary and symbolic deadlocks, let us consider the case of a single message that we call m, while r_1 and r_2 are the two replicas. The three ordinary deadlocks correspond to the following three situations: $r1$ was the only one to send m to the LAN, $r2$ was the only one to send m to the LAN, both r_1 and r_2 have sent m to the LAN. Recall that the first case is possible because the algorithm does not appropriately deal with the case of a very slow LAN. The two symbolic deadlocks represent instead the same three situations, but at a higher level of abstraction, by identifying two macro states: only r_i has sent m to the LAN (corresponding to two ordinary deadlock) and both r_1 and r_2 have sent m to the LAN (corresponding to one ordinary deadlock). We report in the following the two symbolic markings as created by the SWN tool GreatSPN for the readers taht are more acquainted with the formalism (for each symblic marking there is the cardinality of the dynamic subclasses listed right after the marking itself, where REP indicates replicas and MSG messages).

```
SYMBOLIC MARKING D18 # 2 ordinary marking
PThreadL(1<REP0>1<REP1>)PThreadW(1<REP0>1<REP1>)Count(1<MSG0,REP1>)
|MSG0|=1 |REP0|=1 |REP1|=1

SYMBOLIC MARKING D31 # 1 ordinary marking
PThreadL(1<REP0>)PThreadW(1<REP0>)PTooEArly(1<MSG0,REP0>)Count(1<MSG0,REP0>)
|MSG0|=1 |REP0|=2
```

Model Zero-3: more replicas For the analysis under a varying number of replicas we fix the number of messages to one since, by the previous analysis, is should be clear that, if there are enough threads, messages are treated independently. The state space sizes for the reduced model with one message ($|M| = 1$) and 2, 3, and 4 replicas is shown on the columns of Table 3.2.

We observe, as before, that there is a significant saving in considering symbolic states instead than ordinary ones. To check that the behaviour of the model for varying number of replicas correctly represents the algorithm behaviour we can check the deadlock states, that, in the symbolic case, are very few (equal to the number of replicas).

For the case of 4 replicas the four deadlocks identify the following macro behaviour:

- D286: one replica has sent m to the LAN, and three have not: corresponds to 4

	2	3	4
Ordinary Tangible: *O_TRS*	53	1.185	37.129
Ordinary Deadlocks: *O_D*	3	7	15
Ordinary Vanishing: *O_VRS*	186	7.116	331.272
Ordinary Total : *O_VRS*	242	8.308	368.416
Symbolic Tangible: *O_TRS*	29	244	2.209
Symbolic Deadlocks: *O_D*	2	3	4
Symbolic Vanishing: *O_VRS*	93	1.285	16.761
Symbolic Total : *O_VRS*	124	1.532	18.974

Table 3.2: Number of states for the case of $|R| = 2$ and for varying number $|M|$ of messages

ordinary deadlocks;

- D1514: two replicas have sent m to the LAN, and two have not: corresponds to 6 ordinary deadlocks;
- D2152: three replicas have sent m to the LAN, and one has not: correspond to 4 ordinary deadlocks;
- D2213: all replicas have sent m to the LAN: correspond to a single ordinary deadlock.

```
SYMBOLIC MARKING D286 # 4 ordinary marking (dead)
PThreadL(1<REP0>1<REP1>)PThreadW(1<REP0>1<REP1>)Count(1<MSG0,REP1>)
|MSG0|=1 |REP1|=1 |REP0|=3

SYMBOLIC MARKING D1514 # 6 ordinary marking (dead)
PThreadL(1<REP0>1<REP1>)PThreadW(1<REP0>1<REP1>)PTooEArly(1<MSG0,REP0>1<MSG0,CRRECT1>)
|MSG0|=1 |REP1|=2 |REP0|=2

SYMBOLIC MARKING D2152 # 4 ordinary marking (dead)
PThreadL(1<REP0>1<REP1>)PThreadW(1<REP0>1<REP1>)PTooEArly(2<MSG0,REP0>2<MSG0,REP1>)Cou
|MSG0|=1 |REP0|=1 |REP1|=3

SYMBOLIC MARKING D2213 # 1 ordinary marking: dead)
PThreadL(1<REP0>)PThreadW(1<REP0>)PTooEArly(3<MSG0,REP0>)Count(1<MSG0,REP0>)
|MSG0|=1 |REP0|=4
```

3.1.4 Theorems verification

The models produced have been analyzed with the objective of verifying Theorems 1 and 2 of Section 3.1.1. A possible verification technique is model checking, that requires to translate each theorem in an appropriate (temporal) logic formula and to check if the resulting formulas are satisfied by the SWN model. GreatSPN does not include direct access to a model checker, but actually the two properties to be verified are rather simple and can be reduced to reachability analysis of a (modified) net. We have followed the following steps:

- Modify the net behaviour so as to trace violation of the theorems during its evolution.
- Build the modified model and produce the SRG.
- Inspect the dead markings of the SRG and check if the theorems are verified.

For both theorems we have checked two versions of the Basic Model of Figure 3.1.2: one with $|R| = 3$ and $|M| = 1$, with three replicas all in subclass CORRECT, and the other with two replicas in subclass CORRECT and one in FAULTY.

Validity. The first theorem can be translated as: *If a legal message is received by at least one correct replica then it will be forwarded to its destination.*

Since an implication is false only when the antecedent is true and the consequent is false, we simply check that this does not happen during the evolution of the system. We have modified the subnet component Upon_ReceiveWAN adding a test place and some transitions so as to put a token in this place if the antecedent became true, and analogously for the consequent. From this modified model we have generated the SRG: symbolic dead markings in which the antecedent place contains a token but consequent does not, do not exist, therefore we conclude that the theorem is true.

Integrity. We have generated the SRG of the original Basic Model and we have inspected that it does not contain symbolic markings where place LSourceBuf is occupied by a token $< m, t, r >$ with t equal to *illegal*; from this fact we conclude that an illegal message is never forwarded to the LAN.

3.1.5 Discussion

Although the models presented can still be considered preliminary with respect to proving the correctness of CIS in all possible contexts of application, they already show that extreme care should be taken while managing the variables TooEarly and the input buffer from the direct links that connect the replicas. These problems have been acknowledge by the team that has developed the CIS middleware, and they have been solved in the enhanced version of the CIS with proactive and reactive strategies.

3.2 DEEM Models of the Proactive-Reactive Recovery Strategy

A quantitative analysis was performed on the Proactive-Reactive Recovery strategy proposed to reinforce the intrusion tolerance of the CIS in the scope of the protection service.

The relevant characteristics of the system were modeled as Deterministic Stochastic Petri Nets (DSPN) [22] and solved using the DEEM tool [8].

This section contains the description of the models and the evaluation of the obtained results; a discussion about the consequences of the proposed analysis is in [12], together with some proposals about the refinement of the recovery strategy.

3.2.1 System Overview

CIS is a substation gateway interfacing a protected LAN with the WAN (as shown in Figure 3.11). In order to be intrusion-tolerant, the CIS is replicated (with diversity) in n machines and follows its specification as long as at most f of these machines are attacked and behave maliciously, both toward other replicas and toward the station computers in the protected LAN.

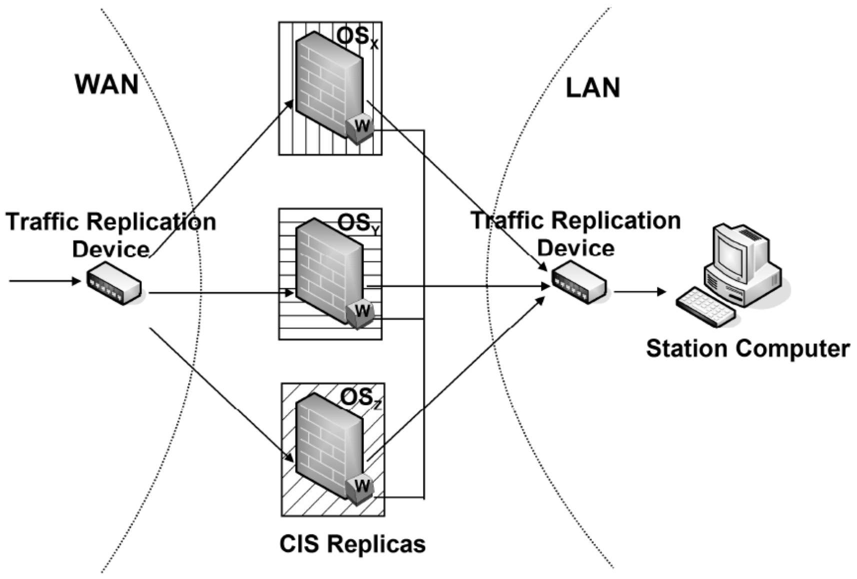


Figure 3.11: CIS intrusion tolerant hybrid architecture

CIS intrusion tolerance is enhanced rejuvenating CIS replicas through recovery actions. In order to guarantee system availability despite the unavailability of recovering replicas, the maximum number of replicas allowed to recover in parallel is defined (k) and the number of replicas in the system is set to $n \geq 2f + k + 1$. This way, the system is able to tolerate (at most) f Byzantine replicas and recover k replicas simultaneously.

The CIS protection service, executed in each payload replica, verifies whether each incoming message m complies with the security policy (OrBAC[1]), notifying the (positive) approval to the local wormhole. As soon as a quorum of $f + 1$ approvals for m is reached, the wormhole signs m and the current leader replica forwards it to the destination node in the LAN.

[1]OrBAC (Organization Based Access Control)

The wormhole is in charge of both triggering the recovery actions when necessary and managing the election of the new leader when necessary.

3.2.2 The Proactive-Reactive Recovery Strategy

The Proactive-Reactive Recovery Wormhole (PRRW) strategy [23] manages the CIS replica recoveries using a mix of proactive and reactive recovery actions. Proactive recoveries are performed based on periodic base, whilst reactive recoveries are performed on replicas “suspected” or “detected” to be compromised. The leader replica is “suspected” to be compromised if it does not forward a signed messages; a generic replica is “detected” to be compromised if it sends a not signed (invalid) message to the LAN. Accusations about a replica being “suspected” or “detected” can be raised by each payload replica and sent to the corresponding local wormhole through a specific interface; a replica is “suspected” or “detected” when a quorum of at least $f+1$ accusations about that replica are collected by the wormhole.

The PRRW strategy is organized as follows: time is divided in $\lceil n/k \rceil$ different time slots that are cyclically repeated. Each slot is divided in two tasks: task A and task R_i, with $i = 1, \ldots, \lceil n/k \rceil$.

Proactive (periodic) recoveries are executed during task R_i only; up to k replicas recover simultaneously in each task R_i, according to the replica index. Replica i, with $i = 1, \ldots, k$, are recovered in task R_1, replica i, with $i = k+1, \ldots, 2 \cdot k$, are recovered in task R_2 and so on. Task R_i lasts for (at most) T_{D} and it is executed again after a period T_{P}.

Two types of reactive (a-periodic) recoveries can be triggered on replica i:

1. “Immediate” reactive recovery, triggered if a quorum of $f+1$ accusations exists about i sending illegal messages; in this scenario replica i is “detected” of being compromised, because at least one correct replica detected that replica i is failed.
2. “Delayed” reactive recovery, triggered if a quorum of at least $f+1$ accusations exists about the current leader i, some about i sending illegal messages, other about i not forwarding a signed message (the signed messages was not forwarded for more then O_{t} times). In this scenario the leader replica i is “suspected” of being compromised, because at least one correct replica raised an accusation about leader replica i, but the wormhole is not able to identify which accuser replica is correct, so it is not able to identify which kind of accusation is correct about leader replica i.

“Immediate” reactive recoveries are immediately triggered on replica i as soon as the replica is detected of being compromised.

“Delayed” reactive recoveries are only triggered on the leader replica, are executed during task A and are coordinated with proactive recoveries. If no “immediate” reactive recovery is already triggered for replica i, the PRRW strategy finds the closest recovery sub-slot where the recovery of replica i does not endanger the availability of the CIS. If the found sub-slot is located in the slot where replica i will be proactively recovered, the "delayed" reactive recovery is not performed. Task A is divided into $\lceil f/k \rceil$ recovery sub-slots identified as S_{ij}; up to k replicas can be recovered simultaneously in each sub-slot. Task A lasts for (at most) $\lceil f/k \rceil T_D$.

Each slot lasts hence for up to $(\lceil f/k \rceil + 1) T_D$ with period T_P. After each R_i task has been executed once, each replica has been proactively recovered once.

A new leader is elected by the wormhole if the current leader is recovering (e.g. because it was suspected of being omissive) or if the local wormhole of the current leader is detected to be crashed. The new leader is chosen as the (currently not crashed) replica more recently recovered by a proactive recovery.

In [6, 12], the leader "suspected" of being compromised is replaced by the election of a new leader as soon as the recovery of the current leader starts, i.e., at the beginning of the closest recovery sub-slot where the leader can be recovered. In this report, a more efficient strategy is adopted, being a new leader elected as soon as the current leader is "suspected" of being compromised, without waiting for its recovery.

3.2.3 Fault Model and Assumptions

This section describes the fault model [28] and the assumptions on which the fault model is based on. Station computers are assumed to only accept messages signed by the wormhole (a symmetric key K is shared between the station computer(s) and the CIS wormhole). The following faults are considered:

f1) The faults related to communication involve both the traffic replication devices, the communication channels among them and the replicas (except the control channel connecting local wormholes). Traffic replication devices can loose messages coming from a port or sometimes delay the traffic forwarding on some ports (for an unbounded time); traffic replication devices cannot generate spurious messages. Communication channels can loose messages or unpredictably delay the traffic forwarding.

f2) A payload replica can be intruded, and hence can be affected by Byzantine faults; if more than f payload replicas fail, the CIS fails.

f3) A local wormhole can only fail by crash; at most $f_c \leq f$ local wormholes are assumed to fail by crash. The crash of a local wormhole is detected by a perfect failure detector. When a local wormhole crashes, the corresponding payload is forced to crash together.

f4) Fault-independence is assumed for payload replicas, i.e., the probability of a replica being faulty is independent of the occurrence of faults in other replicas (this assumption can be substantiated in practice through the extensive use of several kinds of diversity [5]).

f5) The same attack on the same replica has always the same probability of success.

f6) Station computers cannot be compromised.

f7) Replicas are correct after their recovery.

f8) The security policy verified by the CIS is assumed to be perfect; this means that a correct replica applies perfectly the policy verification and there are no policy inconsistencies between replicas (i.e. all correct replicas verify the same policy).

Given the set of faults described above, the corresponding failure modes for a payload replica are the following:

Crash. The payload replica crashes because of the crash of the corresponding local wormhole (f3) or as the effect of an intrusion (f2).

Omission. The replica payload is subjected to a temporary omission because of communication problems (f1) or as the effect of an intrusion (f2) (e.g. the leader payload is omitting to forward a signed message to its destination or the net is flooded with messages by someone else).

Invalid. The payload replica is failing by value as the effect of an intrusion (f2), e.g. it is sending illegal messages toward the LAN or it is flooding the WAN and LAN networks with illegal messages aiming to delay the forwarding of legal messages.

For ease of modeling, we assume that a replica, as soon as it is successfully intruded, explicitly manifest failures (of any kind) and that a failure caused by an intrusion is permanent.

The system fails when the number of invalid replicas is greater than f (the correctness of the system cannot be guaranteed) or when the necessary resources are not available for too long (the CIS seeks perpetual operation); the system is unavailable when the number of correct working replicas is less then $f+1$ (so quorums cannot be reached) or there are more than $f+1$ correct replicas, but the leader is omitting (so legal messages are not forwarded).

3.2.4 Measures of Interest

The relevant measures of interest for the recovery strategy under study are the system failure probability and the system unavailability; moreover, we are interested in assessing the impact of leader's omission over the above mentioned measures, given that a "delayed" reactive recovery can be triggered on the leader replica only.

The system fails at time t if one of the following conditions holds:

1. the number of invalid replicas gets over f;

2. the system is unavailable for an interval of time greater then T_{O}.

Let $P_{\mathrm{FI}}(t)$ be the probability of the system being failed at time t because of condition 1, given that it was correctly functioning at time $t = 0$. Let $P_{\mathrm{FO}}(t)$ be the probability of the system being failed at time t because of condition 2, given that it was correctly functioning at time $t = 0$. System failure probability $P_{\mathrm{F}}(t)$ is defined as the probability of the system being failed at time t, given that it was not failed at time $t = 0$, and it is obtained as

$$P_{\mathrm{F}}(t) = P_{\mathrm{FI}}(t) + P_{\mathrm{FO}}(t).$$

The system is unavailable at time t if one of the following conditions holds:

1. the number of correct replicas is less than $f + 1$ (quorums cannot be reached);
2. there are more than $f + 1$ correct replicas, but the leader is omitting (legal messages are not forwarded).

Let $T_{\mathrm{U}}(0, t)$ be the total time the system is not failed but unavailable within $[0, t]$ because of one of the above conditions. Let $T_{\mathrm{A}}(0, t)$ be the total time the system is not failed within $[0, t]$. System unavailability $P_{\mathrm{U}}(0, t)$ is defined as the probability of the system being unavailable within $T_{\mathrm{A}}(0, t)$, given that it was correctly working at time $t = 0$; system unavailability is obtained as:

$$P_{\mathrm{U}}(0,\ t) = \frac{T_{\mathrm{U}}(0,\ t)}{T_{\mathrm{A}}(0,\ t)}.$$

The leader replica - beyond its role of system replica - has the “special” task of forwarding legal messages towards the LAN; the impact of leader’s omission over the system measures of interest is hence based only on the omission about its “special” task. Let $T_{\mathrm{UL}}(0, t)$ be the total time the system is not failed but unavailable within $[0, t]$ because of leader’s omission; the contribution of leader’s omission over system unavailability, denoted with $P_{\mathrm{UL}}(0, t)$, is obtained as:

$$P_{\mathrm{UL}}(0,\ t) = \frac{T_{\mathrm{UL}}(0,\ t)}{T_{\mathrm{A}}(0,\ t)}.$$

Let $P_{\mathrm{FL}}(0, t)$ be the contribution of leader omission over probability of system failure $P_{\mathrm{FO}}(0, t)$. Assuming that the impact of leader’s omission over $P_{\mathrm{FO}}(0, t)$ is the same as the impact over system unavailability, the contribution of leader’s omission over probability of system failure because of protract unavailability $P_{\mathrm{FO}}(0, t)$ is obtained by solving the following proportion:

$$P_{\mathrm{FL}}(0,\ t) : P_{\mathrm{FO}}(0,\ t) = P_{\mathrm{UL}}(0,\ t) : P_{\mathrm{U}}(0,\ t)$$

from which:

$$P_{\mathrm{FL}}(0,\ t) = \frac{P_{\mathrm{FO}}(0,\ t) \cdot P_{\mathrm{UL}}(0,\ t)}{P_{\mathrm{U}}(0,\ t)}$$

3.2.5 PRRW Modeling

The model and the evaluation proposed in the following Sections are an extensions of those proposed in [12]; the main differences involve the triggering of the "immediate" reactive recovery actions and the triggering of the leader election after a "delayed" recovery action.

From the modeling point of view, a reconfiguration strategy determines a discontinuity in the CIS configuration caused by the temporary unavailability of the replicas subjected to a recovery. Therefore it is possible to represent the entire operational life split into different periods of deterministic duration called phases. This feature makes a reconfiguration strategy belonging to the Multiple Phased System (MPS) class for which a modeling and evaluation methodology exist [22], supported by the DEEM tool [8].

Using DEEM, the net is split into two logically distinct sub-nets: the Phase Net (PhN) representing the schedule of the various phases, each one of deterministic duration, and the System Net (SN) representing the behavior of the system. Each net is made dependent on the other by marking-dependent predicates that modify transition rates, enabling conditions, reward rates etc. Reward measures are defined as Boolean expressions, functions of the net marking. Both the analytic and simulation solutions can be used in order to exercise the models; the measures of interest defined in our quantitative analysis were evaluated by simulation.

Phase Net The phase net (Figure 3.12) models the PRRW scheduling described in Section. 3.2.2. The deterministic transitions *TsubSlot* and *TRi* model the times to perform the tasks A and R_i, respectively. Place *Sij* contains a token during the task A (a-periodic recovery phase) and *Ri* contains a token during the task R_i (periodic recovery phase). The marking of *CountSubSlot* counts the number of the current recovery sub-slot (S_{ij}) within the current recovery slot. The marking of *CountSlot* counts the number of the current recovery slot within the current cycle. The marking of *CountWin* counts the number of the current cycle. The immediate transition *tNextSlot* fires when a periodic recovery slot ends, resetting the marking of *CountSubSlot* to 1. The immediate transition *tNextWin* fires when a new cycle is started, resetting the marking of *CountSlot* to 1. The immediate transitions of the phase net have priority less than the priorities of the immediate transitions of the system net.

System Net The system net of the PRRW model is composed by $n \leq 6$ similar subnets (one subnet for each replica), a subnet to keep track of system failures and a subnet to model the initialization of the system net itself (the description of this last subnet is omitted without affecting the comprehension of the model).

Figure 3.13 shows the subnet modeling replica 1. The left part of the subnet models the replica failures, while the right part of the subnet models the replica recovery and leader election. Places which name ends with digit "1" model replica 1, while the other places

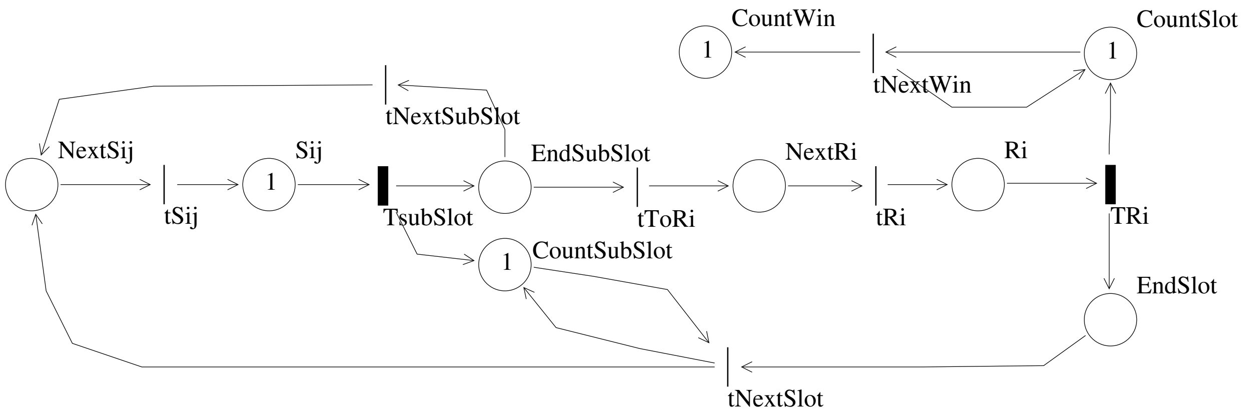


Figure 3.12: The phase net of the PRRW model

(*Leader* and *kRec*) are shared by all the replicas.

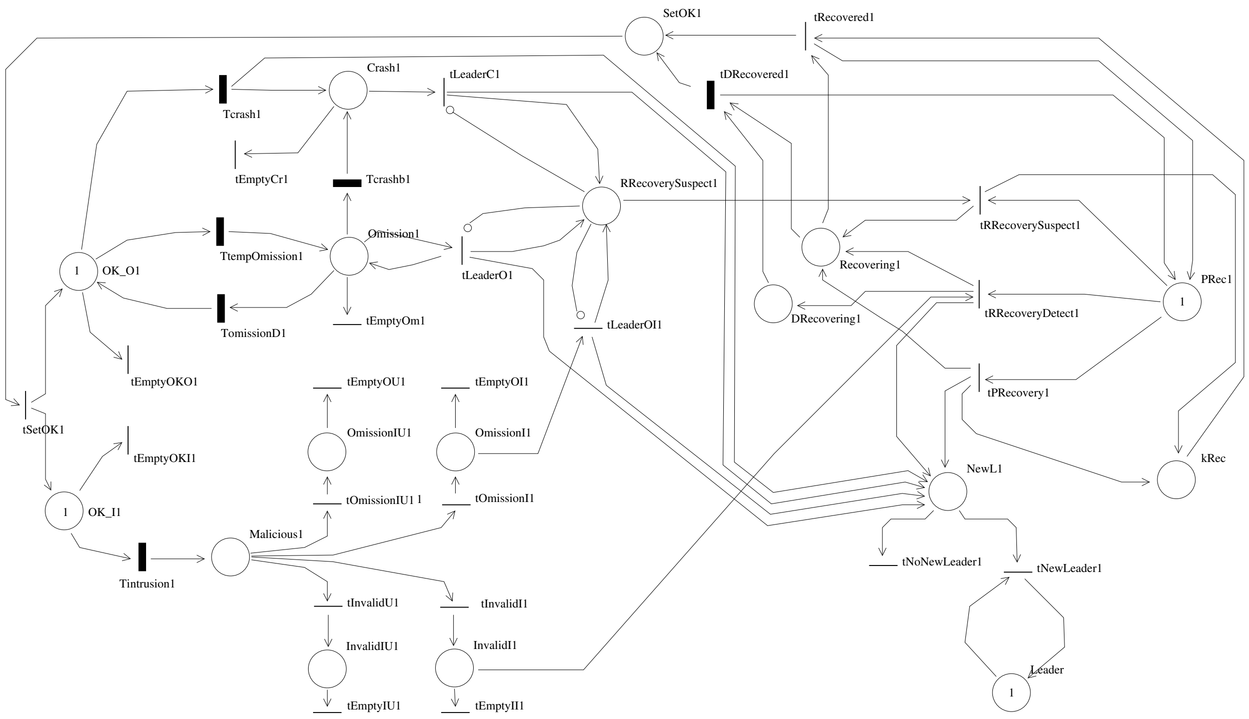


Figure 3.13: The subnet modeling replica 1 in the PRRW model

Replica failures are modeled as follows. As long as both *OK_O1* and *OK_I1* contain one token each, replica 1 is correctly working. One token in places *Crash1* or *Omission1* represents the crash of the replica or an omissive behavior as a consequence of a transient omission, respectively. The exponential transitions *Tcrash1* and *Tcrashb1* represent the time to the crash with rate λ_1^c; when the replica crashes, place *OK_I1* is emptied (the replica cannot be no more intruded). *TtempOmission1* represents the time to a transient omission exponentially distributed with rate λ_1^o. A transient omission disappears after a time modeled by the exponential transition *TomissionD1* with rate λ^{eo}.

The exponential transition *Tintrusion1* represents the time to intrusion with rate λ_1^a;

the effect of the intrusion is modeled by the following immediate transitions (enabled in the same marking) and the associated places:

- *TomissionIU1* for an undetectable omission failure, with probability $(1 - c_{\mathrm{M}})(1 - p_{\mathrm{I}})$,
- *TomissionI1* for a detectable omission failure, with probability $c_{\mathrm{M}}(1 - p_{\mathrm{I}})$,
- *TinvalidIU1* for an undetectable invalid failure, with probability $(1 - c_{\mathrm{M}})p_{\mathrm{I}}$,
- *TinvalidI1* for a detectable invalid failure, with probability $c_{\mathrm{M}}p_{\mathrm{I}}$,

where p_I and c_{M} are the probability of an intrusion manifesting as a permanent invalid behavior and the detection coverage of malicious behavior, respectively.

The replica recovery is modeled as follows. Place *PRec1* contains a token as long as replica 1 is not recovering, while place *Recovering1* contains one token as long as the replica is recovering. Place *DRecovering1* contains a token during an "immediate" reactive recovery (triggered by detections). Place *kRec* is used to count the number of replicas currently recovering. Place *RRecoverySuspect1* contains a token if a crash, an omission or a malicious omission occurs.

Recoveries are triggered by one of the following immediate transitions (ordered by increasing priorities): *tRRecoverySuspect1* ("delayed" reactive recovery triggered by suspects), *tRRecoveryDetect1* ("immediate" reactive recovery triggered by detections) or *tPRecovery1* (proactive recovery). The immediate transition *tRRecoverySuspect1* fires if a new a-periodic recovery sub-slot is starting (*NextSij* contains a token) and less than k replicas are recovering (*kRec* contains less than k tokens) and the replica is not going to be proactively recovered in the next periodic slot (the index of the replica is not in the interval $[(Mark(CountSlot) - 1)k + 1,\ Mark(CountSlot)k]$). The immediate transition *tRRecoveryDetect1* fires independently from the marking of the phase net (in [12] it was triggered at the beginning of a recovery sub-slot). The immediate transition *tPRecovery1* fires if a periodic recovery slot is starting (*NextRi* contains a token) and less than k replicas are recovering (*kRec* contains less than k tokens) and the index of the replica is in the interval $[(Mark(CountSlot) - 1)k + 1,\ Mark(CountSlot)k]$.

When a recovery action starts, all the immediate transitions which name starts with *tEmpty* fire, emptying the following places: *OK_O1*, *OK_I1*, *Crash1*, *Omission1*, *InvalidIU1*, *InvalidI1*, *OmissionIU1*, *OmissionI1* and *OKLeadO*. When the recovery action ends, the immediate transitions *tRecovered1* or *tDRecovered1* fire, resetting the replica subnet.

The election of the leader replica is managed as follows. The marking of place *Leader* corresponds to the index of the current leader; when replica 1 either is going to be recovered or is crashed, one token is added in place *NewL1*. The immediate transition *tNewLeader1* fires if replica 1 is the current leader, triggering the mechanism of election of a new leader, otherwise *tNoNewLeader1* fires. The arc from place *Leader* to transition *tNewLeader1* has multiplicity equal to *Mark*(*Leader*), while the arc from transition *tNewLeader1* to place *Leader* has multiplicity equal to the index of the replica that will be elected as the new leader. The new leader should be the last (not crashed) replica proactively recovered, that is replica with index $j = ((n + (Mark(CountSlot) - 2)k) mod\, n) + k$. If replica j is currently crashed, the next attempt is made on replica $j - 1$, until a not crashed replica is found.

The subnet shown in Figure 3.14 models the system failure. Place *OKSysN* contains a token as long as the system is not failed and it is not omitting (there are more than f correct replicas and the leader is not crashed or omitting). Place *OKSysO* contains a token when the system is not failed but it is omitting; place *OKLeadO* contains a token when the system is not failed, but it is omitting because the leader replica is omitting. Place *SysFailureI* contains a token when the system is failed because of invalid behavior (there are at least $f+1$ invalid replicas). Place *SysFailureO* contains a token when the system is failed because the resource unavailability lasted for an unacceptable period of time represented by the exponential transition *TSysO* with rate $1/T_{\mathrm{O}}$.

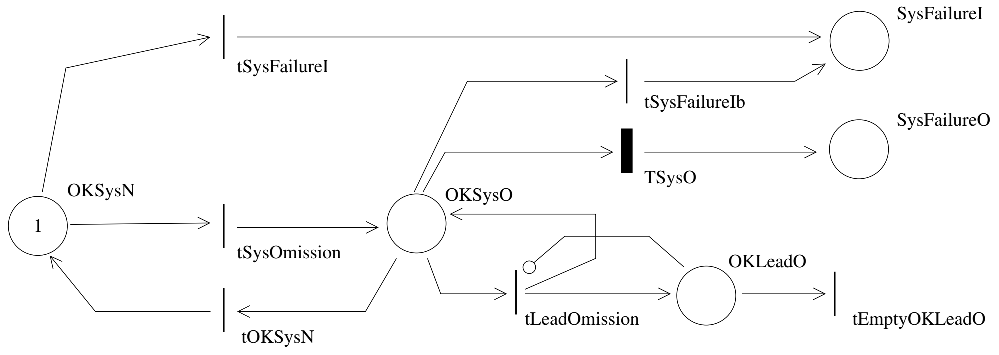


Figure 3.14: The model of system failure in the SN of the PRRW model

Different priorities are associated to the immediate transitions of SN, when no probabilistic choices are required. For example, all the immediate transitions of replica i have priorities lower than those of replica j, if $i < j$.

3.2.6 Reward Structures

This section describes how the measures of interest described in Section 3.2.4 were evaluated using the DEEM tool.

The evaluation of a measure of interest in DEEM involves specifying a performance (reward) variable and determining a reward structure for the performance variable, i.e., a reward structure which associates reward rates with state occupancies and reward impulses with state transitions [27].

The measures of interest related to system failure probability ($P_{\mathrm{F}}(t)$, $P_{\mathrm{FI}}(t)$ and $P_{\mathrm{FO}}(t)$) were evaluated in terms of three "instant of time" performance variables based on the following reward structures respectively:

```
if (Mark(OKSysO)=0 and Mark(OKSysN)=0) then (1) else (0)
if (Mark(SysFailureI)=1) then (1) else (0)
if (Mark(SysFailureO)=1) then (1) else (0)
```

System unavailability $P_{\mathrm{U}}(0,t)$ was evaluated as $P_{\mathrm{U}}(0,\, t) = \frac{T_{\mathrm{U}}(0,\, t)}{T_{\mathrm{A}}(0,\, t)}$. $T_{\mathrm{U}}(0,t)$ and $T_{\mathrm{A}}(0,t)$ were evaluated defining two "interval of time" performance variables which reward structures are the following respectively:

```
if (Mark(OKSysO)=1) then (1) else (0)
if (Mark(OKSysO)=1 or Mark(OKSysN)=1) then (1) else (0)
```

The contribution of leader's omission over system unavailability $P_{\mathrm{UL}}(0,t)$ was evaluated as $P_{\mathrm{UL}}(0,\, t) = \frac{T_{\mathrm{UL}}(0,\, t)}{T_{\mathrm{A}}(0,\, t)}$. $P_{\mathrm{UL}}(0,t)$ was evaluated defining an "interval of time" performance variable which reward structure is the following:

```
if (Mark(OKLeadO)=1 or Mark(OKSysN)=1) then (1) else (0)
```

3.2.7 Model Evaluation and System Analysis

In this Section the results of the evaluation of the measures of interest are shown. The measures of interest were evaluated by simulation [21] with a confidence level of 95% and a half-length confidence interval of 1%.

All the model parameters and the default values used for the evaluations are shown in Table 3.3; the value for T_{D} was taken from [28]. The relevant parameters are:

1. Mission time t. This is the time during which the system is exercised since it starts to work. t varies in [2628, 42048] sec.
2. Probability p_{I} of intrusion within a replica manifesting as a permanent invalid behavior. p_{I} varies in [0, 1]. If $p_{\mathrm{I}} = 0$ then all intrusions manifest as a permanent omissive behavior; in this case, only "delayed" reactive recoveries (on the leader replica) can be triggered. If $p_{\mathrm{I}} = 1$ then all intrusions manifest as a permanent invalid behavior; in this case, intrusions on each replica can only trigger "immediate" reactive recoveries.
3. Detection coverage c_{M} of malicious behavior of a replica. c_{M} is the probability of detecting an intruded replica, and hence the probability of reactively recovering an intruded replica. c_{M} varies in [0, 1]. If $c_{\mathrm{M}} = 0$ then no intrusions are detected; in this case, all intrusions are treated by proactive recoveries and reactive recoveries are only triggered by crash or communication omissions. If $c_{\mathrm{M}} = 1$ then all intrusions are detected and treated by reactive recoveries.

Table 3.3: Parameters and their default values

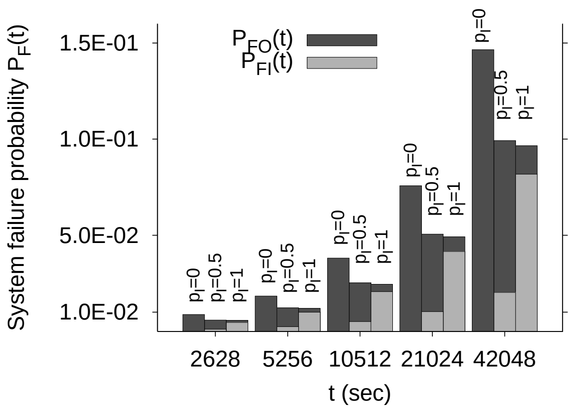
Name	Default Value	Meaning
t	2628	Mission time (sec)
n	4	Number of replicas in the system
k	1	Max number of replicas recovering simultaneously
f	1	Max number of corrupted replicas tolerated by the system
T_{D}	146	Time duration of a recovery operation (sec)
T_{O}	60	Duration of system omission before considering the system failed (sec)
λ_i^{c}	$[1.9 \cdot 10^{-7}, 3.8 \cdot 10^{-7}]$	Crash rate of replica i. Each replica has a diverse crash rate (from 1 per 60 days to 1 per 30 days)
λ_i^{o}	$[1.9 \cdot 10^{-6}, 3.8 \cdot 10^{-6}]$	Transient omission rate of replica i. Each replica has a diverse rate (from 1 per 6 days to 1 per 3 days)
λ^{eo}	$3.3 \cdot 10^{-2}$	Omission duration rate of a replica. A transient omission lasts for 30 seconds (on average)
λ_i^{a}	$[5.8 \cdot 10^{-5}, 1.2 \cdot 10^{-5}]$	Successful attack (intrusion) rate of replica i. Each replica has a diverse rate (from 5 per day to 1 per day)
p_{I}	0.5	Probability of intrusion within a replica manifesting as a permanent invalid behavior (if $p_{\mathrm{I}} = 0$ all intrusions manifest as permanent omissions)
c_{M}	0.7	Probability of detecting malicious behavior of a replica

4. Number n of system replicas in the system, maximum number f of corrupted replicas tolerated by the system itself and maximum number k of system replicas recovering simultaneously, with $n = 2f + 1 + k$.

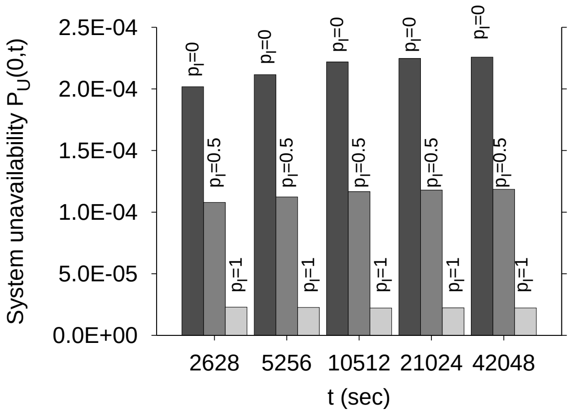
A first study was performed observing both system failure probability $P_{\mathrm{F}}(t)$ and system unavailability $P_{\mathrm{U}}(0,t)$ over mission time t for three different values of p_{I}.

Figure 3.15(a) shows how $P_{\mathrm{FI}}(t)$ and $P_{\mathrm{FO}}(t)$ change over mission time t, with $P_{\mathrm{F}}(t) = P_{\mathrm{FI}}(t) + P_{\mathrm{FO}}(t)$. $P_{\mathrm{F}}(t)$ increases exponentially over time for all the values of p_{I}. $P_{\mathrm{F}}(t)$ behaves in fact like a geometric random variable for the following reasons. System failure probability during each recovery period (cycle) is not null; after each cycle the system is rejuvenated, so we can assume that the system failure probability during the next cycle is the same as the previous one. So system failure probability $P_{\mathrm{F}}(t)$ cumulates over the recovery periods as a geometric random variable. The values of $P_{\mathrm{F}}(t)$ are over 0.01 because of the values assigned to the system parameters. As p_{I} varies from 0 to 1, $P_{\mathrm{F}}(t)$ increases of about 65% for each value of t. For $p_{\mathrm{I}} = 0$, $p_{\mathrm{I}} = 0.5$ and $p_{\mathrm{I}} = 1$ the value of $P_{\mathrm{FI}}(t)$ is about 0%, 20% and 83% of the value of $P_{\mathrm{F}}(t)$, respectively, independently on the values of t.

If $p_{\mathrm{I}} = 0$ then $P_{\mathrm{FI}}(t) = 0$, because there is no invalid behavior, and hence $P_{\mathrm{F}}(t) = P_{\mathrm{FO}}(t)$. As p_{I} varies from 0 to 1, $P_{\mathrm{FO}}(t)$ changes from 100% of $P_{\mathrm{F}}(t)$ to 17% of $P_{\mathrm{F}}(t)$; the number of intrusions does not change, but the effect of intrusions changes. In fact, the value of $P_{\mathrm{FO}}(t)$ depends on the time during which replicas are unavailable, which for $p_{\mathrm{I}} = 0$ is



(a) System failure probability $P_F(t)$



(b) System unavailability $P_U(0, t)$

Figure 3.15: System failure probability $P_F(t)$ and system unavailability $P_U(0, t)$ over mission time t for different values of p_I

given by the sum of the following durations:

- the time spent waiting for a "delayed" reactive recovery of the omissive leader;
- the time spent during the recovery on the omissive leader;
- the time spent waiting for proactive recoveries of (not leader) omissive replicas;
- the time spent for proactive recoveries (not varying for the different values of p_I).

If $p_I = 1$ then the time during which replicas are unavailable is given by the sum of the following durations:

- the time spent during "immediate" reactive recoveries on replicas detected as intruded; the number of these recoveries is about n times the number of "delayed" reactive recoveries performed for $p_I = 0$;
- the time spent for proactive recoveries.

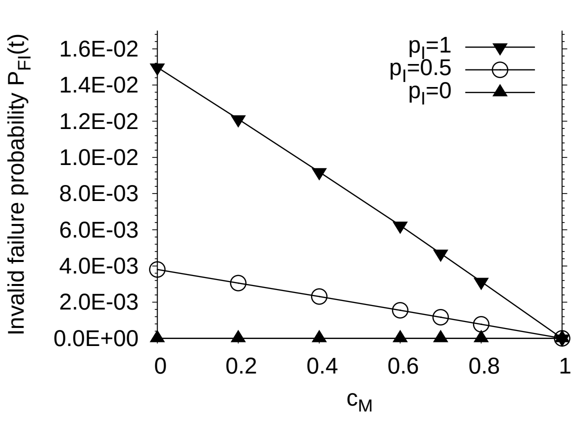
Therefore, the value of $P_{FO}(t)$ for $p_I = 1$ mainly represents the impact of recoveries (both proactive and reactive) on $P_F(t)$ (crashes and transient omissions are still present, but have lower rates than intrusions). The value of $P_{FO}(t)$ for $p_I = 1$ shows that the impact of recoveries on $P_F(t)$ is low (about 17%).

Figure 3.15(b) shows how $P_U(0, t)$ changes over mission time t. $P_U(0, t)$ increases over time for all the values of p_I: for $p_I = 0.5$ and $p_I = 1$ the value of $P_U(0, t)$ is about 53% and 10% of the value of $P_U(0, t)$ for $p_I = 0$, respectively, independently on the values of t.

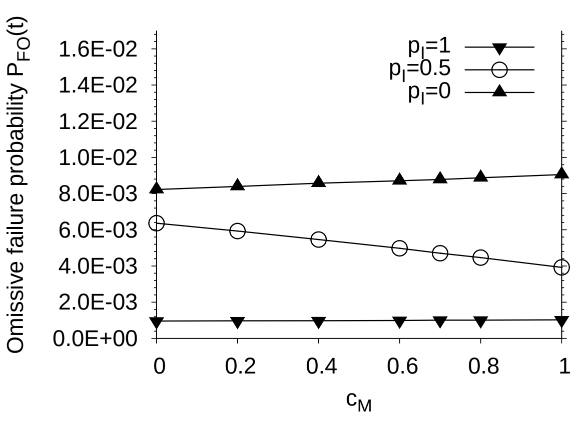
The trend of $P_U(0, t)$ for varying p_I is similar to the trend of $P_{FO}(t)$ for varying p_I shown in Figure 3.15(a): for $p_I = 1$ the value of $P_U(0, t)$ is mainly due to the reactive recoveries, for $p_\mathrm{I} = 0$ and $p_\mathrm{I} = 0.5$ the value of $P_\mathrm{U}(0,t)$ is influenced by the fact that the number of reactive recoveries decreases but the number of omissions increases.

Another study was devoted to evaluate both system failure probability $P_\mathrm{F}(t)$ and system unavailability $P_\mathrm{U}(0,t)$, varying both the detection coverage c_M and the probability p_I of intrusions manifesting as invalid behavior. This study shows how reactive recoveries improve the measures of interest with regard to treating intrusions with proactive recoveries only.

Figures 3.16(a) and 3.16(b) show how $P_\mathrm{FI}(t)$ and $P_\mathrm{FO}(t)$, respectively, change over detection coverage c_M for different values of p_I; in order to make easier their comparison, the same scale for the y-axis is used. $P_\mathrm{FI}(t)$ decreases as c_M increases from 0 to 1 for all the values of p_I ($P_\mathrm{FI}(t)$=0 for $p_\mathrm{I} = 0$ independently from the value of c_M). $P_\mathrm{FI}(t)$ takes larger values for $p_\mathrm{I} = 1$ than for $p_\mathrm{I} = 0.5$. $P_\mathrm{FO}(t)$ shows a different behavior with respect to $P_\mathrm{FI}(t)$, given that $P_\mathrm{FO}(t)$ takes larger values for lower values of p_I. $P_\mathrm{FO}(t)$ is almost constant for p_I=1 (the value for c_M=1 is about 6% larger than the value for c_M=0); it decreases for p_I=0.5 as c_M increases from 0 to 1 (the value for c_M=1 is about 40% of the value for c_M=0); it slightly increases for p_I=0 as c_M increases from 0 to 1 (the value for c_M=1 is about 10% larger than the value for c_M=0).



(a) Invalid failure probability $P_\mathrm{FI}(t)$

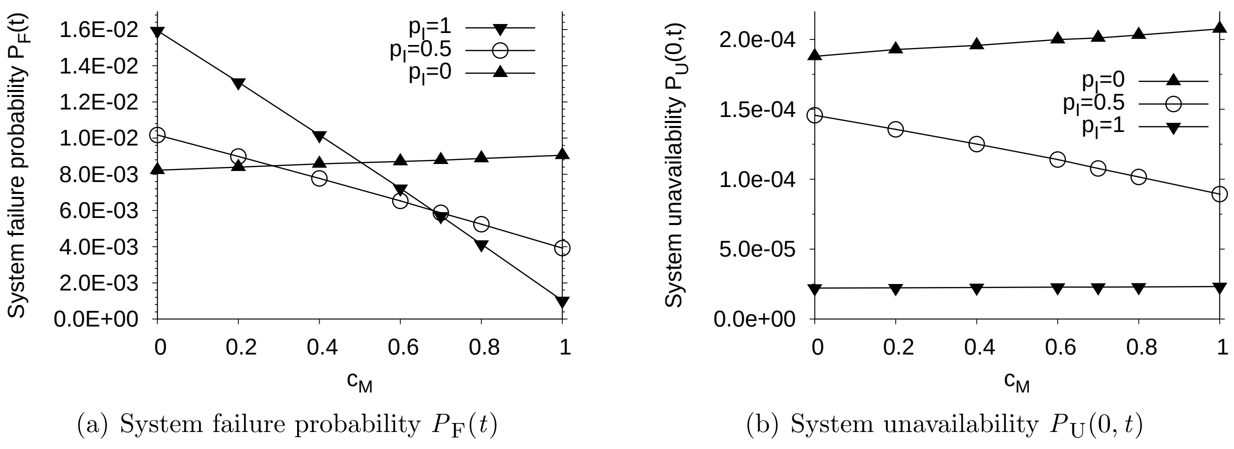


(b) Omissive failure probability $P_\mathrm{FO}(t)$

Figure 3.16: Impact of detection coverage c_M on both $P_\mathrm{FI}(t)$ and $P_\mathrm{FO}(t)$ for different values of p_I

The values of $P_\mathrm{FI}(t)$ and $P_\mathrm{FO}(t)$ for $c_\mathrm{M} = 0$ correspond to the system configuration in which all the intrusions are treated only by proactive recoveries. The difference between the values of $P_\mathrm{FI}(t)$ (and $P_\mathrm{FO}(t)$) for $c_\mathrm{M} = 0$ and $c_\mathrm{M} = 1$ is due to the effect of treating all the intrusions by reactive recoveries: $P_\mathrm{FI}(t)$ decreases, because invalid replicas reactively recovered are no longer weakening the system; $P_\mathrm{FO}(t)$ is almost constant, because there are k "extra" replicas which contribute to system operation while the intruded replicas are recovering. The overall effect, shown in Figure 3.17(a), is that, when most of the intrusions behave as invalid ($p_\mathrm{I} \geq 0.5$), system failure probability $P_\mathrm{F}(t)$ decreases as detection coverage c_M increases. On the contrary, when most of the intrusions behave as omissions ($p_\mathrm{I} < 0.5$), the impact of c_M on $P_\mathrm{F}(t)$ is negligible. This stresses that, in order to improve the value of

$P_F(t)$, it is useful to trigger reactive recoveries and hence to set the value for c_M as higher as possible.



(a) System failure probability $P_F(t)$ (b) System unavailability $P_U(0, t)$

Figure 3.17: Impact of detection coverage c_M on system failure probability $P_F(t)$ and system unavailability $P_U(0, t)$ for different values of p_I

Figure 3.17(b) shows how system unavailability $P_U(0, t)$ changes over detection coverage c_M for different values of p_I. The trend of $P_U(0, t)$ for varying c_M is similar to the trend of $P_{FO}(t)$ shown in Figure 3.16(b). $P_U(0, t)$ takes larger values for lower values of p_I. $P_U(0, t)$ is almost constant for p_I=1 (the value for c_M=1 is about 4% larger than the value for c_M=0); it decreases for p_I=0.5 as c_M increases from 0 to 1 (the value for c_M=1 is about 40% of the value for c_M=0); it slightly increases for p_I=0.5 as c_M increases from 0 to 1 (the value for c_M=1 is about 10% larger than the value for c_M=0).

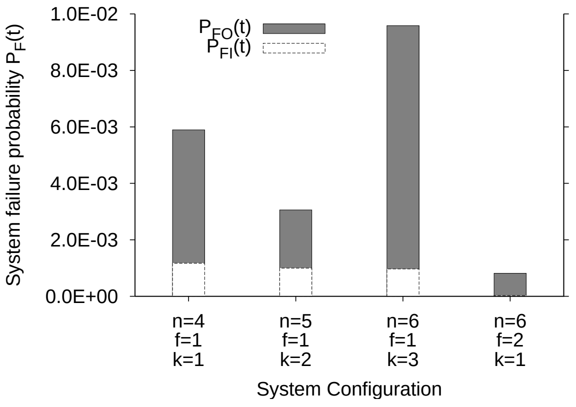
The results of this study show that increasing the detection coverage of intrusions c_M has positive effects on system failure probability $P_F(t)$ and has no negative effect on system unavailability $P_U(0, t)$.

Another study was devoted to evaluate the impact of the number of replicas on both system failure probability $P_F(t)$ and system unavailability $P_U(0, t)$. When dealing with the number of replicas in the system, three parameters are relevant: n, the overall number of replicas in the system, f, the maximum number of corrupted replicas tolerated by the system and k, the maximum number of replicas simultaneously recovering without endangering the availability of the system, with $n = 2f + 1 + k$. The following system configurations were evaluated:

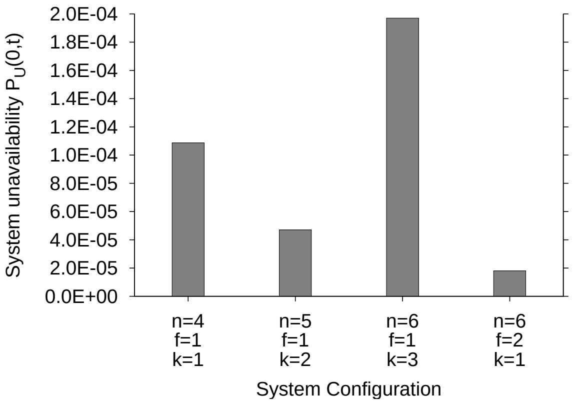
1. $n = 4, f = 1, k = 1$
2. $n = 5, f = 1, k = 2$
3. $n = 6, f = 1, k = 3$
4. $n = 6, f = 2, k = 1$

Figures 3.18(a) and 3.18(b) show system failure probability $P_{\mathrm{F}}(t)$ (decomposed in $P_{\mathrm{FI}}(t)$ and $P_{\mathrm{FO}}(t)$) and system unavailability $P_{\mathrm{U}}(0,t)$ for the system configurations described above.

$P_{\mathrm{FI}}(t)$ decreases as n (and k) increases; the trend of $P_{\mathrm{F}}(t)$ is mainly due to the trend of $P_{\mathrm{FO}}(t)$. The largest value for $P_{\mathrm{F}}(t)$ is obtained for configuration 3 ($n = 6$, $f = 1$). For the same value of $n = 6$ (configuration 3 and 4), the higher is f and the lower is $P_{\mathrm{F}}(t)$ (this behavior is shown both for $P_{\mathrm{FI}}(t)$ and $P_{\mathrm{FO}}(t)$); configuration 4, although having a lower value for k, shows a lower value for $P_{\mathrm{FI}}(t)$ because it has a more robust intrusion tolerance schema ($f = 2$); $P_{\mathrm{FO}}(t)$ is lower because the frequency of proactive recoveries is lower ($k = 1$). The trend of $P_{\mathrm{U}}(0,t)$ is the same of $P_{\mathrm{F}}(t)$.



(a) System failure probability $P_{\mathrm{F}}(t)$



(b) System unavailability $P_{\mathrm{U}}(0,t)$

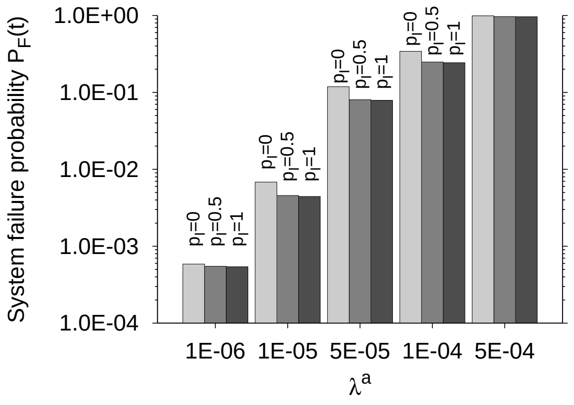
Figure 3.18: System failure probability $P_{\mathrm{F}}(t)$ and system unavailability $P_{\mathrm{U}}(0,t)$ for different system configurations at mission time $t = 2628\,\mathrm{sec}$

We suppose that the increment of the value of $P_{\mathrm{FO}}(t)$ varying from configuration 2 to 3 is due to the combined effect of a larger number of failures (n varies from 5 to 6, but $f = 1$) and a higher frequency for proactive recoveries (k varies from 2 to 3). It turns out that for the setting used (as shown in Table 3.3) the lower values for $P_{\mathrm{F}}(t)$ and $P_{\mathrm{U}}(0,t)$ are obtained for the system configuration 4, i.e., for higher values of f, independently of k.

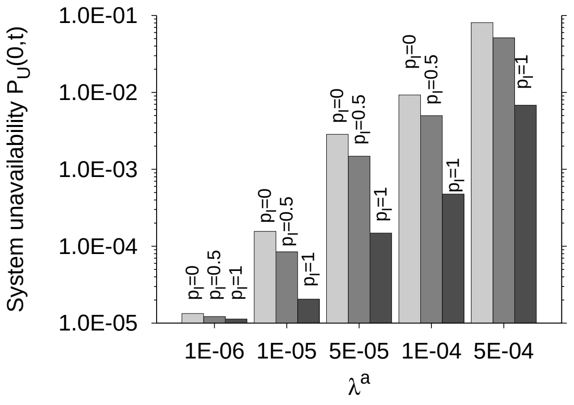
A study was devoted to evaluate the impact of successful attack (intrusion) rate λ^{a} over system failure probability $P_{\mathrm{F}}(t)$ and system unavailability $P_{\mathrm{U}}(0,t)$ for different values of p_{I}.

Figures 3.19(a) and 3.19(b) show that both system failure probability $P_{\mathrm{F}}(t)$ and system unavailability $P_{\mathrm{U}}(0,t)$ increase exponentially as attack rate λ^{a} increases (the y-axis of both figures uses a log scale); in particular the increment is about four orders of magnitudes for both the measures of interest. The behavior of the two measures of interest with respect to varying the p_{I} is in general the following: the value of the measure of interest decreases as p_{I} increases. In particular, looking at the values of $P_{\mathrm{F}}(t)$ for the smallest λ^{a}, Figure 3.19(a) shows that the values for p_{I}=1 and p_{I}=0.5 are, respectively, 92% and 98% of the value for p_{I}=0. The above percentages have the following trend for varying λ^{a}: the values of $P_{\mathrm{F}}(t)$ for p_{I}=1 are 92%, 65%, 67%, 71% and 97% of the values of $P_{\mathrm{F}}(t)$ for p_{I}=0;

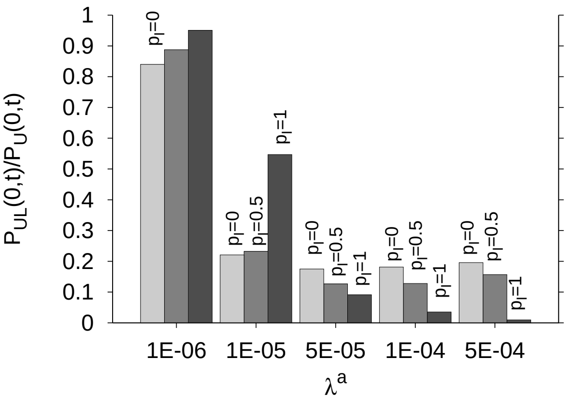
the values of $P_F(t)$ for p_I=0.5 are abput 98% of those of $P_F(t)$ for p_I=0. Looking at the values of $P_U(0,t)$ for the smallest λ^a, Figure 3.19(b) shows that the values for p_I=0 and p_I=0.5 are, respectively, 85% and 93% of the value for p_I=1. The above percentages have the following trend for varying λ^a: 85%, 13%, 5%, 5% and 8% if p_I=0, 93%, 24%, 10%, 10% and 13% if p_I=0.5.



(a) System failure probability $P_F(t)$



(b) System unavailability $P_U(0,t)$



(c) Impact of leader omissions on system unavailability $P_U(0,t)$

Figure 3.19: Impact of attack (intrusion) rate λ^a over system failure probability $P_F(t)$ and system unavailability $P_U(0,t)$ for different values of p_I

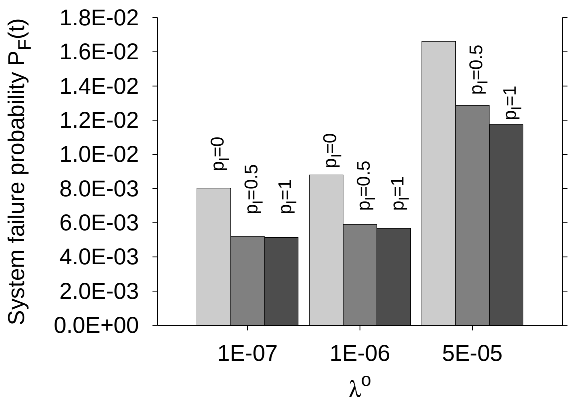
Figure 3.19(c) plots the impact (in percentage) $P_{UL}(0,t)$ of leader omissions on system unavailability; the impact of the leader omission decreases as successful attack (intrusion) rate λ^a increases. The shape of $P_{UL}(0,t)$ for varying p_I changes as successful attack (intrusion) rate λ^a increases: for lower values of λ^a $P_{UL}(0,t)$ has the largest value if p_I=0, whilst the opposite happens for larger values of λ^a.

Figures 3.19(a) and 3.19(b) confirm the intuition that the attack rate deeply impacts on the system measures of interest and that the larger number of reactive recoveries triggered for increasing values of p_I positively impact on both the measures of interest. Figure 3.19(c) confirm that for increasing attack rate the impact of leader omission on system unavailability decreases, and hence that the main cause of system omission is the incapability of reaching quorums.

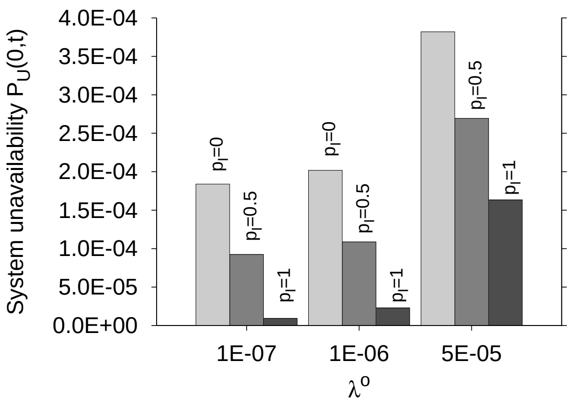
A study was devoted to evaluate the impact of omission rate λ^{o} over system failure probability $P_{\text{F}}(t)$ and system unavailability $P_{\text{U}}(0,t)$ for different values of p_{I} in order to better understand the impact of performing "delayed" reactive recoveries on the leader replica only.

Figures 3.20(a) and 3.20(b) show that in general both the measures of interest increase as the omission rate λ^{o} increases. The mean increment for $P_{\text{F}}(t)$ when moving from λ^{o}=10^{-7} to λ^{o}=10^{-6} is about 9%, whilst the increment when moving from λ^{o}=10^{-6} to λ^{o}=10^{-5} is about 88%. The mean increment for $P_{\text{U}}(0,t)$ when moving from λ^{o}=10^{-7} to λ^{o}=10^{-6} is about 10%, whilst the increment when moving from λ^{o}=10^{-6} to λ^{o}=10^{-5} is about 89%. It is worth to recall that this study was performed using the default setting for the successful attack (intrusion) rate λ^{a} (see table 3.3), that is a value of the order of 10^{-5}. The behavior of the two measures of interest with respect to varying the p_{I} is in general the following: the value of the measure of interest decreases as p_{I} increases.

Figure 3.20(b) shows the impact $P_{\text{UL}}(0,t)$ of leader omission over system unavailability $P_{\text{U}}(0,t)$; the impact of leader omission increases as λ^{o} increases, spanning from 17%, 21% and 40% for p_{I}=0 to 13%, 49% and 71% for p_{I}=1.



(a) System failure probability $P_{\text{F}}(t)$



(b) System unavailability $P_{\text{U}}(0,t)$

Figure 3.20: Impact of omission rate λ^{o} over system failure probability $P_{\text{F}}(t)$ and system unavailability $P_{\text{U}}(0,t)$ for different values of p_{I}

The last study was performed by defining some variants of the reconfiguration strategy, in order to better evaluate the role of the different recovery actions over the measures of interest. In particular, the following recovery strategies were evaluated:

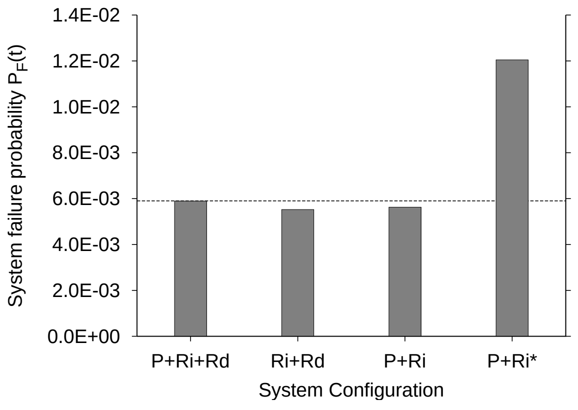
$\mathbf{P+R_i+R_d}$ This is exactly the PRRW strategy, where the following recovery actions are performed: proactive (P), reactive "immediate" (R_i) and reactive "delayed" (R_d). The recovery actions are triggered based on the rationale presented in Section 3.2.2.

$\mathbf{R_i+R_d}$ This recovery strategy triggers only "immediate" and "delayed" recovery actions (R_i and R_d respectively); "delayed" recovery actions are performed during recovery slots disciplined as in PRRW. This strategy does not trigger proactive recovery actions.

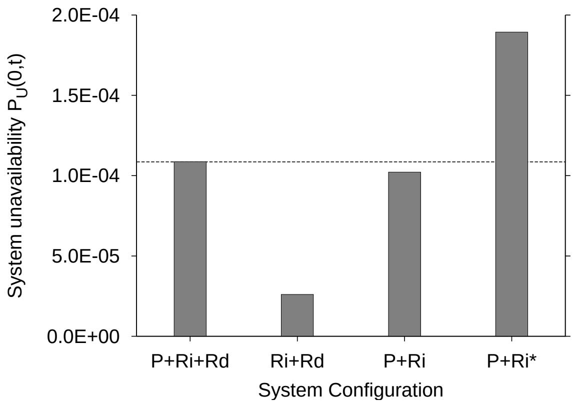
P+R$_i$ This strategy is a variant of the PRRW strategy where the following recovery actions are performed: proactive (P) and "immediate" reactive (R$_i$). "Delayed" recovery actions are not performed.

P+R$_i$* This strategy is a variant of PRRW where the only difference involves the triggering of the recovery actions on the leader suspected of being omissive: PRRW triggers a "delayed" reactive recovery on the omissive leader, whilst this strategy triggers an "immediate" reactive recovery. This strategy hence triggers proactive recoveries on all replicas (based on the rationale presented in Section 3.2.2) and "immediate" reactive recoveries both on replicas detected of being compromised, and on leader replicas suspected of being omissive.

Figures 3.21(a) and 3.21(b) show the values of system failure probability $P_F(t)$ and system unavailability $P_U(0,t)$ for the recovery strategies presented above. A sketched line was plotted to help the reader in comparing the reference value obtained for the PRRW strategy (left-most bar, labelled as "P+R$_i$+R$_d$") and the values obtained for the other recovery strategies.



(a) System failure probability $P_F(t)$



(b) System unavailability $P_U(0,t)$

Figure 3.21: System failure probability $P_F(t)$ and system unavailability $P_U(0,t)$ for different reconfiguration strategies

The evaluations were performed using the default values for the models parameters, in particular setting t=2628, c_M=0.7 and p_I=0.5 (see Table 3.3 for more details). It is worth recalling that PRRW strategy triggers "delayed" recovery actions on the leader replica only.

Figure 3.21(a) shows that the value of $P_F(t)$ for the second, third and fourth strategy is 94%, 95% and 204% of the reference value, respectively. Similarly, figure 3.21(b) shows that the value of $P_U(0,t)$ for the second, third and fourth strategy is 24%, 94% and 174% of the reference value, respectively.

The second strategy "R$_i$+R$_d$" (PRRW deprived of proactive recovery actions) shows to be better than all the other strategies, both for system failure probability and system unavailability. In particular, the value of system unavailability $P_{\mathrm{U}}(0, t)$ for the "$\mathrm{R_i}$+$\mathrm{R_d}$" strategy is 24% of the corrsponding value for the fist reconfiguration strategy (PRRW), showing that proactive recovery actions play a relevant role in negatively impacting on system unavailability (there are many recovery actions performed on correct replicas).

The fourth strategy "P+$\mathrm{R_i}$*" (PRRW where the omissive leader is "immediately" reactively recovered) shows to be worse than all the other strategies, both for system failure probability and system unavailability; in particular, the fact that it is worse than the fist one (PRRW) shows that the reactive recovery actions triggered on the omissive leader replica have a relevant impact on both the measures of interest (204% and 174% respectively). This assertion is supported also by the comparison between the third strategy "P+$\mathrm{R_i}$" (PRRW deprived of the reactive recovery actions on the omissive leader) and PRRW, where the impact of triggering delayed reactive recoveries on the omissive leader is of about 5% on both the measures of interest. We argue that increasing the accuracy of the diagnosis of omission faults could reduce the penalty due to reactive recovery actions performed on a correct leader showing omissive behaviour because of network omissions (the network omissions lasts for λ^{eo}=30 seconds on the average, whilst the replica recovery lasts for T_{D}=146 seconds).

4 Assessment of interdependencies between EI and II

A major research line of the project focuses on the development of a model-based methodology for the dependability analysis of the power grid information infrastructures. In D16 [20] we have developed a modeling framework for the analysis of interdependencies in Electric Power Systems, which is aimed at building generic models of interdependencies, taking into account the various forms of interactions and coupling the different systems and infrastructures to be considered in the models. Specifically, the conceptual modeling framework is well suited: i) to characterize and analyze the interdependencies between the information infrastructure and the controlled power infrastructure, especially the various types of failures that can occur in the presence of accidental and malicious faults, and ii) to assess their impact on the resilience of these infrastructures with respect to the occurrence of critical outages.

In this chapter we apply such modeling framework to selected critical control scenarios, to demonstrate its feasibility and practical utility. The structure of the chapter is the following. For the sake of completeness and for a better understanding of the rest of the chapter, in Section 4.1 the abstraction level of the two considered infrastructures is briefly sketched. Then the focus moves to show how the analysis can be performed in concrete case studies. In particular, two typologies of analysis are presented. Section 4.2 details the timing evolution of the electric grid under selected failure conditions and reconfiguration actions performed by II, with reference to a simple but expressive grid topology. This kind of analysis allows to both trace the propagation of failures and the triggering of related phenomena along time, and to validate the method and models in simple, well understandable situations. Section 4.3 focuses on the analysis of the Scenario number 2 defined in Work Package 2 (deliverable D2). to quantify the effects of a few failure scenarios on blackouts related indicators. This kind of analysis allows to understand the relative impact of involved failure/repair processes and the criticality of the electric grid elements.

4.1 Logical scheme of EPS

The EPS that we consider are, for the time being, limited to a homogeneous region of the transmission grid and to the corresponding regional control system. The logical structure is depicted in Fig. 4.1.

In the bottom part of Fig. 4.1 we can see the main elements that constitute the overall electric infrastructure, and thus in particular a region of the transmission power grid: *generators* (N_G components), *substations* (N_S components), *loads* (N_L components) and *power lines* (A_L components, which also logically include breakers and protections connected to the power lines). The energy produced by the generators is adapted by transformers, to be conveyed with minimal dispersion, to the different types of end users (loads), through different power grids. The power lines are components that physically connect the substations with the power plants and the final users, and the substations are structured components in

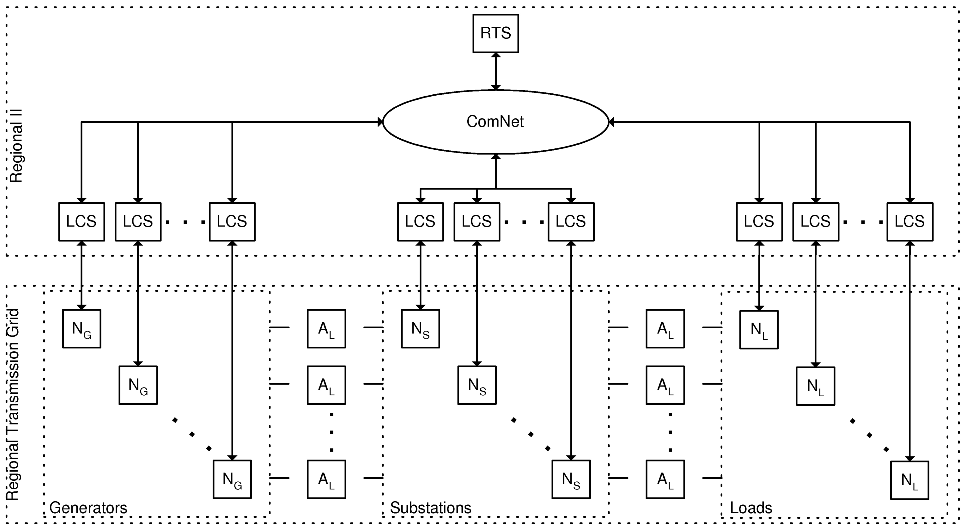


Figure 4.1: Logical structure of the analysed EPS instance

which the electric power is transformed and split over several lines. In the substations there are transformers and several kinds of connection components (like bus-bars, protections and breakers).

Some simplifying assumptions have been made to represent the power flow through the transmission grid, following the same approach used in [26, 15, 9, 3]. Therefore, the state and the evolution of the transmission grid are described by the active power flow F on the lines and the active power P at the nodes (generators, loads or substations), which satisfy linear equations for a direct current DC load flow approximation of the AC system.

The Information Infrastructure implements the information control system managing the electrical grid. Among the several logical components composing II (all detailed in [10]), here we focus the attention on the tele-operation system for a region of the transmission grid (named $TTOS$), since its failure can affect a large portion of the grid, also leading to black-out phenomena. In the upper part of Fig. 4.1 we have depicted a possible logical structure of a regional II, i.e., the part of the information control system controlling and operating on a region of the transmission grid. The components LCS (Local Control System) and RTS (Regional Tele-control System) differ for their criticality and for the locality of their decisions, and they can exchange grid status information and control data over a (public or private) network ($ComNet$ component). LCS guarantees the correct operation of a node equipment and reconfigures the node in case of breakdown of some apparatus. They include the acquisition and control equipment (sensors and actuators). RTS monitors its assigned region in order to diagnose faults on the power lines. In case of breakdowns, it chooses the most suitable corrective actions to restore the functionality of the grid. Since RTS is not directly connected to the substations, the corrective actions to adopt are put in operation through the pertinent LCS.

II controls the correct functioning of EI and activates proper reconfigurations in case of failure of, or integration of, repaired/new EI components. Such operations are not considered in detail but they are abstracted at two levels, on the basis of the locality of the EI state considered by II to decide on proper reactions to disruptions (the same approach adopted in [26]). Each level is characterised by an activation condition (that specifies the events that enable the II reaction), a reaction delay (representing the overall computation and application time needed by II to apply a reconfiguration) and a reconfiguration strategy (RS), based on generation re-dispatch and/or load shedding. The reconfiguration strategy RS defines how the configuration of EI changes when II reacts to a failure. For each level, a different reconfiguration function is considered:

- $RS_1()$, to represent the effect on the regional transmission grid of the reactions of II to an event that has compromised the electrical equilibrium[1] of EI, when only the state local to the affected EI components is considered. $RS_1()$ is performed by LCS components and, because of the limited information necessary to issue its output, it is fast in providing its reaction.
- $RS_2()$, to represent the effect on the regional transmission grid of the reactions of II to an event that has compromised the electrical equilibrium of EI, when the state global to all the EI system under the control of II is considered. Therefore, differently from $RS_1()$, $RS_2()$ is determined on the global EI state and reacts in a longer time. When new events occur changing the status of EI during the evaluation of $RS_2()$, then the evaluation of $RS_2()$ is restarted based on the new topology generated by such events. $RS_2()$ is performed by RTS.

The activation condition, the reaction delay and the definition of the functions $RS_1()$ and $RS_2()$ depend on the policies and algorithms adopted by $TTOS$. An autoevolution function $AS()$ is also considered to represent automatic evolution of EI each time an event modifying the grid topology occurs. In this case, EI tries to find a new electrical equilibrium for the new grid topology, by changing the values of the power flow through the lines but leaving the generated and consumed power unchanged (only redirection of current flows). If a new equilibrium is not reached, LCS and RTS operations, i.e. $RS_1()$ and $RS_2()$ respectively, are triggered. In the current implementation, the output of $RS_1()$ is obtained by the solution (values for active power vectors P and F) of power flow equations while minimizing a simple cost function, indicating the cost incurred in having loads not satisfied and having the generators producing more power. The output values of $RS_2()$ for P and F are derived by solving an optimization problem to minimis the change in generation or load shedding, considering more sophisticated system constraints, as described in [26]. The reconfiguration strategy $RS_1()$ is applied immediately, while $RS_2()$ is applied after a time needed by RTS to evaluate it. All these functions are based on the state of EI at the time immediately before the occurrence of the failure.

[1]Events that impact on the electrical equilibrium are typically an EI component's failure or the insertion of a new/repaired EI component; for simplicity, in the following we will mainly refer to failures.

4.2 Detailed analysis of EI state evolution

In this section we start presenting the framework's capabilities to analyze important aspects of interdependencies between EI and II infrastructures. In particular, the goal of this section is to show how the EI state can evolve in response to EI failure events, or determined by the applications of the autoevolution and LCS/RTS reconfiguration functions, and to analyze how its evolution may impact on the black-out related indicators.

We consider a simple testbed grid with 2 generators (numbered from 0 to 1), 3 substations (numbered from 2 to 3), 3 loads (numbered from 4 to 6) and 7 transmission lines, as shown in Figure 4.2. The label associated to the generators represents the initial

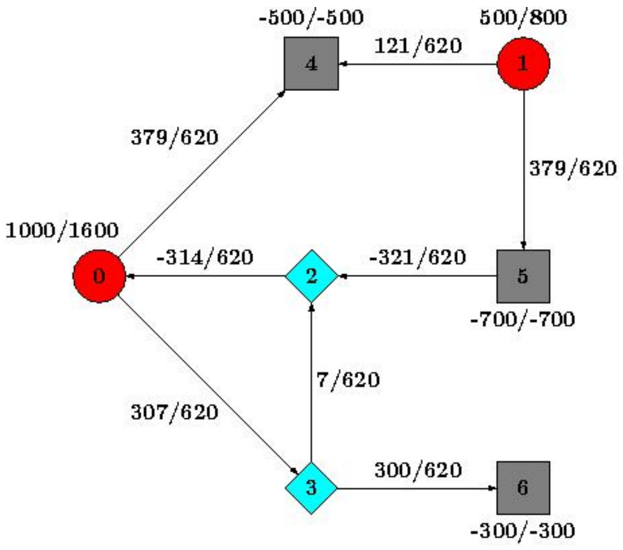


Figure 4.2: Diagram of the EI grid. Generators are red circles, loads are gray squares and substations are cyan rhombi

(active) power and the maximum power that a generator can supply ("P_i/P_i^{max}"). The label associated to the loads represents the power demand of a load ("P_i"). The label associated to the lines represents the initial power flow through the line and the maximum power flow that a transmission line can carry ("F_{ij}/F_{ij}^{max}").

The measure of interest we consider is the percentage $P_{\mathrm{UD}}(0,t)$ of the power demand that is not met in the interval $[0,t]$ (the symbol "UD" stands for "Unsatisfied Demand"). It is a measure of blackout size and can be obtained as the load shed (i.e., the not served power due to a load shedding) divided by the power demand (see [10] for the detailed definition of rate reward function).

In this preliminary analysis we evaluate the impact on $P_{\mathrm{UD}}(0,t)$ of the omission failure affecting the communication network (*ComNet* component of Figure 4.1) when a failure of one transmission line is occurred. The main model parameters and the default values used for the evaluations, unless explicitly specified, are shown in Table 4.1; the values are arbitrary, but reasonable. The external failure rate $\lambda_{0,4}^{AEx}$ of the line $(0,4)$ is 0.01. The probability $p_{0,4}^{\text{AExP}}$ that this failure is permanent is 1.

Table 4.1: Parameters and their default values

Name	Default Value	Meaning
α	1	Power grid stress factor, characterizing different power demand, generated power and power flow through the line
P_i^{max}	$[600, 800]$	The maximum power that a generator i can supply
P_i^D	$[300, 700]$	The power demand associated to the load i
F_{ij}^{max}	600	The maximum power flow that a transmission line can carry (without an overload)
b_{ij}	1	The suceptance of the line ij
T_{RS_1}	0	Time to evaluate $\mathcal{RS}_1()$
T_{RS_2}	10 m	Time to evaluate $\mathcal{RS}_2()$
$\lambda_{ij}^{\text{AEx}}$	10^{-7}	Rate of occurrence of an external failure (excluded lightning) on the line ij
p_{ij}^{AExP}	0.3	Probability that the external failure (excluded lightning) of the line ij is a permanent failure
T_{AR}	24 h	Time to repair a line
T_{NR}	24 h	Time to repair a node
λ_{LCS}	10^{-7} s^{-1}	Failure rate of `LCS`
p_{LCSP}	0.05	Probability that the failure of LCS is a permanent failure
p_{LCSO}	1	Probability that a failure of LCS is an omission failure
T_{LCSR}	18 h	Time to recover `LCS`
λ_{RTS}	10^{-8} s^{-1}	Failure rate of `RTS`
p_{RTSP}	0.05	Probability that the failure of RTS is a permanent failure
p_{RTSO}	1	Probability that a failure of RTS is an omission failure
T_{RTSR}	12 h	Time to recover `RTS`
$MTTF_{\text{CNET}}$	100 h	Mean time to omission failure of the network $TSOcomNet$
p_{CNETP}	1	Probability that the failure of the network $TSOcomNet$ is a permanent failure
p_{CNETO}	1	Probability that the failure of the network $TSOcomNet$ is an omission failure
$MTTR_{\text{CNET}}$	10 h	Mean time to repair of the network $TSOcomNet$
T_{m}	293 K	Temperature of the medium (external temperature)

Figures 4.3 and 4.4 show how $P_{\text{UD}}(0, t)$ varies as a function of $MTTF_{CNET}$ for different values of $MTTR_{CNET}$ and as a function of $MTTR_{CNET}$ for different values of $MTTF_{CNET}$, respectively, with $t = 48$ h. For the considered setting, variations of the values of $MTTF_{CNET}$ impact significantly on $P_{\text{UD}}(0, t)$ only when the value of $MTTR_{CNET}$

is greater than about 1 hour. For values of $MTTR_{CNET}$ less than about 1 hour the variations of $MTTF_{CNET}$ do not impact on $P_{\mathrm{UD}}(0,t)$. For values of $MTTR_{CNET}$ greater than about 1 hour, the impact of $MTTR_{CNET}$ on $P_{\mathrm{UD}}(0,t)$ depends significantly on the values of $MTTF_{CNET}$. Now we show how the sequence of events generated during the

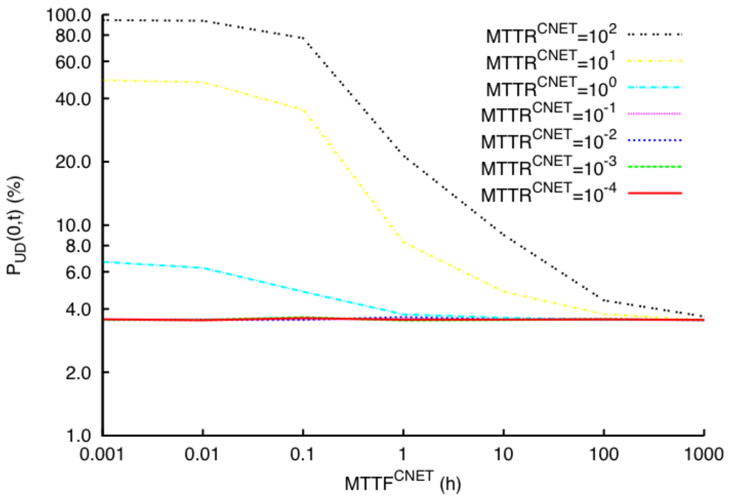


Figure 4.3: Percentage of the power demand that is not met in the interval $[0,t]$, with $t = 48$h, as a function of $MTTF_{CNET}$ for different values of $MTTR_{CNET}$

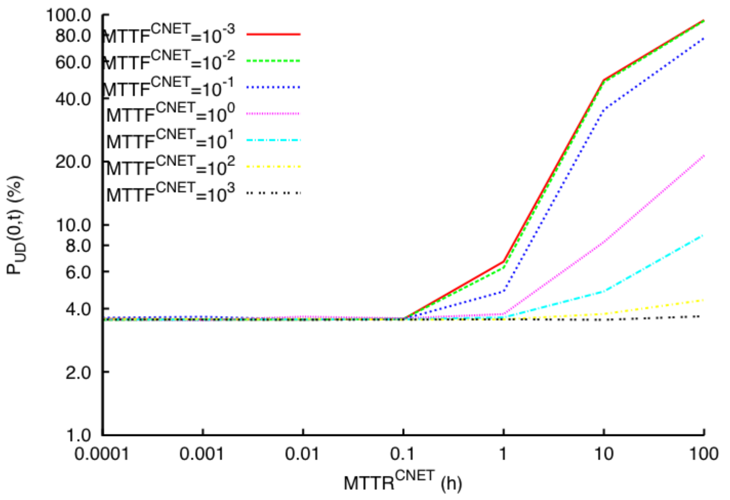


Figure 4.4: Percentage of the power demand that is not met in the interval $[0,t]$, with $t = 48$h, as a function of $MTTR_{CNET}$ for different values of $MTTF_{CNET}$

simulation of the overall EPS model impact on $P_{\mathrm{UD}}(0,t)$. We consider an external permanent failure occurring on the line $(0,4)$ in about a minute and no failures of the network *ComNet*. Table 4.2 shows the sequence of events occurring during a simulation batch with their occurrence times.

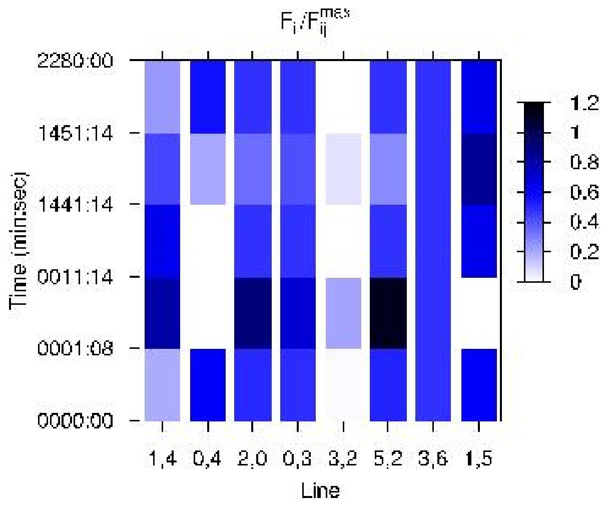
Figure 4.5(a) shows the evolution in time (for the simulation batch) of the ratio $|F_{ij}|/F_{ij}^{max}$ in the interval $[0,t]$, with $t = 48h$ (2 days), for all the transmission lines (on

Occurrence time (min:s)	**Event**
0000:00	- Start of the simulation
0001:08	- External permanent failure of line $(0, 4)$
	- Protections of line $(0, 4)$ fire
	- Line $(0, 4)$ opens
	- Autoevolution $AS()$ is applied $\Rightarrow$ new EI state generated
	Line $(5, 2)$ is overloaded ($F_{5,2} = -699.99$MW)
	Line $(5, 2)$ will fails at time 120:51 (min:s) if the associated protections will not fire
0011:14	- Reconfiguration strategy $RS_2()$ is applied by $RTS \Rightarrow$ new EI state generated
	Overload of line $(5, 2)$ is removed, $F_{5,2} = -300$
	The value of $P_4^{UD}(0, t)$ changes from 0 to 0.2
1441:14	- Repair of line $(0, 4)$
	- Protections of line $(0, 4)$ are closed
	- Autoevolution $AS()$ is applied $\Rightarrow$ new EI state generated
1451:14	- Reconfiguration strategy $RS_2()$ is applied by $RTS \Rightarrow$ new EI state generated
	The same EI state as at time 0
2280:00	- End of the simulation

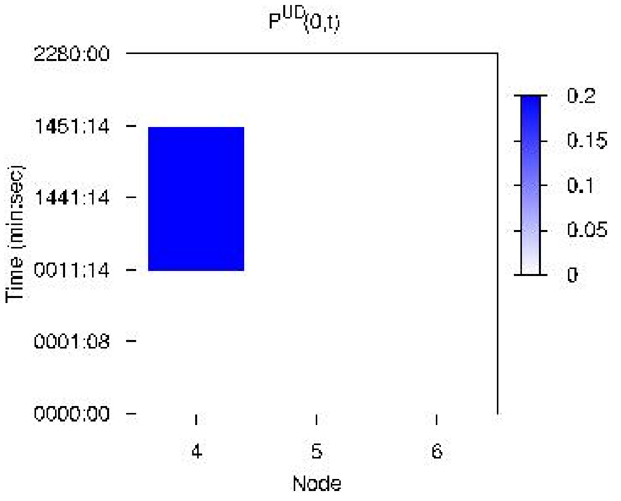
Table 4.2: Sequence of events occurring during a simulation and occurrence times (min:s)

the x-axis). The time since the start of the simulation is on the y-axis. The color of each vertical bar represents the loading of the corresponding line in a given state. The value is normalized with respect to F_{ij}^{max}, so white means zero power flow, while black, representing values greater than 1, means overload.

Figure 4.5(b) shows the evolution in time (for the simulation batch) of the percentage $P_i^{UD}(0, t)$ of the power demand that is not met in the interval $[0, t]$, with $t = 48h$ (2 days), for all the loads (on the x-axis). The color of each vertical bar represents the percentage of power demand of the corresponding load that is not met in a given state. White means that all demand is met, blue represents the maximum demand that is not met (0.2 in this case). The vertical bars associated to loads 5 and 6 are white (and so they are missed), because the power demand is always met in the considered simulation batch.



(a)



(b)

Figure 4.5: (a) F_{ij}/F_{ij}^{max} evolution in the interval $[0, 48h]$, and (b) $P_i^{UD}(0, 48)$ evolution

4.3 Analysis of critical control scenarios: the case of failures of EI power lines and DoS attack affecting the II infrastructure

In this section we will apply the framework for the analysis of Scenario 2 detailed in D2. We first need to define a mapping between the components defined in Scenario 2 and those belonging to the modeling framework. RTS corresponds to the TSO CC of the scenario, LCS corresponds to the MCD-TU, and TSO/DSO communication networks are considered as a single network. Note that the concept of area control center (TSO and DSO ACC of the scenario) is not explicitely modelled. It will be part of the control functions RS_1 and RS_2 to determine which set of LCS are involved in a reconfiguration/load shedding action.

The analyzed electric power grid is depicted in Figure 4.6. The grid is a portion of

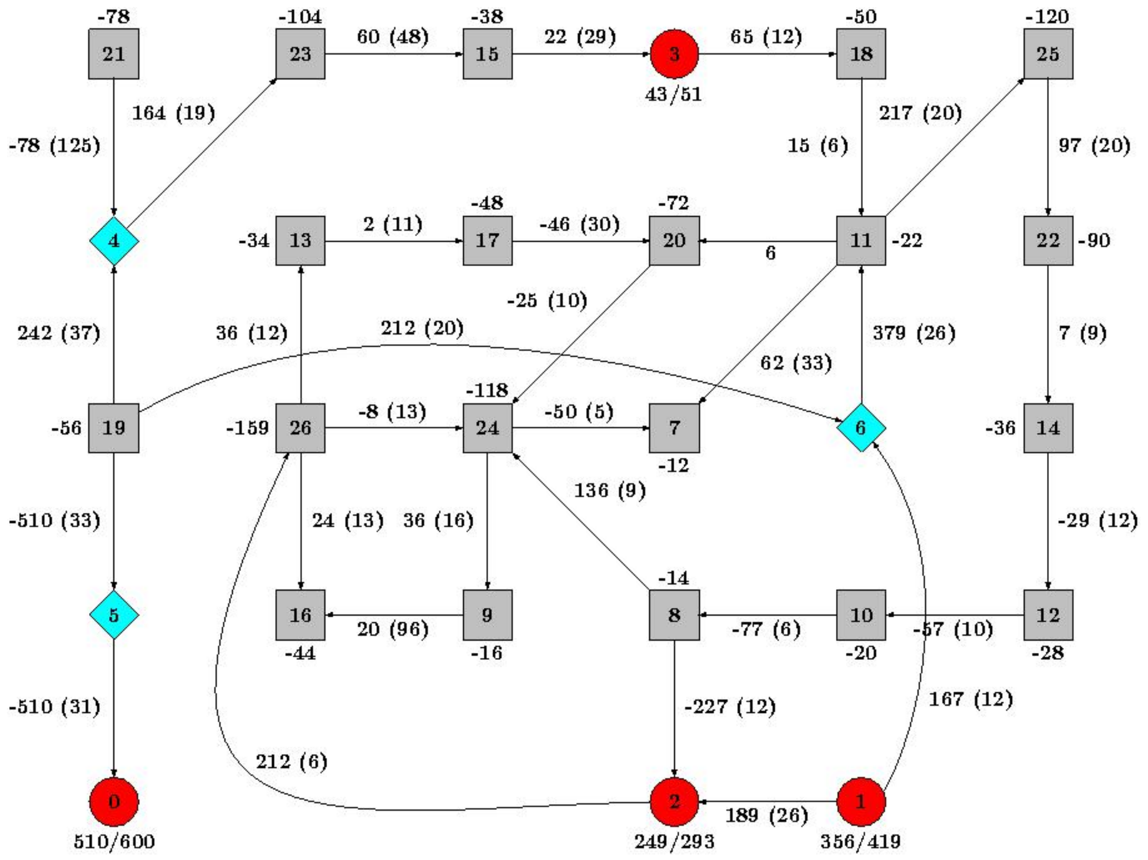


Figure 4.6: Diagram of the EI grid (generators are red circles, loads are gray squares and substations are cyan rhombi). For the sake of clarity, only the integer part of the original values associated to generators, power lines and loads are shown (in MegaWatt)

the IEEE 118 Bus Test Case[2], typically used in other studies related to EPS. The label associated to the generators represents the initial (active) power and the maximum power

[2]http://www.ee.washington.edu/research/pstca/pf118/pg_tca118bus.htm

that a generator can supply (“P_i/P_i^{max}”). The label associated to the loads represents the power demand of a load (“P_i”). The label associated to the lines represents the initial power flow through the line and the susceptance[3] (“F_{ij} (b_{ij})”). We suppose that each line can carry the same maximum power flow ($F_{ij}^{max} = 620$ MW for each i, j). In the initial grid setting all the ratios P_i/P_i^{max} correspond to a fixed value for the power grid stress level $\alpha = 0.85$. By varying α, other EI settings are automatically determined.

The measure of interest we consider is $P_{UD}(t, t+1)$, defined as the percentage of the mean power demand that is not met in the interval $[t, t+1]$ (the symbol “UD” stands for “Unsatisfied Demand”). It is a user-oriented measure of the blackout size and can be obtained as the load shed (i.e., the not served power due to a load shedding) divided by the power demand.

In this Section we aim to assess the impact of cyber interdependencies on the defined black-out related indicator. Among the possible interdependencies (ITCS failures affecting EI), we evaluate the impact on $P_{UD}(t, t+1)$ of the omission failure of the communication network (*ComNet* of Figure 4.1) when a simultaneous failure of a set of transmission lines occurred. This is a scenario inspired by those considered in the project CRUTIAL. More in detail, the EI state is initially set as depicted in Figure 4.6, and it is in electrical equilibrium. At time zero we suppose that n^{LF} power lines are simultaneously affected by a permanent disruption (e.g., due to a tree fall or a terrorist attack), thus becoming unavailable. The power lines that fail are randomly (uniformly) selected from the set of all available power lines. The repair time of the failed power lines is fixed to 24 hours. At the same time zero, the communication network *ComNet* connecting the LCS components to RTS is simultaneously affected by a denial of service (DoS) attack, thus impeding the LCS-RTS communication. Therefore, during a DoS attack, the reconfiguration strategy $RS_1()$ can be applied at any time, while the reconfiguration strategy $RS_2()$ cannot be applied. The DoS attack ends after an exponentially distributed time with mean $MTTR^{CNET}$, and from that time RTS can start computing the $RS_2()$ reconfiguration action that will be applied after 10 minutes. The considered distributions and values for failure, repair and reconfiguration processes do not refer to any specific real case; they are hypothetical but plausible ones and are used just for showing the potentialities of our analysis method. However, to take into consideration to some extent variations of assumed settings, we performed a sensitivity analysis on the following parameters:

- $MTTR^{CNET}$, thus varying the duration of the DoS attack affecting the communication network. If $MTTR^{CNET} \rightarrow \infty$, then we are modeling a RTS omission failure.
- n^{LF}, thus varying the severity of the overall EI failure.
- α, thus varying the initial stress level of the power grid.

[3]The susceptance is used to determine the values for the power flow through the lines.

4.4 Numerical Evaluations and Analysis of the Results

In this section we present some of the results that we obtained through the solution of the overall model previously sketched. A transient analysis has been performed, using the simulator provided by the Möbius tool [13]. For each study we executed a minimum of 2000 simulation runs (batches), and we set the relative confidence interval to 0.1 and the confidence level to 0.95. This means that the stopping criteria will not be satisfied until the confidence interval is within 10% of the mean estimate in 95% of the times.

In Figure 4.7 we show the more critical power lines, i.e. those for which a failure produces high values of expected loss of delivered power in an interval of four days, $P_{UD}(0,t)$, with $t = 4$ days, for the duration of the DoS attack exponentially distributed with mean $MTTR^{CNET} = 24$ hours and $n^{LF} = 1$. They are, in decreasing order of criticity, $(19,4)$, $(19,5)$, $(5,0)$ and $(21,4)$. The failure of the other lines does not impact on $P_{UD}(0,t)$.

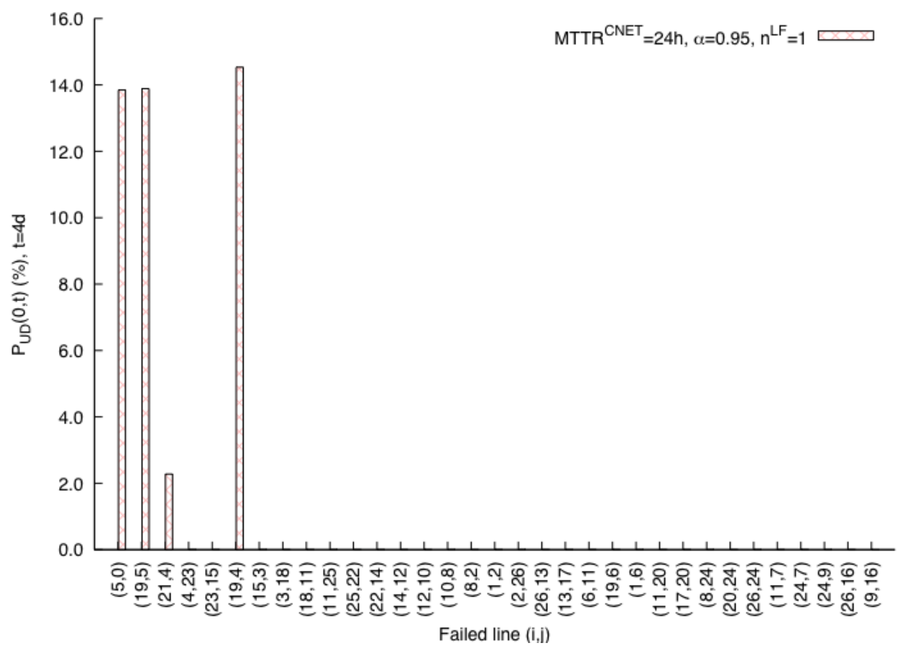


Figure 4.7: Percentage of the mean power demand that is not met in the interval $[0,t]$, with $t = 4$ days, for each power line affected by the failure, fixing $\alpha = 0.95$, $n^{LF} = 1$ and $MTTR^{CNET} = 24$ hours

In Figure 4.8 we show the $P_{UD}(t,t+1)$ variations as a function of time t (hours) for each power line affected by the failure (the power lines for which $P_{UD}(t,t+1) = 0$ are omitted), for the duration of the DoS attack exponentially distributed with mean $MTTR^{CNET} = 24$ hours and $n^{LF} = 1$.

In Figure 4.9 we show the $P_{UD}(t,t+1)$ variations as a function of time t (hours) for different durations of the DoS attack (exponentially distributed with mean $MTTR^{CNET} = 6$ or 24 hours), for a different number of simultaneous power line disruptions ($n^{LF} = 1$ or 2) and for different initial stress levels ($\alpha = 0.85$ or 0.95). We note that the failure of even

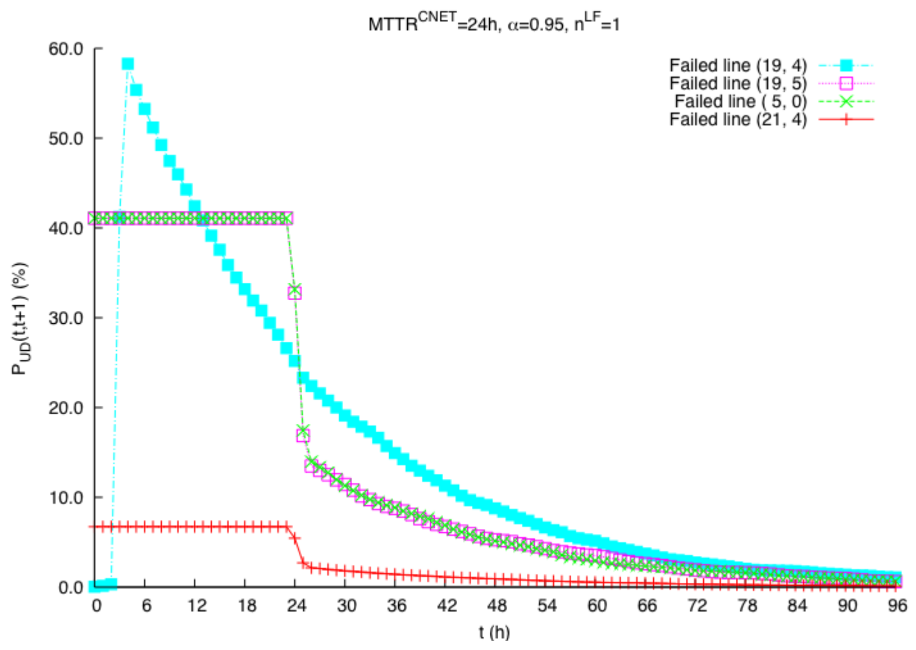


Figure 4.8: Percentage of the mean power demand that is not met in the interval $[t, t+1]$, with $t = 0, 1, \ldots, 96$ hours, for each power line affected by the failure, fixing $\alpha = 0.95$, $n^{LF} = 1$ and $MTTR^{CNET} = 24$ hours

a single random power line at time zero produces an immediate increment of $P_{UD}(t, t+1)$ greater than 2%. For $\alpha = 0.95$, the values of $P_{UD}(t, t+1)$ increase rapidly over time until the reconfiguration strategy $RS_2()$ is applied (i.e., 10 min. after the DoS attack ends). This is the effect of the cascading failures of the overloaded lines and of the too big variation of power demand to generators in a small interval of time. In fact, with a high value of the power grid stress $\alpha = 0.95$, the autoevolution function $AS()$ or the reconfiguration strategy $RS_1()$ triggered by the failure of even a single power line can produce overload of lines or stress of generators. On the contrary, with the lower stress level $\alpha = 0.85$, the failure of only one power line leads EI to reach a stable state that does not need a RTS reconfiguration (no shedding operations are needed), and $P_{UD}(t, t+1)$ remains constant in the interval $[0, 24]$. At $t = 24$ hours there is a big improvement due to the repair of the failed power lines and then the nominal conditions in the system are restored, with the consequent full satisfaction of the power demand after some time. It is worthwhile to note that the impact of the system stress level α is less heavy on the percentage of unsatisfied demand than the failure of power lines: e.g., the curve with $\alpha = 0.95$ and $n^{LF} = 1$ is better than the one with $\alpha = 0.85$ and $n^{LF} = 2$.

Figure 4.10 shows how $P_{UD}(t, t+1)$ varies as a function of time t (hours) for different durations of the DoS attack ($MTTR^{CNET} = 6$ or 24 hours) and for a different number of simultaneous power line disruptions ($n^{LF} = 1, 2, 3, 4$ or 5), fixing $\alpha = 0.95$. As expected, $P_{UD}(t, t+1)$ increases considering higher n^{LF} values, and fixing the value for n^{LF}, $P_{UD}(t, t+1)$ gets worse in the case in which the DoS attack has a longer duration (24 hours). In fact, if $MTTR^{CNET} = 6$, RTS can earlier apply the $RS_2()$ reconfiguration action (on average, after 6 hours and 10 min.), and then EI moves into a state less degraded than the state in which EI

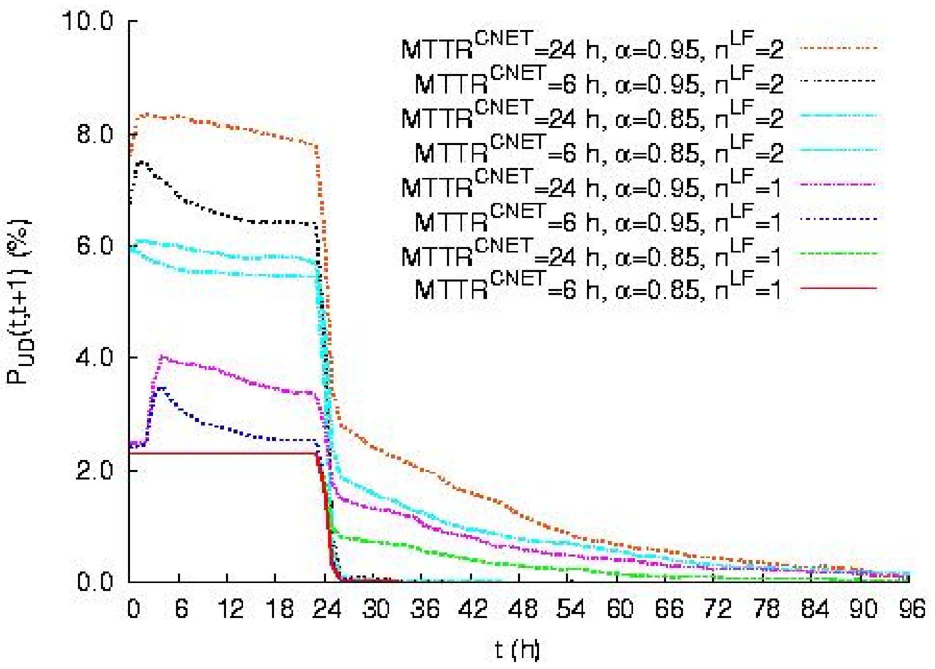


Figure 4.9: Percentage of the mean power demand that is not met in the interval $[t, t+1]$, with $t = 0, 1, \ldots, 96$ hours, for different values of $MTTR^{CNET}$, n^{LF} and $\alpha = 0.85, 0.95$

would be without considering the RTS reconfiguration. After 24 hours the disrupted power lines are repaired, and consequently $P_{UD}(t, t+1)$ rapidly decreases until reaching the zero value, since the original EI grid configuration (with all the loads satisfied) has been restored. The usefulness of applying the RTS reconfiguration can be really appreciated comparing all the plots with the first one, representing the case in which no RTS reconfiguration is performed (RTS omission failure).

In both Figures 4.9 and 4.10 we have provided mean values for the percentage of unsatisfied power demand in an interval $[t, t+1]$ for different values of t. In Figure 4.11 we show the discrete probability distribution function (PDF) of $P_{UD}(t, t+1)$ for different values of $t = 0, 5, 6, 23, 24$ hours, fixing $\alpha = 0.95$, $n^{LF} = 1$ and $MTTR^{CNET} = 24$ hours. Analyzing the corresponding plot in Figure 4.9 we see that the mean value of the percentage of the non delivered power in the interval $[0, 1]$ (first hour) is $P_{UD}(0, 1) \approx 2.5\%$. Analyzing its complete distribution in Figure 4.11, for $t = 0$, we note that: i) with a very high probability 0.9 the percentage of undelivered power is equal to zero; ii) $P_{UD}(0, 1)$ is in the interval $(0, 10]\%$ with a probability of about 0.03, and it is in the interval $(40, 50]\%$ with a probability of about 0.06; iii) all the other probabilities are almost zero. A mean loss of 40–50% of delivered power in the first hour of the system can happen, for example, when the power line affected by the failure is directly connected to a generator. This occurs when one of the power lines $(19, 5)$ or $(5, 0)$ is affected by the failure with probability $2/n_A$ (each line that fail is randomly uniformly selected), as shown in Figure 4.8. In fact, in the considered grid, only 4 power lines, i.e., $(19, 4)$, $(19, 5)$, $(5, 0)$ and $(21, 4)$, contribute to $P_{UD}(0, 1)$, as shown in Figure 4.7. On the contrary, no loss of delivered power occurs with probability $29/n_A = 0.88$ (each line that fail is randomly uniformly selected) when one of the other 29 power lines

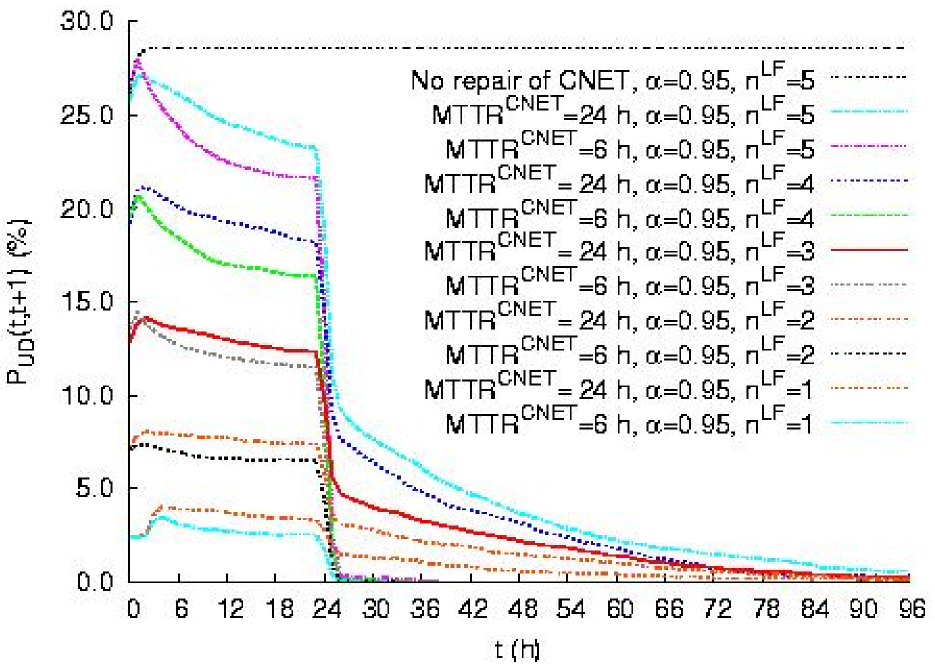


Figure 4.10: Percentage of the mean power demand that is not met in the interval $[t, t+1]$, with $t = 0, 1, \ldots, 96$ hours, for different values of $MTTR^{CNET}$, n^{LF} and $\alpha = 0.95$

fails. The other plots with $t = 5, 6, 23, 24$ hours have similar trends.

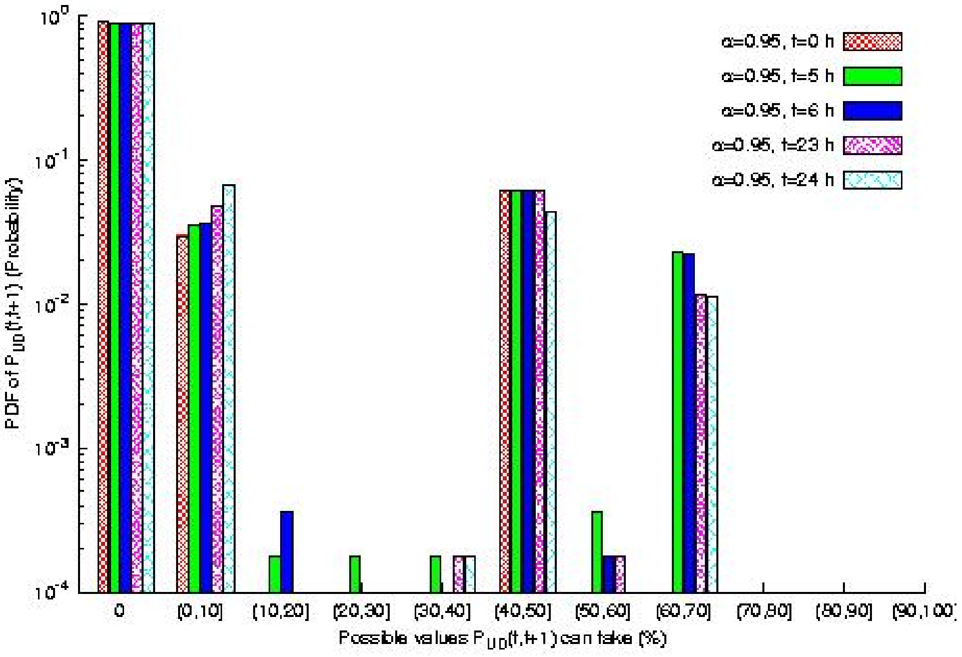


Figure 4.11: Probability that $P_{UD}(t, t+1)$ is equal to 0 or it is in the interval $(a, a+10]\%$, with $a = 0, 10, 20, \ldots, 90$, fixing $\alpha = 0.95$, $n^{LF} = 1$ and $MTTR^{CNET} = 24$ hours

5 Scenario 2: SWN models and joint SAN-SWN analysis

In this chapter we present an integrated approach to model and quantify (inter)dependencies between the Electrical Infrastructure (EI) and the Information Infrastructures (II) that implements the EI control and monitoring system. The quantification is achieved through the integration of two models: a Stochastic Well-formed Net (SWN) [11, 18] model model that captures in a rather detailed manner the protocol for arming and load schedding of Scenario 2 (in [17]) and the SAN model described in the previous chapter. This integrated approach is illustrated on Scenario 2, to study the effects of a II partial failure (a denial of service attack that compromises the communication network) on the remote control of the EI.

Chapter 4 already contains an analysis of the power grid upon a failure and when the ICT is under a DoS attack, but the model is rather abstract, so that the DoS attack is considered equivalent to a situationn in which all substations are unreachable due to DoS, while in this chapter we shall try to combine a reasonably detailed model of the Scenario 2 telecontrol operations protocol with the SAN model of the power GRID.

In this chapter we first recall, in Section 5.1, Scenario 2. In Section 5.2 we present the SWN model of the scenario that allows to quantify the impact of the DoS on the telecontrol, assuming as input data a characterisation of the EI behaviour. The SAN modelling approach has already been presented in Chapter 4 and in Section 5.3 we describe how the SAN and SWN model can interact to obtain a model of EPS that is correctly taking into adequate account both the II and EI behaviour.

5.1 The scenario considered

The scenario considered is scenario 2 of telecontrol operation under a DoS attack: it is a case in which a load shedding activity is needed to re-establish the EI working conditions upon an electrical failure, but the II is not working properly due to a Denial of Service (DOS) attack. In emergency conditions the TSO is authorised by the DSO to activate load shedding activities on the Distribution Grid. The information flows between TSO and DSO ICT components includes (see figure 5.1): a)**Measurements** from DSO Control Centre to TSO Control Centre; b)**Signals** from DSO Control Centre to TSO Control Centre; c) **Arming/unarming requests** from TSO Control Centre to DSO Control Centre; d)**Test packets** from TSO MCD-TU to DSO MCD-TU; e)**Actuation Commands**: substation trip from TSO MCD-TU to DSO MCD-TU

The TSO Regional Control Centre (RCC) monitors the Electric Power System and detects some potentially dangerous conditions that could be recovered with appropriate load shedding commands applied to particular areas of the Grid. In order to actuate this defence action the TSO Centre chooses a subset of HV/MV Substations (SSs) from the list of SSs

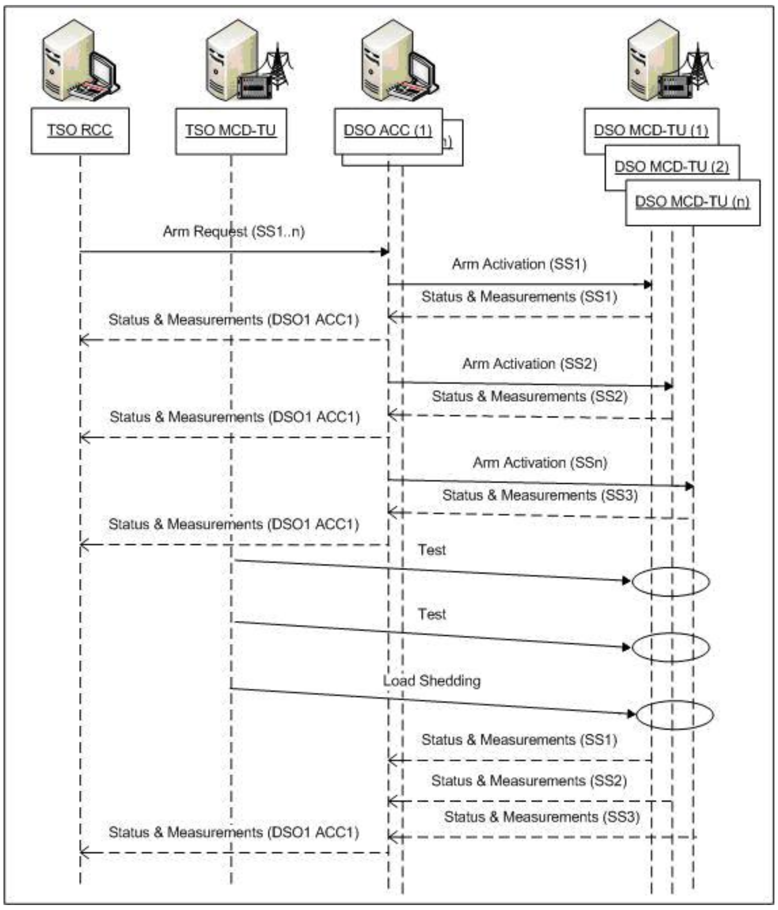

TSO RCC
TSO MCD-TU
DSO ACC (1)
DSO MCD-TU (1)
DSO MCD-TU (2)
DSO MCD-TU (n)
Arm Request (SS1..n)
Arm Activation (SS1)
Status & Measurements (SS1)
Status & Measurements (DSO1 ACC1)
Arm Activation (SS2)
Status & Measurements (SS2)
Status & Measurements (DSO1 ACC1)
Arm Activation (SSn)
Status & Measurements (SS3)
Status & Measurements (DSO1 ACC1)
Test
Test
Load Shedding
Status & Measurements (SS1)
Status & Measurements (SS2)
Status & Measurements (SS3)
Status & Measurements (DSO1 ACC1)

Figure 5.1: The information flow of the considered scenario

participating in the emergency plan, then sends the requests of preventively arming the Automation System (MCD-TUs) of these SSs to the interested DSO Area Control Centres (DSO ACCs). These requests are delivered through communication channels between TSO RCC and DSO ACC. The DSO ACC forwards the arm command to the required SSs, and returns their status to the TSO RCC. If the potential emergency condition evolves into a real emergency situation, the TSO MCD-TU (area sentinel) sends the load shedding command to all the DSO MCD-TU participating to the emergency plan, but only the substations that have been previously armed will be actually detached. TSO sentinels periodically send test packets toward detachable substations. If an armed substation does not receive three test packets in a row, it automatically disarms itself (after 1 minute in the scenario). Disarming also occurs after a fixed amount of time (20 minutes in the scenario) if no load shedding command is issued.

Depending on when the Substation DoS occurs in this scenario evolution, different behaviours may be envisaged.
1. A DoS attack starting before issuing the arming command towards a given substation creates the possibility of preventing the execution of that substations trip.
2. Should the DoS take place when the substation is armed, the attack denies the successful execution of the periodic testing and causes the consequent automatic disarming of the

substation.
3. Finally the DoS may occur just before issuing the substation trip thus denying the possibility of defending the system from extreme contingencies.

The effects on the EPS of the considered DoS will depend on the number of components attacked and on the pattern and intensity of the DoS. The identification of the dependencies of EI and II upon an electrical failure in presence of a DoS attack, and their quantification, is the aim of oue evaluation.

Testbeds have also been implemented in CRUTIAL to be able to perform measure for this scenario, but it is through modelling that both the behaviour of EI and II can be taken into account, while we can also consider EI and II topologies different from the one considered in the testbed.

5.2 The SWN model of Scenario 2

The SWN models focus on the ICT control system implemented by the II infrastructure. The model presented in the sequel represents the automation system behaviour under a DoS attack depicted in scenario 2. In Fig.5.2 a SWN model of the scenario is depicted: the model represents in an abstract way the event causing the unbalance in the EI that triggers the arming and the load shedding procedure (transitions *e-failure*, *StartArmingProcess*, *EndArmingProcess*, *LoadShedding*).

The TSO Control Center (TSO CC) is represented in an abstract way by two places and three transitions: $ProcInfo$ and $ReadyRqArming$ together with transitions $StartArmingProcess$, $EndArmingProcess$ and $TransmitArming$. A token in place $ProcInfo$ means that the TSO CC is analyzing the EI and elaborating some potential emergency conditions that could be fixed with suitable load shedding commands applied to particular areas of the EPS. When the defence actions have been elaborated (transition $StratArmingProcess$), TSO CC is ready to actuate them; this requires to send a request of preventively arming some SSs to the DSO CC (transition $TransmitArming$). The DSO CC sends the arming command to the appropriate substations (sequence $CC\_selection$, $ForwardedPacket$ and $TxDelay$): at this point the influence of the DOS attack (if it has started) is modeled by the choice between transitions $ForwardedPacket$ and $LostPacket$. The firing probability of these two transitions depends on the progress level of the DOS attack, represented by the marking of place $AttackSeverity$ (which in turn is modified by transition $IncreaseSeverity$ while place $Active$ is marked). If the packet is forwarded, its transmission time can still depend on the progress of the DOS attack. If the message is forwarded to the substation, the latter is armed (place SS_armed), moreover it sends back a message to the DSO CC (the return message is subject to the same risk of being lost in case a DOS attack is ongoing, modeled by transitions $LostPacket$ and $ForwardedPacket$).

When a perturbation in the EI is detected (place *Partial e-outage* marked), a load shedding command is sent from the TSO SS (called TSO sentinel) to (a subset of) the

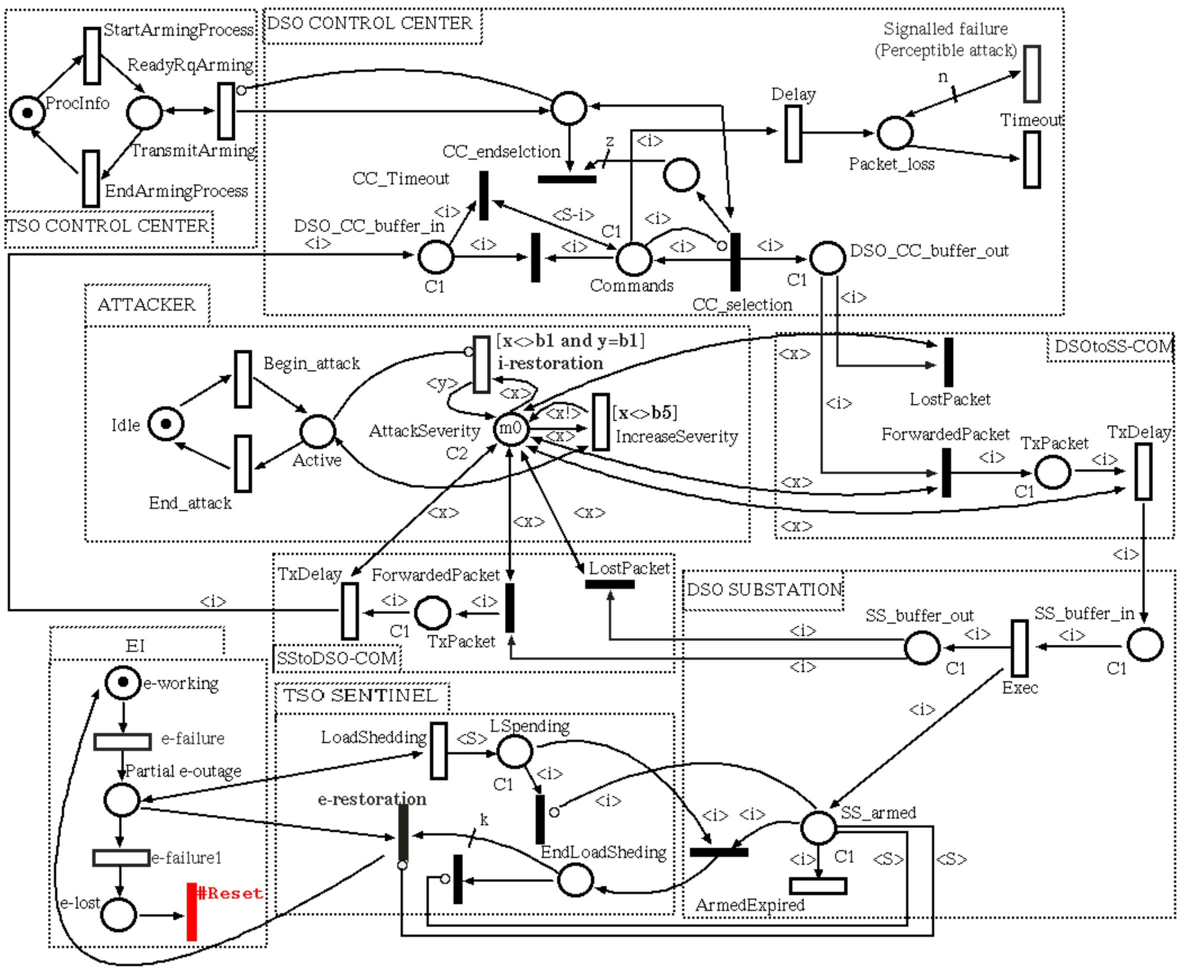


Figure 5.2: SWN reference model for scenario 2

armed distribution substations (transition *LoadShedding*): those that were armed, upon reception of the load shedding command perform the required procedure. If enough substations complete such procedure, the whole process ends successfully (firing of transition $e - restoration$) and the EI is brought back in a new balanced state (place *e-working*). An arming command not followed by any load shedding, expires after some time (transition *armedExpired*). The DOS attack is represented in an abstract way by three places and three transitions: places *Idle* and *Active* together with transitions *Begin_attack* and *End_attack* model the presence/absence of an attack and the transitions from one state to the other; place *AttackSeverity* on the other hand models the status of the connection towards the attacked site(s): when the attack is ongoing, the connection capacity degrades (transition *IncreaseSeverity*) while the absence of attack restores the status back to normal. The DSO control center may detect the fact that there is a communication problem with some substations when the acknowledge messages are not received within a certain time. If a long enough time elapses, the lack of an ack message is registered (place *Packetloss*): a packet loss registration is kept for a given amount of time, after which it is discarded. If enough packet loss registrations accumulate in a short time (transition *Timeout* removes old lost messages) then the presence of a failure is signalled (transition $Signalledfailure$). The latter part of the model is useful to compute performance indices related to the detection of a suspect behaviour. Observe that the transition *#Reset* in the EI submodel is a special transition, with the following semantics: when it is enabled its firing brings the model back to the initial marking: the relative throughput of this transition with respect to the throughput of the *TransmitArming* transition provides an indication of the percentage of unsuccessful load shedding activities under a DoS attack. The particular semantics of the reset transitions allows this measure to be computed in steady state.

For the SWN model of Fig. 5.2 to correctly represents the global behaviour of the EPS under a DoS attack, a number of parameters should be estimated, grouped in three sets:
1. parameters related to the“physical” scenario: how many substations, communication network between them, number of substations controlled by an area control center (and identification of the substations involved in an arming command or a load shedding one).
2. parameters related to EI: distribution or time value from arming and consequent load shedding command, and from arming to unarming. These times depend on the electrical state.
3. parameters related to the II: distribution or time value related to the communication channel (e.g the package delay, probability of loosing a packet) and to DoS attack (duration and severity of the attack).

Given a scenario the first set of parameters is fixed, the third set could be part of a sensitivity analysis performed with the SWN model, while the second set depends from the first set in a way that is not explicitely modeled in the SWN model (since it requires to model the electrical state and its evolution).

5.3 Models interaction

Before discussing the interaction between SAN and SWN we recall how the SAN model takes into account the control of the GRID through the two reconfiguration functions. II controls the correct functioning of EI and activates proper reconfigurations in case of failure of, or integration of, repaired/new EI components. Such operations are not considered in detail but they are abstracted at two levels, on the basis of the locality of the EI state considered by II to decide on proper reactions to disruptions (the same approach adopted in [26]). Each level is characterised by an activation condition (that specifies the events that enable the II reaction), a reaction delay (representing the overall computation and application time needed by II to apply a reconfiguration) and a reconfiguration strategy (RS), based on generation re-dispatch and/or load shedding. The reconfiguration strategy RS defines how the configuration of EI changes when II reacts to a failure. For each level, a different reconfiguration function is considered:
a. $RS_1()$, to represent the effect on the regional transmission grid of the reactions of II to an event that has compromised the electrical equilibrium[1] of EI, when only the state local to the affected EI components is considered. $RS_1()$ is performed by LCS components and, because of the limited information necessary to issue its output, it is fast in providing its reaction.
b. $RS_2()$, to represent the effect on the regional transmission grid of the reactions of II to an event that has compromised the electrical equilibrium of EI, when the state global to all the EI system under the control of II is considered. Therefore, differently from $RS_1()$, $RS_2()$ is determined on the global EI state and reacts in a longer time. When new events occur changing the status of EI during the evaluation of $RS_2()$, then the evaluation of $RS_2()$ is restarted based on the new topology generated by such events. $RS_2()$ is performed by RTS. The reconfiguration strategy $RS_1()$ is applied immediately, while $RS_2()$ is applied after a time needed by RTS to evaluate it. All these functions are based on the state of EI at the time immediately before the occurrence of the failure. While each single model in isolation can be considered as a model of the behaviour of an EPS whose communication infrastructure has been attacked by a DoS, it is quite clear that there are a number of simplifying hypothesis behind.

The SWN model assumes that the load shedding command is issued whenever the EI is in a *Partial e-outage* state, but this is not always the case in reality, it depends on the state of the EI, similarly, the model assumes that the e-restoration can take place or not depending on the number of armed substations, and that any restoration is successful: again this depend on the state of the power grid, that is not included in the model. Nevertheless the modelling of the arming and successive load shedding command by the sentinel, in presence of a DoS, represents quite faithfully the scenario's behaviour.

For what concerns the SAN model there are also a number of discrepancies with respect to the considered scenario: there is no explicit modelling of the Distribution grid

[1]Events that impact on the electrical equilibrium are typically an EI component's failure or the insertion of a new/repaired EI component; for simplicity, in the following we will mainly refer to failures.

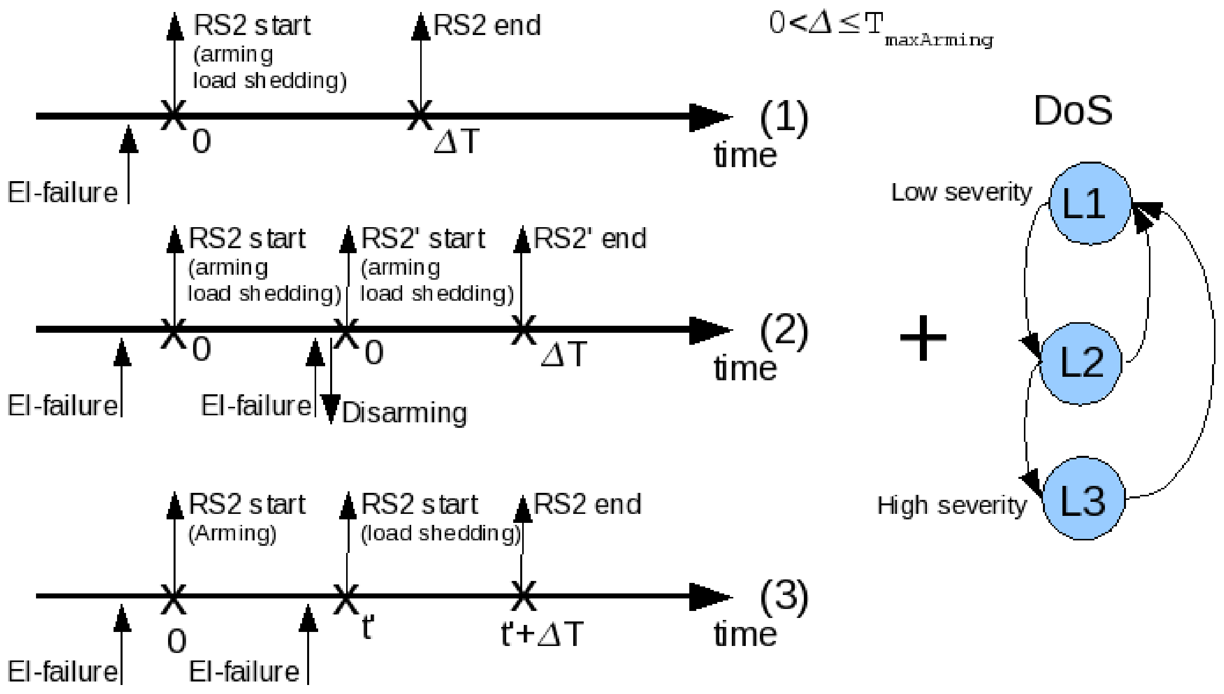


Figure 5.3: Timed evolution of the EI and II upon EI-failure and the DoS attack behaviour.

and its control. The control is a two level hierarchy (while it is three levels - Regional Control Center, Area Control Center and MCDTU of substations - in the scenario). Many of this information are taken into account directly by the reconfiguration function $RS_2()$, for example the reconfiguration can be computed on a limited portion of the grid, thus ”emulating” the concept of Area Control Center. The SAN model that has been presented in Chapter 4 can account for a loss of an electrical component, which is what triggers the computation of a new configuration of the electrical state of the grid through $RS_2()$. The SAN model can also account for a failure in the II in the following manner: the $RS_2()$ functions is computed on the subset of LCS that are estimated being reachable at the time $RS_2()$ is called. Therefore all types of II failures are modeled in terms of the number of LCS components available for a reconfiguration. In the SAN model this number is computed based on a probability of an LCS being reachable or not. The weak point is that this probability should be given, moreover an important characteristics of the DoS is that its behaviour changes over time, and this should be taken into account in the model.

Fig. 5.3 depicts the behaviour of the EI and II upon EI-failures (left) and the behavior of the DoS attack in terms of its severity levels (right), so that a DoS severity level is associated with each event in the time-line

The first time-line of Fig. 5.3depicts a case in which an EI-failure causes the start of the computation of $RS_2()$ at time 0, and, at time ΔT the computation terminates, the new reconfiguration is applied. If the reconfiguration was adequate a stable state is reached, if not, some electrical component will disconnect due to the local protections and a new $RS_2()$ is computed, leading potentially to a load shedding request. Observe that the reconfiguration success depends also on the DoS attack severity.

The second time-line shows instead a case in which between time 0 and ΔT a second EI failure takes place. In this case the $RS_2()$ function is aborted, all the armed LCSs are disarmed, and a new $RS_2()$ function is started on the new state of the grid. Finally, the third time-line shows a case where a low severity EI-failure happens followed by a high severity one: the former EI-failure moves the system in alert state and triggers the arming process at time 0, while the latter one moves the system in alarm state and triggers the load shedding process at time t.

The interaction between SWN and SAN takes place precisely on *the computation of the number of available LCSs at time t given the arming process started at time* 0*, and given an initial DoS severity level.* This number is a random variable whose value at time t is distributed according to $(prob(NumStation, t|InitDosLevel))$. This distributioon is computed in isolation on the SWN as the number of armed substations at a finite time horizon t and for an initial DoS severity level. If we consider a behaviour like that depicted in the first two time-lines of Fig. 5.3, and we consider that the DoS level at time 0 is L, then the SWN should compute the distributions at times ΔT given the initial DoS severity level L $(prob(*, \Delta T|L))$. Instead, if we consider the policy depicted in the last time-line, we should to compute the distributions at times $t + \Delta T$ given the initial DoS severity level L $(prob(*, t + \Delta T|L))$. This is due to the fact that the arming process is trigger by the first EI-failure, while the load shedding process by the second EI-failure happened at time t. To compute this distribution the SWN model can be significatly simplified (as depicted in Fig.5.4, since all aspects concerning the Electrical behaviour (issue of arming and load shedding commands, electrical failure, etc.) are already taken into account by the SAN model.

Practically, these distributions can be used in the SAN model in two different ways: to decide the execution success of a reconfiguration computed by RS2 considering at least k available LCSs or to evaluate the number of available LCSs on which the RS2 reconfiguration computed considering k available LCSs is applied. In the first case, RS2 reconfiguration is computed on k LCSs, then we use $prob(k, \Delta T|L)$ to decide if the reconfiguration induced by RS2 leads the EI in a stable state. In the second case, RS2 reconfiguration is still computed on k LCSs, but we use $prob(*, \Delta T|L)$ to compute the number of available LCSs at time t that will be really involved in the reconfiguration given the initial DoS severity level L.

Finally we have to highlight that $prob(*, \Delta T|L)$ can be also used to evaluate “a posteriori” the quality of the performance measures obtained on the SAN model where the $RS_2()$ functions are computed on the fixed subset of LCSs that are estimated being reachable at the time $RS_2()$.

Now we are going to introduce a set of experiments performed on the SAN model using $prob(*, \Delta T|*)$ to decide the execution success of a reconfiguration computed by RS2 considering at least k available LCSs. For this set of experiments, $prob(*, \Delta T|*)$ has been computed on a simplified model in which all the transitions that depend on the electrical state of the grid have been removed (Fig. 5.4). Moreover we have considered three levels of

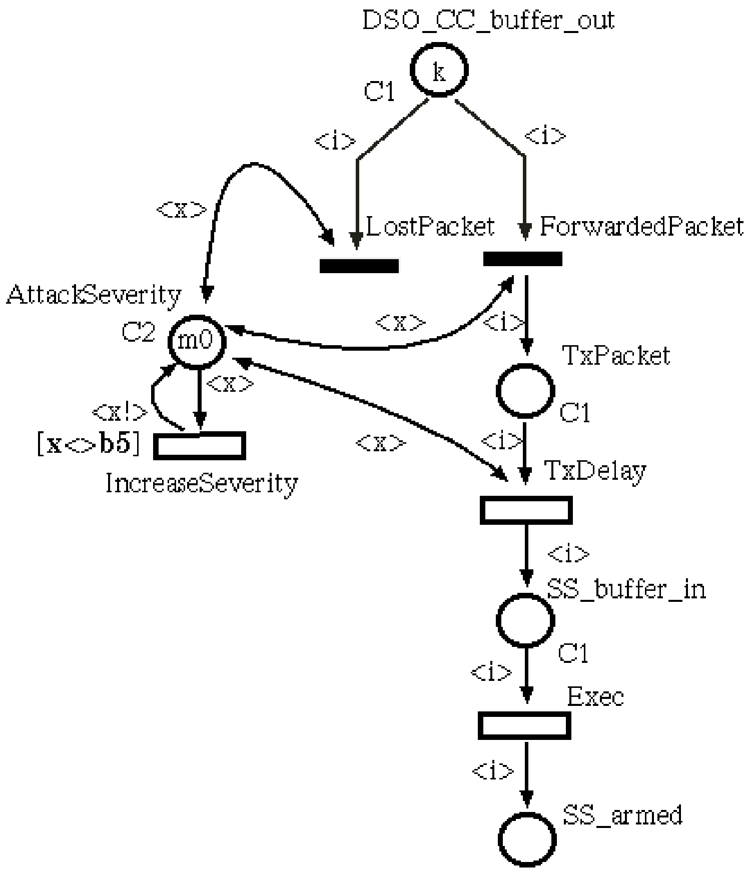


Figure 5.4: Simplified SWN model for the interaction with the SAN model.

Transition	**Type**	**rate/weight**	
LostPacket	Immediate	0.001	L1
		0.1	L2
		0.2	L3
ForwardPacket	Immediate	0.999	L1
		0.9	L2
		0.8	L3
TxDelay	Timed	2	L1
		0.2	L2
		0.02	L3
IncreaseSeverity	Timed	0,0083	
Exec	Timed	5	
ArmedExpired	Timed	0,00083	

Table 5.1: Rates and weights associated with the transitions of the model in Fig. 5.4

DoS severity and 27 LCSs. The rates and the weights associated with the model transitions are shown in Tab. 5.1. Observe that the weights and the rate associated with the transitions *LostPacket*, *ForwardPacket* and *TxDelay* depend on the DoS severity level.

The obtained $prob(*, \Delta T|*)$ are shown in Fig 5.5 where ΔT is 600.

The next (sub)sections explain the setting of the joint SAN and SWN experiments and the preliminary numerical results obtained.

5.3.1 Analyzed Power Grid, Measures of Interest and Failure Scenario

The analyzed electric power grid is depicted in Figure 5.6. The grid is the IEEE Reliability Test System published in 1979 (RTS-79[2]), typically used in other studies related to EPS. A dummy node (with the label "D" associated to the index) is added to represent the lines which are assumed to be on a common right of way or common tower for at least a part of their length. The label associated to the generators represents the initial (active) power and the maximum power that a generator can supply ("P_i/P_i^{max}"). The label associated to the loads represents the power demand of a load ("P_i"). The label associated to the lines represents the initial power flow through the line, the maximum power flow that each line can carry and the susceptance[3] ("F_{ij}/F_{ij}^{max} (b_{ij})"). In the initial grid setting all the ratios P_i/P_i^{max} are equal to a fixed value $\alpha = 0.95$, called the power grid stress level.

At time zero we suppose that one power lines is affected by a permanent disruption (e.g., due to a tree fall or a terrorist attack), thus becoming unavailable. The repair time of the failed power lines is fixed to 24 hours. The DoS attack ends after an exponentially distributed time with mean $MTTR^{CNET}$, and from that time RTS can start computing the $RS_2()$ reconfiguration action that will be applied after 10 minutes.

[2]http://www.ee.washington.edu/research/pstca/rts/pg_tcarts.htm

[3]The susceptance is used to determine the values for the power flow through the lines.

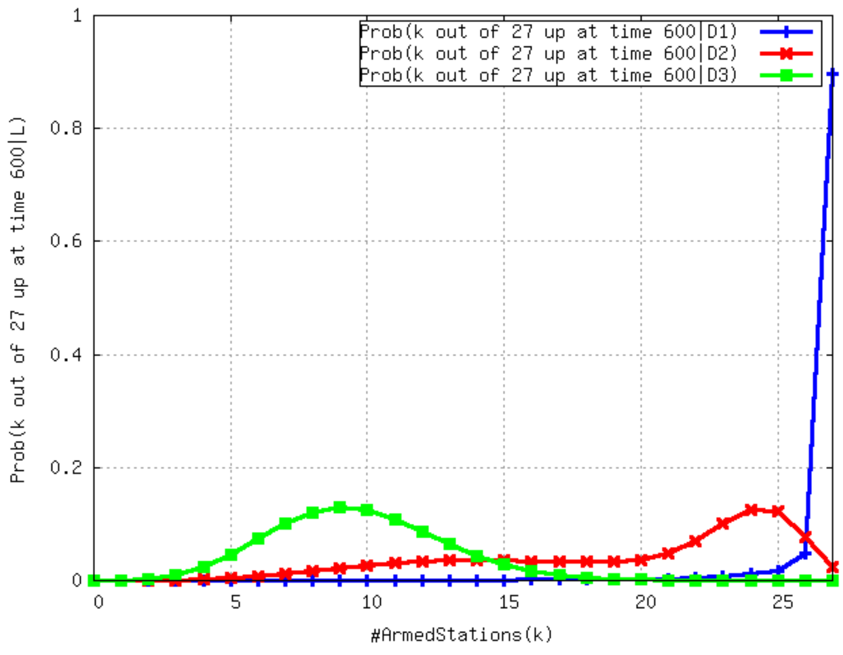


Figure 5.5: $prob(*, \Delta T|*)$ computed on the SWN model in Fig. 5.4

The measure of interest we consider is $P_{UD}(0,t)$, defined as the percentage of the mean power demand that is not met in the interval $[0,t]$ (the symbol 'UD' stands for 'Unsatisfied Demand'). It is a user-oriented measure of the blackout size and can be obtained as the load shed (i.e., the not served power due to a load shedding) divided by the power demand. In particular, we evaluate the impact on $P_{UD}(0,t)$ of the DoS failure of the communication network when a simultaneous failure of a transmission line occurred, and we make a sensitivity analysis varying the power line that fails at time zero, with $t = 4$ days.

5.3.2 Numerical Evaluations and Analysis of the Results

We now present some of the results that we obtained through the solution of the overall model previously sketched. A transient analysis has been performed, using the simulator provided by the Möbius tool [13]. For each study we executed a minimum of 2000 simulation runs (batches), and we set the relative confidence interval to 0.1 and the confidence level to 0.95. This means that the stopping criteria will not be satisfied until the confidence interval is within 10% of the mean estimate in 95% of the times.

In Figure 5.7 we show $P_{UD}(0,4)$ for different failed power lines, at varying of the number of LCS that become unavailable due to the DoS attack, for $MTTR^{CNET} = 24$ hours. The first case considers that all LCS are unavailable, and in this case we have 6 "critical" power lines, i.e., those lines whose failing induce an unsatisfied power demand. The second case considers a random number of unavailable LCS, following the distribution computed by the SWN model assuming that when the electrical failure happens, the DoS has

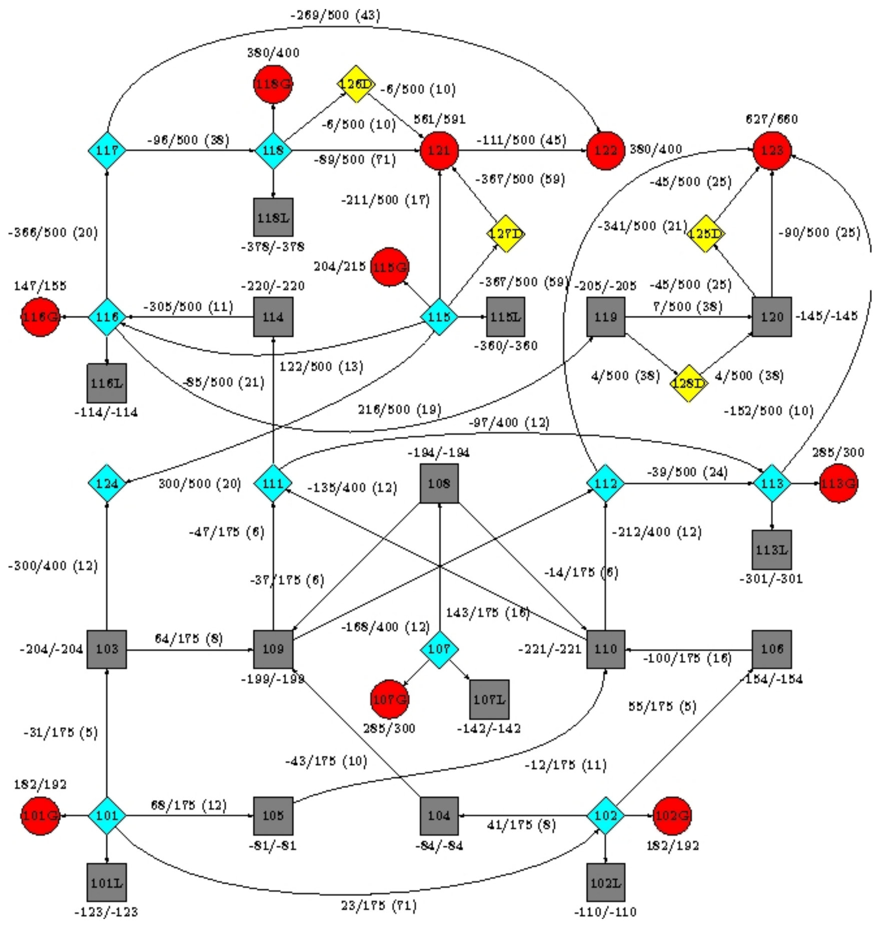


Figure 5.6: Diagram of the EI grid (generators are circles, loads are squares and substations are rhombi). For the sake of clarity, only the integer part of the original values associated to generators, power lines and loads are shown (in MegaWatt).

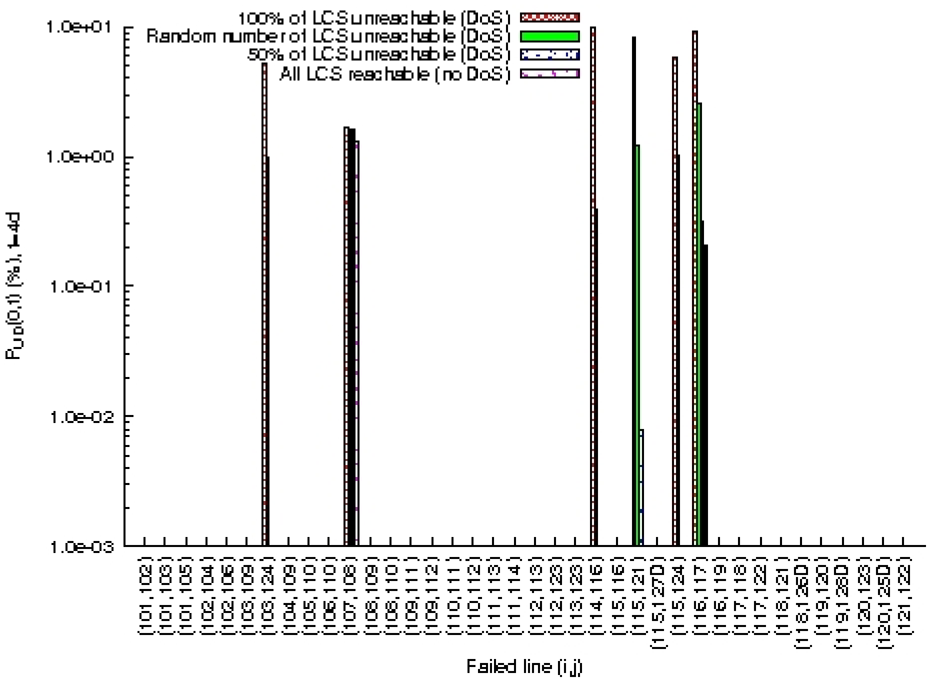


Figure 5.7: Percentage of the mean power demand that is not met in the interval $[0, 4]$ days, for different failed power lines, at varying of the number of LCS that become unavailable due to the DoS attack

already reached its maximum severity. In this case, the same 6 power lines are identified as critical, but the impact of their failures on $P_{UD}(0, 4)$ is less significant than in the previous case. The third case considers that only 50% of LCS are unavailable, and in this case there are 3 critical power lines only. In addition we can note that, moving from 100% of unavailable LCS to 50%, there is a significant reduction of $P_{UD}(0, 4)$: for example, $P_{UD}(0, 4)$ for line (15,21) becomes 3 orders of magnitude smaller moving from 100% of unavailable LCS to 50%. Finally, the last case represents the best case where no DoS is occurred and all the LCS are always connected to the network. In this best case only two power line are "critical".

5.4 Discussion

In this chapter we have shown how SWN models can be used to model the ICT nehaviour of an EPS. The SWN can very well represent the telecontrol, and the protocol that TSO and DSO use for reacting to electrical failures, characterized by the two phases of arming and load shedding. The SWN model also clearly points out (through its list of input parameters) what are the data of the electrical grid that need to be considered for a correct evaluation of the scenario. Since some of these parameters depend one on the other, evaluating the model on a set of varying parameters (with the goal of performing a sensitivity analysis) may lead to a very limited insight in the EPS behaviour. Examples of input parameters dependencies that are not easy to model significantly without including in the model the electrical state, are: the time between an electrical failure and the subsequence load shedding command, and the fact that the load shedding command is successful or not in limiting the effect of the electrical failure, the consequences of a load shedding on a subset of the target substations (due to an ICT malfunctioning). To realistically model both the ICT behaviour required by scenario 2 and the consequent electrical behaviour we have built an interaction between the SWN and the SAN model of EPS. The interaction has required a certain amount of work to be fixed, since we had to map the behaviour of the whole scenario onto the reconfiguration functions used by the SAN model to change the electrical state. We have chosen to limit to a minimum the number of things to be changed in the SAN model (that is already quite complex), and we have prefeerred to change the SWN to correctly fit with the SAN, while still maintaing an adequate level of representation of the ICT behaviour in scenario 2.

6 Conclusions

In this deliverable we have reported on three different validation and evaluation activities: definition of models for the attacks based on collected time series of attacks, evaluation and correctness validation of the CIS (with and without self healing capabilities) and study of interdependences between II and EI.

The work on the statistical characterization of the attacks complements in a very natural manner the work on the honeypot data collection described in D20, and makes those results available in a synthetic and polished form that can be easily included in other models.

The work on the CIS validation and evaluation has seen a strong interaction between the architectural partners that have defined the CIS and the modelling partners: we believe that this was a good example of the use of model to support the definition and tuning of an architectural solution.

Finally, the work on the interdependencies has attacked the main topic of this project: how do the Electrical Infrastructure and the Information infrastructure depends one on the other? The SAN models allow to build, in a parametric manner, a model of an electrical grid, and to consider a number of failure situations, both electrical and ICT. The interaction of SAN and SWN has then shown how the SAN model can be enriched with a detailed model of the ICT control, and in particular of the control defined in scenario 2, to provide even more realistic results of the behaviour of the power grid when the Information Infrastructure is under attack.

Bibliography

[1] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization based access control. In *4th IEEE Int. Workshop on Policies for Distributed Systems and Networks*, 2003.

[2] E. Alata. *Observation, characterization and modelling of Internet attack processes.* PhD thesis, Institut National des Sciences Appliques de Toulouse, LAAS-CNRS, Toulouse, December 2007.

[3] M. Anghel, K. A. Werley, and A. E. Motter. Stochastic model for power grid dynamics. In *40th Hawaii Int. Conference on System Sciences (CD-ROM)*, pages 113–122, Waikoloa, Big Island, Hawaii, January 2007. IEEE.

[4] S. Bernardi, S. Donatelli, and A. Horváth. Implementing compositionality for stochastic Petri nets. *International Journal of Software Tools and Technology Transfer*, 3(4):417–430, September 2001.

[5] A. Bessani, H.P. Reiser, P. Sousa, I. Gashi, V. Stankovic, T. Distler, R. Kapitza, A. Daidone, and R. Obelheiro. FOREVER: Fault/intrusiOn REmoVal through Evolution & Recovery. In *Middleware'08 Companion (Dec 1-5, 2008, Leuven, Belgium)*, 2008.

[6] A. Bessani, P. Sousa, M. Correia, N.F. Neves, and P. Veríssimo. Intrusion-tolerant protection for critical infrastructures. DI/FCUL TR 07–8, Department of Informatics, University of Lisbon, April 2007.

[7] A.N Bessani, P. Sousa, Miguel Correia, Nuno F. Neves, and P. Verissimo. Cheap intrusion-tolerant protection for crutial things. Technical report, LASIGE, Faculdade de Ciências da Universidade de Lisboa-Portugal, 2007.

[8] A. Bondavalli, I. Mura, S. Chiaradonna, R. Filippini, S. Poli, and F. Sandrini. DEEM: a tool for the dependability modeling and evaluation of multiple phased systems. In *DSN-2000 IEEE Int. Conference on Dependable Systems and Networks (FTCS-30 and DCCA-8)*, pages 231–236, 2000.

[9] J. Chen, J. S. Thorp, and I. Dobson. Cascading dynamics and mitigation assessment in power system disturbances via a hidden failure model. *Electrical Power and Energy Systems*, 27(4):318–326, 2005.

[10] S. Chiaradonna, P. Lollini, and F. Di Giandomenico. On a modeling framework for the analysis of interdependencies in electric power systems. In *IEEE/IFIP 37th Int. Conference on Dependable Systems and Networks (DSN 2007)*, pages 185–195, Edinburgh, UK, June 2007.

[11] G. Chiola, G. Franceshinis, R. Gaeta, and M. Ribaudo. GreatSPN1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation, North Holland Journal*, 24, 1997.

[12] A. Daidone, S. Chiaradonna, A. Bondavalli, and P. Veríssimo. Analysis of a redundant architecture for critical infrastructure protection. In R. De Lemos, F. Di Giandomenico, C. Gacek, H. Muccini, and M. Vieira, editors, *Architecting Dependable Systems V*, volume 5135 of *LNCS*, pages 78–100. Springer, Heidelberg, 2008.

[13] D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster, and W. H. Sanders. Möbius: An extensible tool for performance and dependability modeling. In B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, editors, *11th International Conference, TOOLS 2000*, volume 1786 of *LNCS*, pages 332–336. Springer Verlag, 2000.

[14] X. Defago, A. Schiper, and P. Urban. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Survey*, 36(4):372–421.

[15] I. Dobson, B. A. Carreras, V. Lynch, and D. E. Newman. An initial model for complex dynamics in electric power system blackouts. In *34th Hawaii Int. Conference on System Sciences (CD-ROM)*, Maui, Hawaii, January 2001. IEEE.

[16] S. Donatelli, E. Alata, J. Antunes, M. Kaniche, V. Nicomette, N.F. Neves, and P. Verissimo. Experimental Validation of Architectural Solutions. Technical Report Deliverable D20, Critial Utility InfrastructuAL Resilience, Project co-funded by the European Commission within the Sixth Framework Programme, 2009.

[17] G. Dondossola, F. Garrone, G. Deconinck, Beitollahi H., and T. Rigole. Testbed deployment of representative control algorithms. Workpackage 3 Deliverable D9, CRUTIAL consortium, Jan. 2008. 2008.

[18] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-formed Coloured nets for symmetric modelling applications. *IEEE Transactions on Computers*, 42:(11): 1343 – 1360, 1993.

[19] Mohamed Kaâniche, Eric Alata, Vin cent Nicomette, Yves Deswarte, and Marc Dacier. Empirical analysis and statistical modeling of attack processes bas ed on honeypots. In *Workshop on empirical evaluation of dependability and security (in conjunction with the international conference on dependable systems and netw orks,* DSN *'06) (*WEEDS *'06)*, Philadelphia, USA, 2006.

[20] M. Kaniche, M. Beccuti, C. Brasca, S. Chiaradonna, S. Donatelli, F. Di Giandomenico, G. Franceschinis, K. Kanoun, J.-C. Laprie, P. Lollini, and F. Romani. Preli;inary Modelling Framework. Technical Report Deliverable D16, Critial Utility InfrastructuAL Resilience, Project co-funded by the European Commission within the Sixth Framework Programme, 2009.

[21] M. Moretto. Progettazione, realizzazione ed utilizzo di un generatore di simulatori per sistemi a fasi multiple. Master's thesis, Università degli Studi di Pisa, 2004.

[22] I. Mura and A. Bondavalli. Markov regenerative stochastic Petri nets to model and evaluate the dependability of phased missions. *IEEE Transactions on Computers*, 50(12):1337–1351, 2001.

[23] N.F. Neves, P. Veríssimo, A. Abou El Kalam, A. Baina, H. Beitollah, A. Bessani, A. Bondavalli, M. Correia, A. Daidone, G. Deconinck, Y. Deswarte, F. Garrone, F. Grandoni, H. Moniz, T. Rigole, and P. Sousa. Preliminary specification of services and protocols. *Critical Utility InfrastructurAL Resilience. Project co-funded by the European Commission within the Sixth Framework Programme. Deliverable D10*, 2008.

[24] V. Nicomette, M. Kaniche, and E. Alata. Attackers Behavior: Experiment and Results. Technical Report Report, LAAS-CNRS, 2009.

[25] Wolfgang Reisig. Petri Net Models of Distributed Algorithms. In *Computer Science Today*, pages 441–454. 1995. Also published as Informatik-Bericht Nr. 48, Humboldt-Universitt zu Berlin, August 1995.

[26] F. Romani, S. Chiaradonna, F. Di Giandomenico, and L. Simoncini. Simulation models and implementation of a simulator for the performability analysis of electric power systems considering interdependencies. In *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*, pages 305–312, 2007.

[27] W.H. Sanders and J.F. Meyer. A unified approach for specifying measures of performance, dependability and performability. In A. Avizienis and J. Laprie, editors, *Dependable Computing for Critical Applications, Vol. 4 of Dependable Computing and Fault-Tolerant Systems*, pages 215–237. Springer Verlag, 1991.

[28] P. Sousa, A. Bessani, M. Correia, N. Neves, and P. Veríssimo. Resilient intrusion tolerance through proactive and reactive recovery. In *13th IEEE Pacific Rim Dependable Computing conference*, 2007.

[29] P. Verissimo and N. Neves (editor). Preliminary specification of services and protocols. Deliverable D10, EC project IST-2004-27513 (CRUTIAL), January 2008.