Università degli Studi di Salerno

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione
Ciclo 33 – a.a 2019/2020

TESI DI DOTTORATO / PH.D. THESIS

Statistical Techniques for Mitigation and Prevention of Distributed Attacks over Communication Networks

MARCO TAMBASCO

SUPERVISOR: **PROF. VINCENZO MATTA**

PHD PROGRAM DIRECTOR: **PROF. PASQUALE CHIACCHIO**

Dipartimento di Ingegneria dell'Informazione ed Elettrica
e Matematica Applicata
Dipartimento di Informatica

Contents

Acronyms and Abbreviations

APT Advanced Persistent Threat.

BIC Botnet Identification Condition.

cIMS containerized IMS.

CNF Containerized Network Function.

CNR Containerized Network Replica.

CNT Container.

CSCF Call Session Control Function.

CTMC Continuous-Time Markov Chain.

DCK Docker daemon.

DDoS Distributed Denial of Service.

DNS Domain Name System.

EDR Emulation Dictionary Rate.

GAN Generative Adversarial Network.

HPV Hypervisor.

HSS Home Subscriber Server.

HTTP HyperText Transfer Protocol.

HW Hardware components.

I-CSCF Interrogating CSCF.

IaaS Infrastructure as a Service.

IDS Intrusion Detection System.

IMS IP Multimedia Subsystem.

IoT Internet of Things.

MANO Management and Orchestration.

MIR Message Innovation Rate.

MRM Markov Reward Model.

MTTF Mean Time to Failure.

MTTR Mean Time to Repair.

NFV Network Function Virtualization.

P-CSCF Proxy CSCF.

RBD Reliability Block Diagram.

S-CSCF Serving CSCF.

SDN Software Defined Networking.

SFC Service Function Chain.

SIP Session Initiation Protocol.

SMTP Simple Mail Transfer Protocol.

SRN	Stochastic Reward Networks.
VM	Virtual Machine.
VNF	Virtual Network Function.

Chapter 1

Introduction

1.1 Security Issues over Networks

Nowadays, dealing with security aspects of modern telecommunication networks is extremely challenging. On one hand, novel paradigms such as the 5G, Internet of Things (IoT), Network Function Virtualization (NFV), Software Defined Networking (SDN), just to mention a few, represent a great opportunity to have an extremely interconnected world. On the other hand, the common all-IP structure inherited from the aforementioned paradigms makes them exposed to a huge number of network threats. For instance, a 5G network can be jeopardized by a man-in-the-middle attack; the sensors belonging to an IoT system might be compromised so as to become vectors of a fast-propagating threat which follows an avalanche effect; the weak nodes of an NFV/SDN architecture (i.e. the network interfaces) can be attacked to gain the access to critical elements such as the operating system or the hypervisor. It is interesting to analyze some official data provided by the Clusit [1], the credited Italian association for information security. For instance, Fig. 1.1 reports the severity (graded according to 3 levels - critical, high, medium) for 10 representative network targets monitored in 2019. It is possible to notice that the governmental and critical infrastructures were subject to most attacks with "critical" severity, followed by banking/finance and other relevant applications. Figure 1.2 shows the severity for some specific network threats monitored in 2019. Here it is possible to see that techniques such as Advanced Persistent Threat (APT), 0-day attacks and Distributed Denial of Service (DDoS) attacks are the most dangerous in terms of severity. In the next section we focus on the last category (DDoS) which, in particularly in its L7 variant (where L7 refers to the application layer of the TCP/IP protocol stack), represents one of the most alarming threats nowadays affecting communication networks.

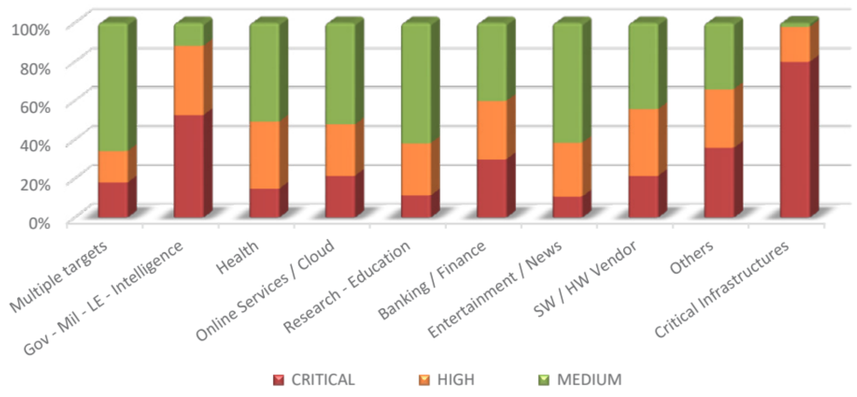


Figure 1.1: Severity distribution for 10 representative targets in 2019 (Source: 2020 Clusit Security Report).

1.2 Distributed Denial of Service Attacks

The purpose of a classical Denial of Service (DoS) attack is to make a service unavailable. This cyber-attack is typically performed by overwhelming the target with spurious requests, overload it and block the use of the service requested by legitimate users. One dangerous evolution of a DoS attack is a *Distributed* DoS (DDoS) attack, which takes place when the target is flooded by many

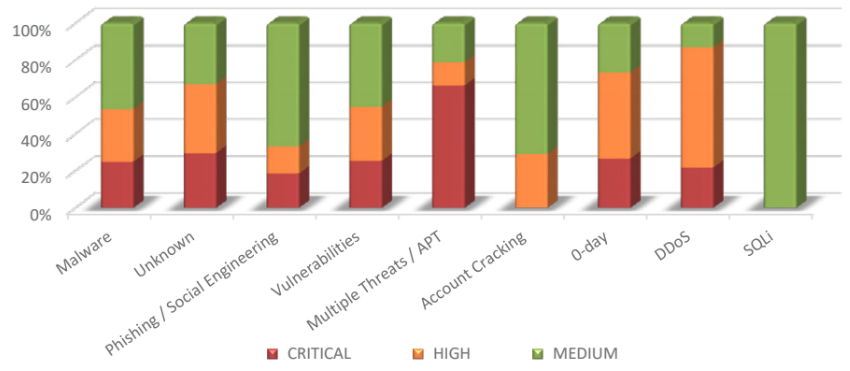


Figure 1.2: Severity distribution for attack type in 2019 (Source: 2020 Clusit Security Report).

sources with relatively small (thus, non-suspicious) individual requests rates. In the DoS scenario, an Intrusion Detection System (IDS) can easily detect the attack since the attacker generates a great volume of traffic from a single source. In comparison, in a DDoS attack the small rate of fake requests from different sources makes practically impossible to detect them using indicators of the network traffic volume. The sources of the attack are represented by many compromised hosts (the *bots*) forming a *botnet* coordinated by a *botmaster*. The botmaster instructs the bots to send specific attack packets to the victims. Botnet launching DDoS attacks can be very dangerous and can make rapidly unavailable significant portions of the network. A DDoS attack can be bought on some darknets in the form of a service (DDoS as a Service) for as little as $10 for one hour or $60 for one day. Due to its flexibility, a DDoS attack can be designed to damage network resources in a number of application domains as shown in Fig. 1.3. Such a figure shows that the DDoS phenomenon appears to be “no holds barred”. The most targeted sectors are gaming, finance/insurance, services and entertainment.

Typically, DDoS attacks can be implemented at various layers of the TCP/IP stack. Some examples are briefly described below:

- **SYN flood attack**: attackers send a huge number of SYN requests to a victim starting the three-way-handshake procedure of the TCP protocol without completing it. The consumed resources of the half-opened connections at the victim's side can make the system unavailable to legitimate users.

- **Teardrop attack**: attackers send to a target machine IP packets assembled with a wrong fragmentation offset. The reassembling phase can cause a crash of the target machine.

- **Smurf attack**: attackers send ICMP messages to the broadcast address of a network using the victim's IP address (IP *spoofing*). All hosts of the network receive such ICMP messages and in turn answer to the source IP address (victim), producing a message amplification that makes the victim overwhelmed by these responses.

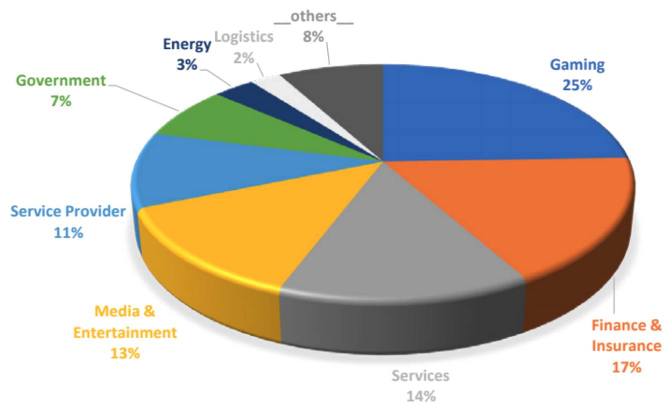


Figure 1.3: Targets of DDoS attack - 2019 (Source: 2020 Clusit Security Report).

The aforementioned DDoS attacks have in common the possibility of acting on specific fields of the TCP/IP stack, mainly positioned at the network (e.g. Teardrop and Smurf attacks) or transport (e.g. SYN flood) layers. Nowadays, revealing and/or stemming such attacks is no longer a hard task. In fact, an intrusion detection system can be easily instructed to selectively inspect the values of critical flags within the network and transport layers of the TCP/IP structure, and to raise an alarm in case of suspicious combinations of such flags. In contrast, the new class of application-layer (L7-DDoS) attacks is arising recently as one of the most powerful threats. In principle, the L7 qualification refers to all those protocols designed to run at the application layer of the TCP/IP stack: HTTP, DNS, SIP, SMTP, and many others. In practice, an L7-DDoS attack becomes very dangerous when implemented through the HTTP protocol that can be maliciously exploited to compromise a web resource through artificially crafted GET and POST requests, giving rise to a broad variety of admissible messages to emulate legitimate traffic patterns. Accordingly, in this thesis we will broadly use the term L7-DDoS to denote an application-layer attack exhibiting great richness of admissible messages used to emulate legitimate traffic, such as HTTP-based DDoS. These types of attacks are expected to be particularly relevant to next-generation networks. This is because such networks will exploit widely the HTTP protocol, owing to the convenient possibility of encapsulating a great variety of messages through a dedicated REST API interface. For instance, the HTTP protocol coexists with the SIP protocol to handle the nodes interconnections in many modern implementations of the IP Multimedia Subsystem (see Clearwater [2]), an infrastructure having a critical role in managing the multimedia content across 4G/5G core networks. Moreover, HTTP acts as a protocol *vector* for the Management and Orchestration (MANO) framework, which governs the lifecycle of virtualized services in accordance to the NFV paradigm. In fact, MANO implementations (e.g. OSM, Open Baton, ONAP, etc.) expose a broad set of HTTP REST APIs [3] useful to interact both with internal modules and with the external web resources.

The detection of L7-DDoS attacks is demanding for a series of reasons, for example [4]: *i)* the bandwidth is lower in comparison to DDoS attacks perpetrated at network and transport layer; *ii)* GET and POST requests can be forged using fake contents with

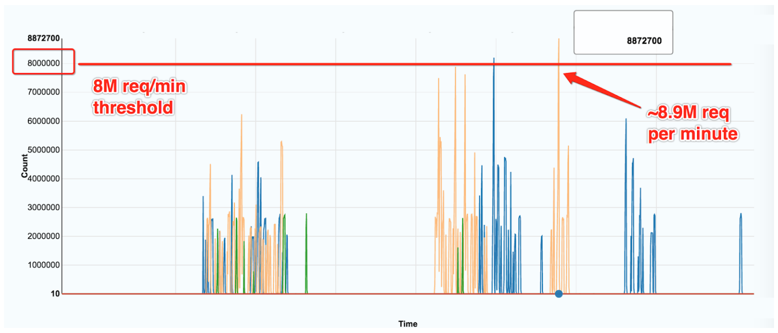


Figure 1.4: An example of L7-DDoS attack against Logitravel, a famous worldwide travel agency (Source: Verizon provider).

potentially infinite possibilities; *iii)* during a L7-DDoS, the TCP and UDP protocols are actually used in a legitimate way. There exist specific real-world examples showing the relevance and power of L7-DDoS attacks. One is the case of *Logitravel*, a worldwide travel agency interested by an L7-DDoS attack with peaks reaching approximately 9 million requests per minute (corresponding to about 150000 requests per second). The analysis of such attack has been carried out by Verizon [5] and is reported in Fig. 1.4. To put this in perspective, the Logitravel agency has daily peaks of about 3300 requests per second across the infrastructure hosted on Verizon Edgecast CDNs. Accordingly, the considered attack has produced a spike 50 times larger than the normal load on the original infrastructure. The sources of the attack are primarily originated from Europe (in particular from France, Spain and Italy). Moreover, further investigation revealed that the bunch of L7 requests were HTTP POSTs to Logitravel's quoting and pricing API endpoints, and then translated into an exceptional amount of back-end database queries as shown in Fig. 1.5.

From a topological viewpoint, DDoS attacks might rely on botnets that exhibit different architectures. Some exemplary architectural scenarios are shown in Fig. 1.6 where we can consider the following taxonomy:

- **Centralized Architecture**: this is probably the most

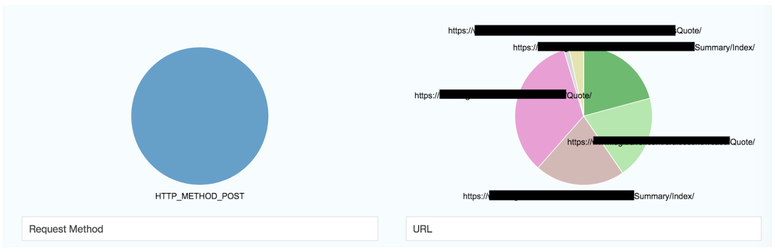


Figure 1.5: Back-end database queries relative to the L7-DDoS attack shown in Fig. 1.4 (Source: Verizon provider).

common architecture. The botmaster instructs the botnet through the so-called Command & Control (C&C) centers, which are often exploited as load balancers.

- **P-2-P Architecture**: the bots share some information about the target. Such an architecture exhibits more resiliency since the problem of a single point of failure (present in the centralized solution) is avoided.

- **Hybrid Architecture**: this architecture represents a kind of mix between centralized and P2P scenarios. Two types of bot agents are present: server and client bots. The former have management roles, whereas the latter perform the attack.

From a technological point of view, the device in charge of detecting and mitigating DDoS attacks is a Network IDS. Such devices (implemented either by hardware and software) are strategically located in "hot" points of networks (e.g. close to routers, border gateways, sensible network targets) to collect and analyze the data flows. The technical literature suggests two kind of approaches: *misuse* and *anomaly* detection. The former relies on a set of rules (similar to the rules implemented by anti-viruses) allowing to reveal suspicious behavior within a traffic flow (e.g. a weird combination of TCP flags possibly resulting in a SYN flood attack). The latter is based on the idea that the deviation from a regular activity might be a symptom of a network attack. Un-

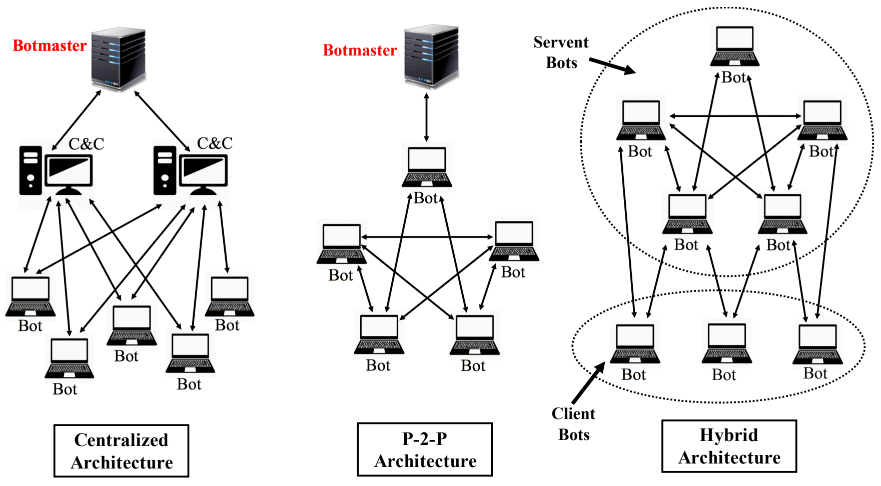


Figure 1.6: Botnet architectures and models.

fortunately, both misuse and anomaly detection approaches could result ineffective to deal with modern L7-DDoS attacks. Some of these attacks, in fact, exhibit a very subtle behavior consisting in emulating almost perfectly the web activity of legitimate users. Devising reliable strategies for botnet identification under L7-DDoS attacks is one of the main contributions of this thesis, which will be discussed in great detail in Chapters 2 and 3.

1.3 Network High Availability and Performability

A very interesting peculiarity of a DDoS attack is given by its duration. As can be seen in Fig. 1.7, over 95% of DDoS attacks lasted less than 3 hours, and just a tiny fraction (0.24%) goes beyond 24 hours. Such information suggests that, in many cases, it is impossible to quickly react to such incredibly fast attacks. In this spirit, an effective botnet detection and identification strategy can be coupled with the design of a robust and resilient network. This latter aspect can be addressed through the high availability and performability concepts. High availability (HA) means that

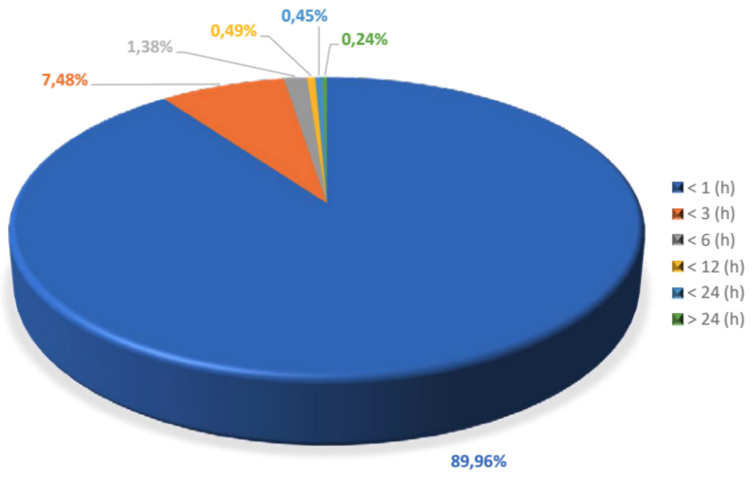


Figure 1.7: Duration of DDoS attacks - 2019 (Source: 2020 Clusit Security Report)

the maximum tolerated downtime of a system (i.e. a network infrastructure) amounts to 5 minutes and about 15 seconds per year. Another way to express HA is that the probability that the system is working when requested for use amounts to 0.99999 (the so-called "five nines"). Five nines is an industry benchmark historically adopted to characterize telecommunication systems (from traditional telephony to IP-based networks) in terms of their ability of providing a service. It also holds within modern networks, where the impelling softwarization process imposes new challenges. Virtualized and Containerized environments arising with the Network Function Virtualization (NFV) paradigm, in fact, drastically modify the structure of network nodes (e.g. router, switches, web servers, firewalls, and many others) whose software logic (often known as Virtual Network Function - VNF) is decoupled from the hardware resources through intermediate entities such as hypervisors or container engines. On the other hand, performability (performance + availability) takes into account also performance metrics of a system, namely its ability to provide a service according to a specific *demand* level. For instance, in case of a system such as IP Multimedia Subsystem (IMS), the performance is measured in terms of the maximum number of concurrent multimedia sessions that the system is able to support. Generally speaking, to observe an improvement of availability and performability, the design of specific redundancy strategies is a fundamental step. In fact, having more redundant network elements (or "replicas") such as hardware, hypervisor, software, guarantees more resiliency to faults. Obviously, one must account for the costs associated to redundancy, putting in field strategies that maximize the trade-off between availability/performability and costs. This critical issue will be addressed in Chapter 4.

1.4 Main Contributions and Overview

In this thesis it is possible to recognize two main contributions. The first one pertains to distributed network attacks, and in particular to the randomized DDoS attacks originally introduced in [6], for which we provide some useful advances. Precisely, we: *i*) formalize a novel challenging scenario that accounts for botnets spread over different clusters where the emulation dictionaries may be partially shared (overlap); *ii*) devise a consistent algorithm to correctly identifying the bot clusters; when it is not possible, the algorithm allows to discriminate bots from legitimate users anyway; *iii*) carry out an extensive experimental campaign over real datasets, which allows us to examine a broader range of users' behavior as compared to previous results [6].

The second contribution pertains to the design of architectural countermeasures against network attacks. In particular, we: *i*) exploit the Stochastic Reward Networks (SRN) formalism to characterize a "virtualized" network in terms of failure and repair events of its composing elements (hardware, virtualization layer, software); in order to go beyond the classic Markov models, we consider also some examples encompassing more realistic semi-Markovian models; *ii*) devise optimized network resiliency strategies where performability and costs are jointly considered;

iii) design a software tool adhering to the model-driven approach for automating the network resiliency life cycle [7–15].

As a result, we provide guidelines to implement optimal resilience strategies in the context of distributed network paradigms such as NFV, Cloud Computing, SDN. On one hand, it is worth noticing that the methodologies and strategies proposed in this thesis to devise architectural countermeasures are not unique to distributed network attacks, and can be fruitfully used in many other applications. On the other hand, the interplay between "distributed" and "virtualized" becomes particularly relevant for our purposes, since virtualization technologies introduce specific challenges and opportunities in distributed environments. For example, in a distributed system, multiple virtual machines can be deployed to add redundancy in face of attacks, and the availability target of such ensemble of machines can be flexibly achieved by increasing the virtual resources with a limited impact on hardware costs.

The thesis is organized as follows. Chapter 2 introduces some useful background that is relevant to the analysis conducted in the subsequent sections. In Chapter 3 we present the formal model of L7-DDoS attacks with multiple emulation dictionaries and the relative novel results. Chapter 4 delves into a detailed analysis of availability and performability models with focus on a virtualized version of IMS. Chapter 5 contains some concluding remarks and hints for future research.

Chapter 2

Background

In this section, we introduce some useful background concepts that will be exploited throughout this thesis. In particular, we focus on two crucial aspects. The first one concerns the Randomized L7-DDoS model, originally conceived in [6], which represents the starting point for the characterization of the DDoS attacks addressed in this work, where the more challenging setting of multiple overlapped dictionaries is considered. The second aspect pertains to the description of some formalisms useful to characterize the availability of modern network infrastructures, with a particular focus on the Stochastic Reward Networks (SRN) technique.

2.1 Related Work on DDoS attacks

Networks are consistently exposed to a variety of dangerous cyber-threats. Providing reliable processing and inference strategies to contrast these threats is a critical part of the network security chain [16–22].

In recent years, several useful works addressed the problem of devising countermeasures toward DDoS identification and mitigation — we invite the Reader to consult [4] and [23] for an overview on the subject. A popular approach to tackle DDoS attacks and bot identification is based on statistical methods aimed at revealing anomalous patterns (e.g., repetition patterns, unusual traffic rates). The DDoS identification strategy devised in [24] relies on the detection of anomalies observed in the entropy and in the (frequency-sorted) distribution of selected packet attributes. In a similar vein, a technique based on entropy detection is proposed in [25], where the degree of randomness associated to suitable packet attributes is elected as the critical marker of a DDoS. A hierarchical method aimed at capturing shifts within spatio-temporal traffic patterns is designed in [26]. Such shifts are then exploited to reveal a DDoS flooding attack. The framework *Umbrella* is proposed in [27], focusing on amplification DDoS attacks and offering to the victims the possibility to customize the level of defense. In [28], the generalized entropy and the information distance metrics are used to reveal low-rate DDoS attacks, where dissimilarity between legitimate and malicious traffic is evaluated. The case of low-rate *shrew* DDoS attacks (TCP flows constrained to sustain a small fraction of their ideal rate at low attack costs) is tackled in [29], where a model amenable to control the TCP congestion window at the target's side is exploited. In [30], the Authors propose a fingerprinting approach relying on the analysis of packet-level features in a traffic flow (e.g., length, number of transmissions). The obtained fingerprints are used to generate "normal" profiles, and then a clustering-based detection algorithm decides about the legitimacy of an *unknown* fingerprint possibly associated to a DDoS source.

Since DDoS attacks are typically based on the repetition of predefined actions or patterns, their power can be greatly amplified when they are implemented at the application layer of the TCP/IP protocol stack (L7-DDoS) [31–35]. In this case, the botnet has at its disposal a numerically consistent set of messages (e.g., HTTP GET requests), which helps the bots avoid massive repetitions that could appear suspicious. One notable real-world L7-DDoS attack was recently performed through a sophisticated malware named Mirai [36]. This malware compromised a network of cameras (hence, the bots), which started forming variously interacting clusters to launch a distributed attack to a web server.

Even if the application layer offers to the attacker the possibility of assembling fairly rich request patterns, we should keep in mind that requests must be performed in some structured way in order to impair an IDS. For example, when attacking a certain website, only some requests are admissible/meaningful at a certain time. This concept can be conveniently abstracted by saying that the bots can access an *emulation* dictionary (the set of admissible messages) in order to emulate the behavior of a normal (i.e., legitimate) network user. The emulation dictionary can be learned continually (and, hence, enriched continually) by the botmaster through monitoring and collection of legitimate network traffic patterns. Recently, this class of DDoS attacks with emulation dictionary has been described in some detail in [6, 37, 38].

One useful framework for implementing and counteracting network attacks is the framework of Generative Adversarial Networks (GANs), originally conceived by Goodfellow [39] and typically applied into the realm of fake image generation. GANs are based on the idea that the attacker (or generator) and the defender (or discriminator) can establish their strategies through two neural networks which are trained together. Based on training sets of real samples, the generator attempts to learn the best functions to disguise the generated (fake) samples as legitimate, while the discriminator attempts to provide the best possible response. In this way, the generator and the discriminator play within an adversarial setting, in the game theory sense. In the specific context of network attacks, there are some recent works that apply the GAN framework to the problem of characterizing/revealing flooding attacks, where the competing neural networks (also used in some deep-learning variant) operate on pre-labeled datasets of network traffic (e.g., KDD99, NSL KDD, CIC IDS) [40–43]. To the best of our knowledge, an application of the GAN framework to the L7-DDoS scenario (which is relevant to this thesis) is currently missing. In such scenario, the GAN framework introduces new possibilities and challenges, as succinctly detailed below, where a GAN-based approach is specifically contrasted to the approach pursued in this work.

- At the attacker's side, the DDoS mechanism considered in this work emulates normal traffic patterns by picking uniformly at random a request from a set (the emulation dictionary) of admissible messages. In comparison, a neural network can emulate traffic patterns in a more sophisticated way, e.g., by exploiting structures arising over time in the traffic patterns collected in the training set.

- At the defender's side, the solution provided by a GAN would be typically costly, hardly interpretable, and highly dependent on the training set. In comparison, our defense strategy provided by the botnet identification algorithm is simple, sharply interpretable and relies on statistical descriptors that apply to a broad ensemble of settings.

- In the GAN framework, training the neural networks will be costly, especially due to the peculiarities of our distributed network attacks, since the (raw) features feeding the neural networks are given in terms of a large number of time-series (i.e., the traffic patterns). Exploiting the structure hidden in these time-series to provide good emulation (and appropriate countermeasures) is expected to be a complex task involving optimization over many hyperparameters, necessity of large datasets and noteworthy computational resources.

2.1.1 Randomized L7-DDoS Model

The novel L7-DDoS model presented in [6] relies on the idea that an attack would be even more effective if an attacker has the ability of emulating normal network traffic. In other words, an attacker tries to mimic the behavior of "legitimate" users by using the same application layer (L7) messages. Such messages are gathered by the botmaster, and exploited by the malicious nodes of a botnet to impair the resources of a network target. Precisely, the bots try to bypass the network Intrusion Detection System (IDS) by implementing a "drop-by-drop" strategy, where *i*) each *individual* bot sends a relatively low number of legitimate requests to a target site; and *ii*) the global rate of requests produced by the entire botnet is so large to saturate the computational resources of the target.

The goal of the network administrator is to mitigate these attacks, and to this aim a critical task is to identify (and then ban) the bots. In order to accomplish botnet identification, the network administrator collects and examines the messages that form the traffic patterns produced by the different users. When appropriate, sets of messages will be generally referred to as "dictionaries", for example, the set of messages used by the bots to emulate normal traffic patterns will be called "emulation dictionary". Likewise, the set of messages representing the actual activity of a certain user or subnet will be called "empirical dictionary".

In order to formalize the randomized L7-DDoS model, some network indicators are introduced in [6]. The first one is the *empirical transmission rate* at time t, viz.

$$\hat{\lambda}_{\mathcal{S}}(t) \triangleq \frac{N_{\mathcal{S}}(t)}{t}, \tag{2.1}$$

where $N_{\mathcal{S}}(t)$ represents the number of messages transmitted by users of a subnet $\mathcal{S}$ up to time t. When a limiting rate is meaningfully defined (as t goes to infinity), it is denoted by $\lambda_{\mathcal{S}}$, which is simply referred to as the transmission rate of subnet $\mathcal{S}$.

Precisely, two kinds of transmission schedulings are considered in [6], namely, the synchronous (constant-rate), and the independent Poisson scheduling. In the synchronous case, all nodes transmit in a synchronous manner, and the interval between two transmissions has a constant duration amounting to $1/\lambda$. In the second case, the transmission of user u is modeled as a Poisson process with rate λ_u, and the processes of distinct users are mutually independent.

A second network indicator pertains to the *content* of the sent messages, and is called Message Innovation Rate (MIR). First, we define $\mathscr{D}_{\mathcal{S}}(t)$ as the empirical dictionary composed by the *distinct* messages sent, up to time t, by users within $\mathcal{S}$. Accordingly, the

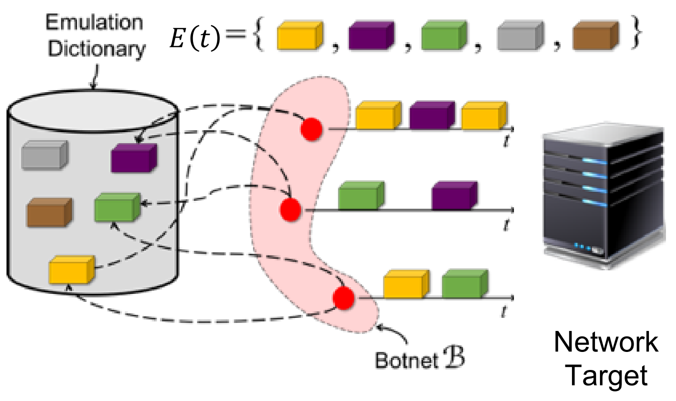


Figure 2.1: The emulation dictionary contains legitimate messages exploited by a botnet to impair a network target.

empirical Message Innovation Rate (MIR) is defined as follows:

$$\hat{\rho}_{\mathcal{S}}(t) \triangleq \frac{|\mathscr{D}_{\mathcal{S}}(t)|}{t}. \tag{2.2}$$

If $\hat{\rho}_{\mathcal{S}}(t) \xrightarrow{\text{p}} \rho_{\mathcal{S}}$, the limiting value $\rho_{\mathcal{S}}$ is simply referred to as the MIR of subnet $\mathcal{S}$. In a sense, the MIR quantifies the "innovation power" of a botnet. Intuitively, a high MIR implies that bots have a great ability of being innovative (namely, of being similar to legitimate users), thus, the network attack is most likely to be successful. In contrast, a low MIR implies that bots are pretty similar (namely, they have in common many messages perhaps due to a limited emulation dictionary), thus, a network defender might easily recognize such a similarity.

The last network indicator refers to the degree of innovation sustained by the bots when they assemble legitimate traffic patterns emulating the activity of normal users. Preliminarily, it is necessary to introduce the *emulation dictionary* $\mathscr{E}(t)$, which is formally defined as the set of legitimate messages available at time t to all botnet members — see Fig. 2.1. In order to mimic normal patterns, the bots pick uniformly at random the legitimate messages (represented as blocks in Fig. 2.1) from the emulation dictionary, which is learned continually to sustain a reasonable innovation rate. Then, the Emulation Dictionary Rate (EDR) is defined as:

$$\alpha \triangleq \lim_{t\to\infty} \frac{|\mathscr{E}(t)|}{t}. \tag{2.3}$$

One criticism that could be raised with reference to the uniform/independent picking mechanism adopted in randomized DDoS attacks is that, when surfing over a webpage, the sequence of HTTP requests is likely to exhibit a certain correlation between subsequent samples. In this connection, it is worth remarking that, as we will see in the forthcoming sections, the core of the identification strategy proposed in [6] relies on dependence/independence across network users (dependence across bots and independence across normal users) and *not* over time. For this reason, even changing the granularity at which messages are sent, the activities of normal users are expected to feature a smaller degree of dependence than the activities of bots. One possibility to take into account the aforementioned granularity at a webpage level, is that the attacker identifies as the atomic element of the emulation dictionary a *sequence* of requests. Then, the analysis presented in the thesis applies if we assume that the defender has some approximate knowledge of the rule adopted by the attacker to parse the requests. For example, if the attacker uses the granularity of the webpage, then the defender could use the same parsing when examining the captured traffic. Another possibility is to consider more sophisticated models (e.g., Markov models) for the emulation of traffic patterns embodying some temporal dependence.

2.1.2 The Botnet Identification Condition

As specified before, the MIR indicator provides a kind of "correlation degree" among users. For instance, if two users are bots, they are expected to exhibit a low MIR (low ability of being innovative). In contrast, if the two users are normal, they are expected to exhibit a high MIR (high independence). This behavior suggests that a meaningful way to declare whether a subnet is made by bots, is to compare its empirical MIR to the MIR that the same subnet would produce if it were made by bots. Therefore, it is critical to establish what is the MIR produced by a botnet. With reference to the case of a single emulation dictionary, the MIR of a botnet has been analytically characterized in [6].

Preliminarily, it is necessary to introduce the following function:

$$\mathscr{R}(\alpha,\lambda) \triangleq \frac{\alpha\,\lambda}{\alpha+\lambda}. \tag{2.4}$$

Theorem 1 (Botnet MIR)**.** *Let $\mathcal{B}_{\rm tot}$ be a botnet, where the node transmission policies can be either synchronous, or independent Poisson processes, with rates λ_u, for $u \in \mathcal{B}_{\rm tot}$. Let now $\mathcal{B}$ be a subset of the botnet $\mathcal{B}_{\rm tot}$. Then, the MIR of $\mathcal{B}$ is:*

$$\frac{|\mathscr{D}_{\mathcal{B}}(t)|}{t} \xrightarrow{\rm P} \rho_{\mathcal{B}} = \mathscr{R}(\alpha, \lambda_{\mathcal{B}}) \tag{2.5}$$

where $\lambda_{\mathcal{B}} = \sum_{u\in\mathcal{B}} \lambda_u$ is the aggregate transmission rate of $\mathcal{B}$, and where the convergence in (2.5) is meant to be in probability.

Once that the MIR of a botnet is available, it is necessary to understand to what extent it is distinguishable from the MIR of normal users. To this aim, we introduce a *Botnet Identification Condition* (BIC). For the sake of clarity, let us assume that the empirical EDRs pertaining to a couple of users are comparable, namely $\hat{\alpha}_1 \approx \hat{\alpha}_2 \approx \hat{\alpha}$.

Theorem 1 guarantees that, when both users belong to a botnet, for t large enough, it is possible to write:

$$\hat{\rho}_{\{1,2\}} \approx \mathscr{R}(\hat{\alpha}, \hat{\lambda}_1 + \hat{\lambda}_2) \triangleq \hat{\rho}_{\rm bot}. \tag{2.6}$$

Furthermore, it is possible to show that the empirical MIR of the aggregate subnet $\{1,2\}$ can be upper bounded by the MIR pertaining to the disjoint dictionaries, namely,

$$\begin{aligned}\hat{\rho}_{\{1,2\}} &\le \hat{\rho}_1 + \hat{\rho}_2 = \mathscr{R}(\hat{\alpha}_1, \hat{\lambda}_1) + \mathscr{R}(\hat{\alpha}_2, \hat{\lambda}_2) \triangleq \hat{\rho}_{\rm sum}\\ &\approx \mathscr{R}(\hat{\alpha}, \hat{\lambda}_1) + \mathscr{R}(\hat{\alpha}, \hat{\lambda}_2).\end{aligned} \tag{2.7}$$

Since $\hat{\rho}_{\text{bot}} < \hat{\rho}_{\text{sum}}$ (see [6]), it is possible to introduce a threshold γ lying between $\hat{\rho}_{\text{bot}}$ and $\hat{\rho}_{\text{sum}}$, namely

$$\hat{\rho}_{\text{bot}} < \gamma = \theta\hat{\rho}_{\text{bot}} + (1-\theta)\hat{\rho}_{\text{sum}} < \hat{\rho}_{\text{sum}}, \tag{2.8}$$

where $\theta \in (0,1)$ is a tuning threshold parameter.[1]

When the two users belong to a botnet, it is possible to show that, for large t, the empirical MIR $\hat{\rho}_{\{1,2\}}$ converges to $\hat{\rho}_{\text{bot}}$. In contrast, by exploiting Theorem 1, one can verify that $\hat{\rho}_{\text{sum}} - \hat{\rho}_{\text{bot}}$ converges in probability to a positive quantity, which implies that, for *any* $\theta \in (0,1)$, as time elapses, the empirical MIR will stay sooner (lower θ) or later (higher θ) *below* the threshold γ, yielding:

$$\text{1 AND 2 are bots} \Rightarrow \hat{\rho}_{\{1,2\}} < \gamma \tag{2.9}$$

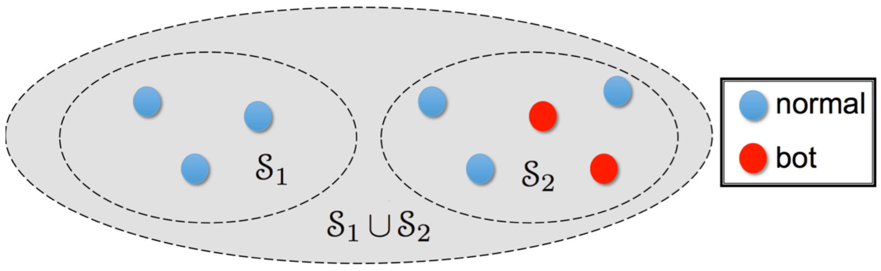
Let us switch to another case: at least one user is normal. In case the dictionaries of the two users would be perfectly disjoint, we would observe that $\hat{\rho}_{\{1,2\}} \approx \hat{\rho}_{\text{sum}} > \gamma$ (for any $\theta \in (0,1)$). Thus, when at least one user is normal, for sufficiently big θ, the empirical MIR is expected to lie above the threshold, namely:

$$\text{1 OR 2 are normal} \Rightarrow \hat{\rho}_{\{1,2\}} > \gamma. \tag{2.10}$$

In summary, in case the empirical MIR lies below γ, we declare that the two users form a botnet, otherwise, we declare that at least one user is normal. This behavior is pictorially sketched in Fig. 2.2.

Before concluding this section, it is useful to remark that in (2.2) the cardinality of the empirical dictionary is divided by the current time epoch, and not by the transmission rate. Accordingly, if we compare two users or subnets featuring very different transmission rates, normalization by the transmission rate would provide a more fair comparison. However, it must be noticed that the BIC is not based upon comparison of MIRs. Rather, the BIC

[1] In [6], the tuning threshold parameter was defined as $\epsilon = 1 - \theta$. The different choice adopted in the present work is more convenient to deal with overlapped emulation dictionaries, as will become clear in chapter 3.



$$\rho_{\text{bot}} \triangleq \underbrace{\frac{\alpha(\lambda_{\mathcal{S}_1}+\lambda_{\mathcal{S}_2})}{\alpha+(\lambda_{\mathcal{S}_1}+\lambda_{\mathcal{S}_2})}}_{\mathcal{S}_1\text{and } \mathcal{S}_2 \text{ form a botnet}} < \underbrace{\rho_{\mathcal{S}_1\cup\mathcal{S}_2}}_{\text{aggregate net MIR}} < \underbrace{\rho_{\mathcal{S}_1}+\rho_{\mathcal{S}_2}}_{\text{independent/disjoint case}} \triangleq \rho_{\text{sum}}$$

Figure 2.2: Pictorial representation of the Botnet Identification Condition

relies on comparing the MIR of a certain subnet to the MIR that this subnet would produce if it were made by bots. In this respect, normalization by the (estimated) transmission rate becomes immaterial.

2.1.3 Consistent Botnet Identification

In [6], an algorithm is provided that guarantees correct botnet identification, assuming that the BIC is satisfied. This result has been proved with reference to the setting where one and the same emulation dictionary is available to all bots. This is a simplifying assumption that will be removed in this thesis. The novel results corresponding to the general case where multiple emulation dictionaries are disseminated across the network are dealt with in great detail in chapter 3. In this section we would like to illustrate briefly the results available in the simplified scenario of randomized L7-DDoS with a unique emulation dictionary.

Two useful indicators are exploited to quantify the performance of a botnet identification algorithm. Considering a network $\mathcal{N} = \{1, 2, \ldots, N\}$ containing a botnet $\mathcal{B}$, and letting $\hat{\mathcal{B}}(t)$ be the botnet estimated at time t by the algorithm, we introduce the following two performance indices:

$$\eta_{\text{bot}}(t) = \frac{\mathbb{E}[|\hat{\mathcal{B}}(t) \cap \mathcal{B}|]}{|\mathcal{B}|}, \quad \eta_{\text{nor}}(t) = \frac{\mathbb{E}[|\hat{\mathcal{B}}(t) \cap (\mathcal{N} \setminus \mathcal{B})|]}{|\mathcal{N} \setminus \mathcal{B}|}. \quad (2.11)$$

The first one represents the expected percentage of *correctly banned users* (i.e., discovered bots), whereas the second one is the expected percentage of incorrectly-banned users (i.e., legitimate users mistakenly declared as bots).

The desired asymptotic behavior of the two indices is: $\eta_{\text{bot}}(t) \to 1$, and $\eta_{\text{nor}}(t) \to 0$. The Authors in [6] formally prove that if the BIC is verified, $\eta_{\text{bot}}(t) \to 1$ and $\eta_{\text{nor}}(t) \to 0$ as $t \to \infty$. Such a behavior is confirmed by some experimental analysis as reported in Figs. 2.3 and 2.4. In both figures we see that $\eta_{\text{bot}}(t) \to 1$ and $\eta_{\text{nor}}(t) \to 0$ as t increases. Moreover, Fig. 2.3 explores differ-

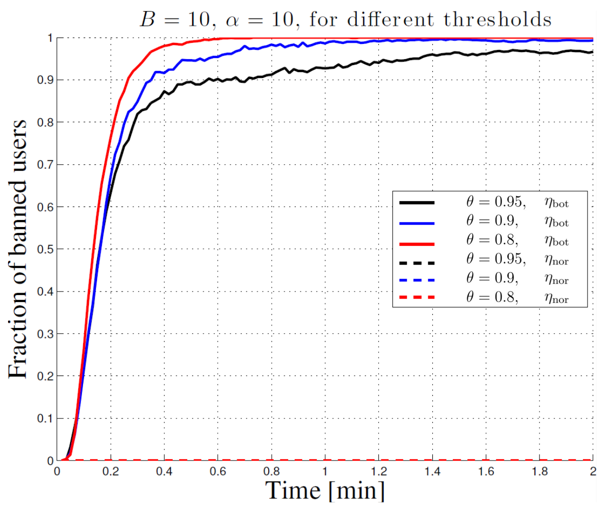


Figure 2.3: Percentage of banned users as a function of time, for different values of the tuning parameter θ. The network is composed of 10 legitimate users, and contains $B = 10$ bots.

ent values of a tunable threshold parameter θ (more details about this parameter will be given in chapter 3). We see that the algorithm exhibits little sensitivity to the choice of this parameter.

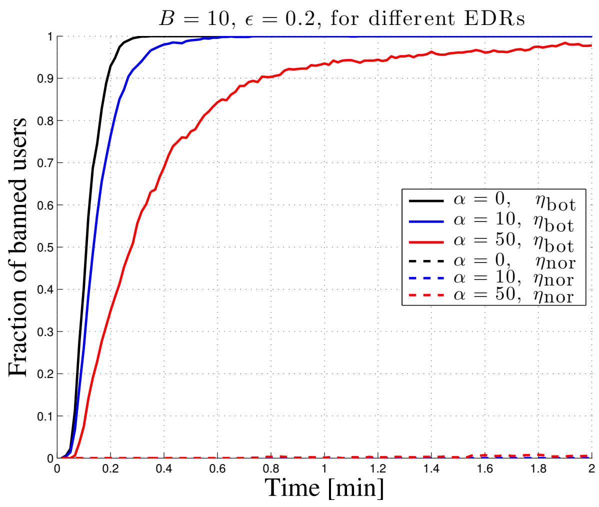


Figure 2.4: Percentage of banned users as a function of time, for different values of the EDR α. The network is composed of 10 legitimate users, and contains $B = 10$ bots.

Figure 2.4 examines instead the effect of different EDRs. Notably, we see that the performance worsens as the EDR increases, which sounds perfectly reasonable since the EDR represents in a sense the attacker's power.

In summary, we have seen that under certain conditions the attacker can be revealed and, hence, successfully defeated. For this to be true, it was critical to exploit the specific constraints at the attacker's side. It is therefore legitimate to ask what happens in the presence of other types of constraints and/or attacking schemes. For example, what if *i*) each bot replays the requests from a different user, or *ii*) different users request the same set of popular objects from the same web server? The scheme advocated in *i*) would require that the bots cooperatively act so as to select deterministically distinct messages, and that the richness of the available messages is so high to prevent from repetition of distinct messages. These are clearly strong assumptions that, if verified, could create perfect independence among bots and, hence, perfect indistinguishability from normal users. For what concerns *ii*), it is clear that the verification of the BIC can depend upon the specific class of websites. However, it should be remarked that nowadays websites try to protect themselves from robot-like behavior by introducing deliberately redundant dynamic content (such as banners). This dynamic content reduces the impact of repetitive behavior, and is in turn responsible for the need of continuous updates of the emulation dictionary to be performed by the botmaster.

2.2 Related Work on Availability

The interest in techniques and methods for evaluating the performance, reliability, and availability of novel communication systems based on cloud models is recently growing [44]. The attention of industry and academia is increasing also in accordance with the new 5G telecommunication networks requirements. In particular, the need to have strict Service Level Agreements (namely, the performance indicators that operators must guarantee) encourages ever deeper analyses about the performability of the systems, where performance and availability are considered in a unified manner ([45, 46]).

One contribution of this thesis focuses on the reliability assessment of network architectures designed to furnish complex services by integrating several interconnected virtualized units or functions. This ensemble of virtualized entities is usually referred to as a Service Function Chain (SFC).

There exist several useful aspects involved in the availability studies of an SFC, for example: the optimal VNF placement [47–49], the optimization of routing paths [50, 51], the design and analysis of provisioning strategies in view of efficient VNF deployment [52, 53].

The élite tool to perform a reliability assessment of complex systems made by interconnected entities is given by a Reliability Block Diagram (RBD), where failure/repair events are conveniently abstracted in terms of a chain of individual series/parallel interconnections. The RBD tool is automatically suited to the SFC structure addressed in this thesis, due to the inherent block structure exhibited by an SFC.

As a matter of fact, the RBD framework has been successfully applied in the context of availability of distributed network architectures. For example, an availability study of a private cloud architecture has been faced in [54], where the RBD framework is used to describe the high-level interconnections between the infrastructure layers, whereas the inner structure of these layers is described by means of a Continuous-Time Markov Chain (CTMC) framework. A similar combined analysis (RBD with CTMC) has been exploited in [55], where a Video-on-Demand service for cloud-based Eucalyptus platform has been taken into account. Again, in [56] the Authors perform a CTMC-Based availability analysis of a multiple cluster system characterized by common mode failures.

One major drawback when applying Markov chains to model complex systems is the largeness of the pertinent state space. In this thesis, the complexity of the infrastructures under analysis (containerized or virtualized network service chains such as the IMS architecture) stems from the need of taking into account: *i*) the failure/repair events of the single layers of a node; *ii*) the common mode failures due to the nested structure among the layers (e.g. the hardware fault implies the hypervisor fault); *iii*) the single point of failures due to the chained structure of nodes connected in series (in fact, the series arrangement is typical in network service chains); and *iv*) the replication of single nodes to guarantee certain availability requirements. Implementing a Markov model able to capture all the aforementioned aspects can be unaffordable. Stochastic Reward Networks emerge as a powerful tool to overcome the aforementioned drawbacks, since they offer the possibility to automate the generation of the underlying Markov chain directly starting from a compact description of the system. In a sense, the SRN formalism provides a high level interface to automatically specify the underlying Markov model. The SRN framework has been profitably exploited in [57], where a stochastic model to characterize the aspects of an IaaS infrastructure has been considered. A technological-based analysis aimed at comparing classic virtualized environments against container infrastructures, according to a high availability perspective, has been carried out in [58], where some open-source and commercial systems have been considered for the comparison, but without proposing a failure/repair mathematical model.

For instance, the Authors in [59] examine availability issues of an SFC, with respect to the minimum number of backup VNFs to deploy. The model proposed in [59] addresses only failure events but not repair actions. Similarly, a repair model is not taken into account in [60], where a framework for a reliability evaluation of a virtualized deployment has been proposed. In [61], the problem of distributing VNF replicas between primary and backup paths aimed at maximizing the SFC availability has been faced via a heuristic algorithm, without taking into account failure/repair models. Another study concerning microservice oriented architectures and inspired to Google Kubernetes services is described in [62], but is limited to an availability analysis, leaving performability for future works. A performability analysis lacks also in [63], where the Stochastic Petri Net framework is adopted, and where a cloud server, a load balancer and a database distribute different requests across VMs to realize a Disaster-Recovery-as-a-Service paradigm.

2.2.1 Stochastic Reward Networks

In this thesis we deal with complex network architectures, thus we find it beneficial to adopt the SRN formalism to take into account the huge number of failure and repair states. Originally conceived for reliability prediction in [64], the SRN model derives from the Markov Reward Model (MRM), which enhances the traditional Continuous-Time Markov Chain (CTMC) by adding a reward rate to each state. With state-space based models (such as MRMs), a classic problem of modeling real-world systems is related to the growth of the state space. In comparison, an SRN-based representation allows a more compact description of the underlying system,

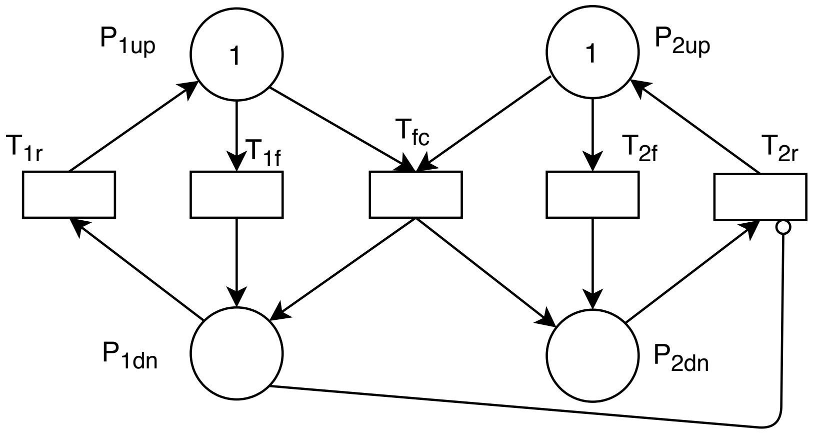


Figure 2.5: The SRN model of a network node consisting of two parts P_1 and P_2, as described in the main text.

by identifying repetitive structures. In this way, it is possible to automatically generate the underlying MRMs [65]. To this end, the SRN model adopts a bipartite directed graph representation where: *i) places* (represented by circles) specify a condition (e.g., the system is *down* or *up*); and *ii) transitions* (represented by rectangles) denote actions (e.g., a system crash). Places and transitions are connected by *arcs* denoted by directed edges. In the SRN formalism, the transition times take into account *probabilistic* delays, and are particularly modeled as exponential random variables. A place typically contains a number called *token*, which represents a holding condition. In case of a condition change, a transition is *fired*, and the token is moved from one place to another. A measure of interest is the distribution of tokens, called *marking*, that denotes the possible assignment of tokens to all places of the underlying Markov model, and is useful to capture the dynamics of the overall system.

It is useful to examine a simple example of an SRN. Let us consider a basic network node consisting of two parts, say P_1 and P_2. We consider the node to be working as long as one of two parts is working. We assume that failures and repairs of the two parts are exponentially distributed random variables, and there is a failure mode where both parts can fail simultaneously. Moreover, we assume the priority repair of P_1 over P_2. Translating such a process into the corresponding SRN is quite straightforward, and the resulting graphical representation in shown in Fig. 2.5. The four places P_{1up}, P_{2up}, P_{1dn} and P_{2dn} represent the conditions in which the parts P_1 and P_2 are up or down, respectively. The transitions T_{1f} and T_{2f} denote the failure of the parts P_1 and P_2, respectively; the transitions T_{1r} and T_{2r} denote the repair of the parts P_1 and P_2, respectively; the transition T_{fc} denotes the so-called common-mode failure. The number inside the places P_{1up} and P_{2up} is the token. The particular arc drawn from P_{1dn} to T_{2r} is called an *inhibitory arc*. Such an arc is useful to encode into the SRN representation the priority repair of P_1 over P_2. It means that, in case of failure of P_1, P_2 cannot work until P_1 has been repaired.

Metrics such as availability, reliability and performability are computed by assigning appropriate *reward rates* to the states of the SRN. Precisely, we have to identify the states in which the node is working, and assign to them a reward rate amounting to 1. Conversely, a reward rate amounting to 0 is assigned to the other states. In the SRN of Fig. 2.5, the node is working as long as there exists a token in either of the two places, P_{1up} or P_{2up}. Accordingly, the reward rate assignment can be specified as follows:

$$r_i = \begin{cases} 1 & \text{if} \quad (\#P_{1up} = 1) \vee (\#P_{2up} = 1) \text{ in marking } i, \\ 0 & \text{otherwise.} \end{cases} \tag{2.12}$$

Here, r_i is the reward rate assigned to the state i of the SRN, and $\#P$ represents the number of tokens in the place P. The operator $\vee$ represents the logical OR. The availability of the node at time t is defined as the expected instantaneous reward rate $\mathbb{E}[X(t)]$ at time t [64]. Thus, we can write:

$$A(t) = \mathbb{E}[X(t)] = \sum_{i \in \mathbb{I}} r_i \, p_i(t), \tag{2.13}$$

where $\mathbb{I}$ is the set of markings, and $p_i(t)$ is the probability of being in state i at time t.

Chapter 3

DDoS with Multiple Emulation Dictionaries

The botnet identification strategies considered in previous works assume either that the emulation dictionary is *one and the same for all bots* [6, 37] or that different groups of bots can use *completely disjoint* dictionaries [38]. These assumptions appear somehow restrictive, since in practice it is expected that coordination among the bots is only partial, for different reasons, including: decentralized control of the botnet, decentralized learning of the emulation dictionary, imperfect communication. In comparison, it is more realistic to assume that, inside the botnet, there exist clusters of bots that pick messages from the same emulation dictionary, but that at the same time a certain degree of *diversity* exists among the dictionaries of different clusters. Such augmented diversity implies in turn an augmented variability in the traffic patterns produced by the botnet, making it more difficult to distinguish them from legitimate users. In this chapter we tackle this issue and provide the following main contributions. We obtain an analytical characterization of the *message innovation rate* of the DDoS attack with multiple emulation dictionaries. Exploiting this result, we design a botnet identification algorithm equipped with a *cluster expurgation rule*, which, under appropriate technical conditions, is shown to provide exact classification of bots and normal users as the observation window size increases. Then, an experimental campaign over real network traces is conducted to assess the validity of the theoretical analysis, as well as to examine the effect of a number of non-ideal effects that are unavoidably observed in practical scenarios.

3.1 Model

When formalizing the aforementioned new model with multiple emulation dictionaries, we can use the empirical transmission rate as defined in (2.1), and the empirical MIR as defined in (2.2). The third descriptor (EDR) requires a different characterization w.r.t. the original one presented in (2.3). Let $\mathcal{B} \subseteq \mathcal{N}$ denote the ensemble of all bots disseminated across the network. We assume that the botmaster has collected a set of messages available for the purpose of emulating legitimate traffic patterns. The botmaster distributes fairly the dictionary across the botnet, in such a way that different clusters of bots receive approximately equal-sized portions of the dictionary. Formally, the botnet is partitioned into C clusters and at time t the bots belonging to the i-th cluster can access an emulation dictionary $\mathscr{E}_i(t)$, whose cardinality grows over time at rate:

$$\alpha \triangleq \lim_{t\to\infty} \frac{|\mathscr{E}_i(t)|}{t}, \quad i = 1, 2, \ldots, C, \tag{3.1}$$

which is the considered EDR. In order to send apparently legitimate requests, a bot belonging to the i-th cluster picks admissible messages from $\mathscr{E}_i(t)$. We remark that the botmaster must steadily learn new admissible messages (i.e., the cardinality of the emulation dictionary must increase over time) in order to guarantee a non-suspicious innovation rate.

One fundamental feature of this work is that we allow for *non-negligible interaction between distinct clusters.* More specifically, the emulation dictionaries of two clusters i and j have an intersection given by:

$$\mathscr{E}_{ij}(t) \triangleq \mathscr{E}_i(t) \cap \mathscr{E}_j(t), \tag{3.2}$$

whose *overlap degree* is quantified as:

$$\lim_{t\to\infty} \frac{|\mathscr{E}_{ij}(t)|}{|\mathscr{E}_i(t)|} = \lim_{t\to\infty} \frac{|\mathscr{E}_{ij}(t)|}{|\mathscr{E}_j(t)|} = \omega_{ij} = \omega_{ji} \in (0,1), \tag{3.3}$$

yielding, in view of (3.1), the EDR of the intersection:

$$\lim_{t\to\infty} \frac{|\mathscr{E}_{ij}(t)|}{t} = \alpha\,\omega_{ij}. \tag{3.4}$$

One example of the scenario with multiple dictionaries is illustrated in Fig. 3.1, with reference to three clusters. Even though our model allows intersection among more than two dictionaries (as shown in the top panel), the *pairwise* overlap degrees ω_{ij} (shown in the bottom panel) will play a critical role in the botnet identification algorithm, as explained in the forthcoming section.

3.2 Pairwise Cluster Interaction

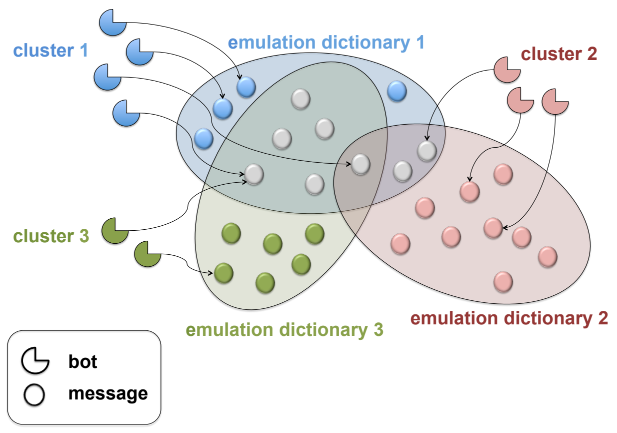
Preliminarily, it is useful to introduce the following function, for $\alpha, \lambda > 0$:

$$\mathscr{R}(\alpha, \lambda) \triangleq \frac{\alpha\lambda}{\alpha + \lambda}. \tag{3.5}$$

In [6] it is shown that this function corresponds to the MIR of a botnet (using one and the same emulation dictionary) with EDR α and transmission rate λ, under either a synchronous or Poisson scheduling. For later use, we report also the following inequality, holding for any choice of positive $\alpha, \lambda_1, \lambda_2$ [6]:

$$\mathscr{R}(\alpha, \lambda_1 + \lambda_2) \leq \mathscr{R}(\alpha, \lambda_1) + \mathscr{R}(\alpha, \lambda_2), \tag{3.6}$$

which reflects the fact that the innovation rate of a botnet made of two components (both using the same emulation dictionary with EDR α, and transmitting at rates λ_1 and λ_2, respectively) is up-



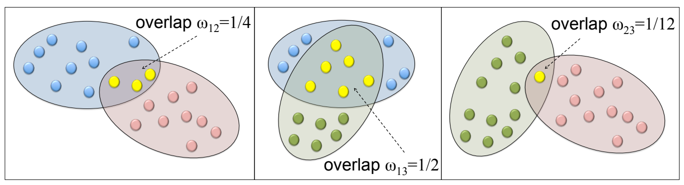

overlap ω_{12}=1/4
overlap ω_{13}=1/2
overlap ω_{23}=1/12

Figure 3.1: *Top.* Illustration of the DDoS attack with multiple emulation dictionaries. *Bottom.* Pairwise overlaps relative to the setting displayed in the top.

per bounded by the sum of the innovation rates of the individual components.

The following theorem establishes the coupling effect between clusters that arises from the nonzero overlap between their emulation dictionaries.

Theorem 1 (Pairwise MIR)**.** *Let $\mathcal{B}_i$ and $\mathcal{B}_j$ be subsets of the i-th and j-th cluster, respectively. Assume that the transmission policies of all bots are either synchronous (having constant transmission rate) or governed by independent Poisson schedulings. Then, the (limiting) MIR of the joint subnet $\mathcal{B}_i \cup \mathcal{B}_j$ is (the symbol $\xrightarrow{\text{m.s.}}$*

denotes mean-square convergence as $t \to \infty$*):*

$$\begin{aligned}\widehat{\rho}_{\mathcal{B}_i \cup \mathcal{B}_j}(t) \quad &\xrightarrow{\text{m.s.}} \quad \rho_{\mathcal{B}_i \cup \mathcal{B}_j} = \omega_{ij}\, \mathscr{R}(\alpha, \lambda_{\mathcal{B}_i} + \lambda_{\mathcal{B}_j}) \\ &+ \quad (1-\omega_{ij})\, [\mathscr{R}(\alpha, \lambda_{\mathcal{B}_i}) + \mathscr{R}(\alpha, \lambda_{\mathcal{B}_j})]. \qquad (3.7)\end{aligned}$$

The proof is reported in Appendix A

Equation (3.7) admits a clear interpretation. The MIR of the botnet $\mathcal{B}_i \cup \mathcal{B}_j$, which aggregates bots from clusters i and j, is a convex combination (with weights ω_{ij} and $1-\omega_{ij}$) of two types of MIRs. A fraction ω_{ij} of the aggregate MIR is given by the MIR corresponding to a botnet with transmission rate $\lambda_{\mathcal{B}_i} + \lambda_{\mathcal{B}_j}$, whose members pick messages from the same emulation dictionary. In comparison, a fraction $1-\omega_{ij}$ of the aggregate MIR is given by the *sum* of the MIRs corresponding to $\mathcal{B}_i$ and $\mathcal{B}_j$. This fraction represents messages *not* picked from the intersection between the two emulation dictionaries.

It is also useful to compare (3.7) with the MIRs corresponding to the cases of total overlap and absence of overlap, respectively. We have from (3.6) and (3.7):

$$\mathscr{R}(\alpha, \lambda_{\mathcal{B}_i} + \lambda_{\mathcal{B}_j}) \le \rho_{\mathcal{B}_i \cup \mathcal{B}_j} \le \mathscr{R}(\alpha, \lambda_{\mathcal{B}_i}) + \mathscr{R}(\alpha, \lambda_{\mathcal{B}_j}). \qquad (3.8)$$

In other words, the aggregate botnet $\mathcal{B}_i \cup \mathcal{B}_j$ is: *i*) *more innovative* than a botnet where both clusters pick messages from the same emulation dictionary (left inequality in (3.8), total overlap); and *ii*) *less innovative* than a botnet where the clusters pick messages from disjoint dictionaries (right inequality in (3.8), no overlap at all).

3.3 Botnet Identification Algorithm

A botnet identification algorithm nicknamed *BotBuster* was proposed in [6]. This algorithm is able to detect a botnet possibly hidden in the network *under the assumption that there is a single emulation dictionary common to all bots (i.e., no multi-cluster allowed)*, and with detection error that vanishes as the observation time grows. We now manage to illustrate how efficient botnet identification can be achieved under the multi-cluster setting addressed in this work.

To this aim, we build an algorithm nicknamed *BotCluster-Buster*, which is basically composed of two stages. The first stage pertains to the *formation of candidate bot clusters*. The routine implemented at this stage is inherited from BotBuster, which, as said, was designed for the single-cluster case. Proving that this routine can be useful in the multi-cluster setting as well, requires significant effort and a novel analysis that relies primarily on the results contained in Theorem 1. The second stage pertains to *cluster expurgation*, a procedure necessary to discard possibly fake clusters from the set of candidates produced in the first stage.

3.3.1 Cluster Formation

The core of BotClusterBuster is a pairwise comparison between a *pivot* element (a single node or an estimated botnet) and a test node. The final goal is to decide whether or not the pivot and the test node form a botnet. Let us start by considering a pivot node p, and a test node τ.

First, the algorithm computes the empirical MIRs corresponding to node p, to node τ, and to their union, namely, $\widehat{\rho}_{\{p\}}(t)$, $\widehat{\rho}_{\{\tau\}}(t)$, and $\widehat{\rho}_{\{p,\tau\}}(t)$ — see 2.2. Then, since the EDR is not known beforehand, and since when it is estimated from the data one gets distinct values for the two nodes p and τ (either because one or both nodes correspond to normal users, or because of estimation errors), it is necessary to compute a common reference EDR. This is not an easy task, since, e.g., intuitive solutions such as the average between the EDRs of the individual nodes might lead to inconsistent results — see [6]. In order to overcome this issue, in [6] a *Replacement and Reassignment* (R&R) procedure is proposed. This procedure relies on the intuition that, if some messages from the node with higher estimated EDR are *fictitiously* reassigned to the other node, the resulting EDRs tend to move close to each other, until a common EDR $\widehat{\alpha}(t)$ is reached. Once this reference

EDR is obtained, BotClusterBuster computes the MIR that would correspond to a botnet with *one and the same* emulation dictionary (*total overlap*) having reference EDR $\widehat{\alpha}(t)$, namely,

$$\widehat{\rho}_{\text{tot}}(t) \triangleq \mathscr{R}\left(\widehat{\alpha}(t), \widehat{\lambda}_{\{p,\tau\}}(t)\right). \tag{3.9}$$

We further introduce the sum MIR:

$$\widehat{\rho}_{\text{sum}}(t) \triangleq \widehat{\rho}_{\{p\}}(t) + \widehat{\rho}_{\{\tau\}}(t). \tag{3.10}$$

Let us now delve into the analysis of the possible cases that the algorithm encounters when testing nodes p and τ. Depending on the nature of these nodes, there are three possibilities — see the summary in Table 3.1.

— *Case I.* If p OR τ are normal, we have that:

$$\widehat{\rho}_{\{p,\tau\}}(t) \approx \widehat{\rho}_{\text{sum}}(t), \tag{3.11}$$

which intuitively stems from the fact that low correlation is observed between normal users and between normal users and bots. Moreover, it can be shown that [6]:

$$\widehat{\rho}_{\text{tot}}(t) \leq \widehat{\rho}_{\text{sum}}(t). \tag{3.12}$$

— *Case II.* If p AND τ are bots from the same cluster, for sufficiently large t we have $\widehat{\alpha}(t) \approx \alpha$ and $\widehat{\lambda}_{\{p,\tau\}}(t) \approx \lambda_p + \lambda_\tau$, yielding:

$$\widehat{\rho}_{\{p,\tau\}}(t) \approx \widehat{\rho}_{\text{tot}}(t) \approx \mathscr{R}(\alpha, \lambda_p + \lambda_\tau). \tag{3.13}$$

— *Case III.* If p AND τ are bots from distinct clusters i and j, from Theorem 1 we have:

$$\begin{aligned}
&\widehat{\rho}_{\{p,\tau\}}(t) \\
&\quad\approx\quad \omega_{ij}\mathscr{R}(\alpha, \lambda_p + \lambda_\tau) + (1 - \omega_{ij})[\mathscr{R}(\alpha, \lambda_p) + \mathscr{R}(\alpha, \lambda_\tau)] \\
&\quad\approx\quad \omega_{ij}\, \widehat{\rho}_{\text{tot}}(t) + (1 - \omega_{ij})\, \widehat{\rho}_{\text{sum}}(t),
\end{aligned} \tag{3.14}$$

where in the second approximate equality we exploited the rela-

Nodes p, τ	Indicators
I. at least one normal	$\widehat{\rho}_{\{p,\tau\}}(t) \approx \widehat{\rho}_{\mathrm{sum}}(t)$
II. bots from the same cluster	$\widehat{\rho}_{\{p,\tau\}}(t) \approx \widehat{\rho}_{\mathrm{tot}}(t)$
III. bots from clusters i and j	$\widehat{\rho}_{\{p,\tau\}}(t) \approx \omega_{ij}\widehat{\rho}_{\mathrm{tot}}(t) + (1-\omega_{ij})\widehat{\rho}_{\mathrm{sum}}(t)$

Table 3.1: Behavior of the pertinent MIRs for the three cases described in the main text.

tionship $\widehat{\rho}_{\mathrm{tot}}(t) \approx \mathscr{R}(\alpha, \lambda_p + \lambda_\tau)$, which can be proved as in case II. Indeed, by examining the R&R procedure it is readily seen that convergence of $\widehat{\alpha}(t)$ to α is not affected by the fact that p and τ use distinct emulation dictionaries.

Assume now that we set a threshold equal to, for $\theta \in (0,1)$:

$$\gamma(t) = \theta\, \widehat{\rho}_{\mathrm{tot}}(t) + (1-\theta)\, \widehat{\rho}_{\mathrm{sum}}(t), \tag{3.15}$$

and that the algorithm adopts the following rule:

$$\widehat{\rho}_{\{p,\tau\}}(t) \leq \gamma(t) \Rightarrow \text{form an estimated botnet } \{p,\tau\}. \tag{3.16}$$

Let us examine how such classification rule works under the aforementioned three cases — see Fig. 3.2. When p or τ are normal users (case I), from (3.11), (3.15) and (3.16) we conclude that the algorithm rejects the hypothesis that $\{p,\tau\}$ is a botnet, since the empirical MIR $\widehat{\rho}_{\{p,\tau\}}(t)$ will be sufficiently close to the upper boundary $\widehat{\rho}_{\mathrm{sum}}(t)$. Likewise, when p and τ are bots sharing the same emulation dictionary (case II), from (3.13), (3.15) and (3.16) we conclude that the algorithm accepts the hypothesis that $\{p,\tau\}$ is a botnet, since the empirical MIR $\widehat{\rho}_{\{p,\tau\}}(t)$ will be sufficiently close to the lower boundary $\widehat{\rho}_{\mathrm{tot}}(t)$. Finally, let us examine the situation where p and τ are bots from distinct clusters (case III). Assume that:

$$\theta > \omega_{ij}. \tag{3.17}$$

In this case, we see from (3.14), (3.15) and (3.16) that the algorithm rejects the hypothesis that $\{p,\tau\}$ is a botnet, since $\widehat{\rho}_{\{p,\tau\}}(t)$

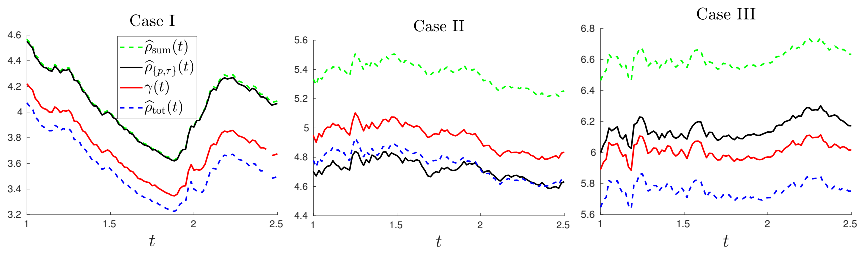


Figure 3.2: Time evolution of the relevant MIRs, for the three cases described in the main text.

will stay above the threshold. In summary, we see that under condition (3.17) the algorithm ends up estimating a botnet only when p and τ belong to the same cluster.

Iterating the above procedure yields the BotClusterBuster algorithm (pseudo-code at the top of next page). The algorithm starts with node 1 as a pivot. If nodes 1 and 2 are classified as a botnet, $\widehat{\mathcal{B}}(1;t) = \{1,2\}$, otherwise $\widehat{\mathcal{B}}(1;t) = \{1\}$. At the end of the loop, the algorithm returns a candidate botnet cluster $\widehat{\mathcal{B}}(1;t)$ (in case the candidate cluster has cardinality equal to 1, it is discarded). The loop is iterated across the whole set of pivots, yielding a sequence of candidate clusters, namely, $\widehat{\mathcal{B}}(1;t), \widehat{\mathcal{B}}(2;t), \dots, \widehat{\mathcal{B}}(N;t)$.

We remark that the analysis of cases I–III is basically unchanged if we replace the pivot node p with a subnet of bots belonging to the same cluster. Accordingly, since for large t the pairwise checks are all correct, they can produce either empty sets or subnets of bots belonging to the same cluster. In particular: i) if the initial pivot node is a normal user, BotClusterBuster produces an empty cluster; ii) if the initial pivot node is a bot from a certain cluster, BotClusterBuster produces a candidate cluster matching the cluster the pivot belongs to. Thus, provided that (3.17) is verified, as $t \to \infty$ the algorithm ends up estimating all the botnets corresponding to the different clusters, reaching the twofold goal of discriminating bots from normal users and identifying the local structure of the individual clusters. Remarkably, condition (3.17)

```
**Algorithm:** $\widehat{\mathcal{B}}(t)$=BotClusterBuster(traffic patterns until time $t$, $\theta$, $\kappa$, $\xi$)
$\mathcal{N} = \{1, 2, \ldots, N\}$
for $p \in \mathcal{N}$ do
    $\widehat{\mathcal{B}}(p;t) = \{p\}$
    for $\tau \in \mathcal{N} \setminus \{p\}$ do
        $\gamma(t) = \theta\widehat{\rho}_{\rm tot}(t) + (1-\theta)\widehat{\rho}_{\rm sum}(t)$
        if $\widehat{\rho}_{\widehat{\mathcal{B}}(p;t)\cup\{\tau\}}(t) \le \gamma(t)$ then $\widehat{\mathcal{B}}(p;t) = \widehat{\mathcal{B}}(p;t)\bigcup\{\tau\}$
    end
    if $|\widehat{\mathcal{B}}(p;t)| = 1$ then $\widehat{\mathcal{B}}(p;t) = \emptyset$
    % begin cluster expurgation
    if $\widehat{\lambda}_{\widehat{\mathcal{B}}(p;t)}(t) \le \frac{\xi\kappa}{1+\kappa}\widehat{\lambda}_{\mathcal{N}}(t)$ then $\widehat{\mathcal{B}}(p;t) = \emptyset$
    % end cluster expurgation
end
$\widehat{\mathcal{B}}(t) = \bigcup_{p=1}^{N} \widehat{\mathcal{B}}(p;t)$
```

does not require detailed knowledge of the degree of overlap ω_{ij}. A rough prediction is sufficient, so as to let $\theta > \omega_{ij}$.

3.3.2 Cluster Expurgation

According to the analysis reported in the previous section, the first stage of BotClusterBuster ends up delivering one (possibly empty) candidate cluster per each initial pivot node. The most natural way to build the final estimated botnet is to retain all the (non-empty) botnet clusters, namely, to consider their union (*union-rule*). From a theoretical standpoint such rule would provide perfect botnet identification. However, in practice there are several non-ideal effects that can induce coupling between normal users and/or between normal users and bots. First of all, our analysis guarantees convergence of certain relevant quantities for sufficiently long time, and in practice we must face issues related to finite observation windows. Second, the condition that normal users (or normal users and bots) are mutually independent (yielding an aggregate MIR approximately equal to $\widehat{\rho}_{\rm sum}(t)$, hence resulting into an easy threshold up-crossing) cannot be perfectly met in practice. Occasionally, we can have normal users that appear correlated to other normal users or to some bots. As a matter of fact, the algorithm usually delivers the true clusters *plus* some fake (small) clusters containing legitimate network users, due to spurious correlations arising during the pairwise checks. In the traditional single-cluster setting, this issue is easily remediated by retaining only the estimated botnet with largest cardinality (*max-rule*). This rule would obviously fail in the multi-cluster scenario, since it will surely discard all but one cluster.

In order to face this issue, the BotClusterBuster algorithm implements a *cluster expurgation* strategy, aimed at: *i*) retaining the true clusters, while *ii*) discarding the fake ones. In order to reach these goals, a first important step is to observe that the power of a DDoS attack is determined by its global transmission rate, $\lambda_{\mathcal{B}}$. More precisely, we are under a (meaningful) DDoS attack if the botnet activity is at least comparable to the normal users' activity, namely if, for some $\kappa \geq 1$, we have that:

$$\lambda_{\mathcal{B}} = \kappa \, \lambda_{\mathcal{N} \setminus \mathcal{B}}, \tag{3.18}$$

with $\lambda_{\mathcal{N} \setminus \mathcal{B}}$ being the global transmission rate of normal users. The lower bound $\kappa = 1$ corresponds to an optimistic assumption (at the botnet side) that a DDoS with relatively low rate is sufficient to damage the target site. By noticing that $\lambda_{\mathcal{N} \setminus \mathcal{B}} = \lambda_{\mathcal{N}} - \lambda_{\mathcal{B}}$, we can reformulate (3.18) in terms of the global network transmission rate:

$$\lambda_{\mathcal{B}} = \frac{\kappa}{1+\kappa} \lambda_{\mathcal{N}}. \tag{3.19}$$

A second critical observation is that, in order to deserve countermeasures, a botnet cluster must sustain a reasonable part of the global attack rate, namely,

$$\lambda_{\mathcal{B}_i} \geq \xi \lambda_{\mathcal{B}}, \tag{3.20}$$

for some $\xi \in (0, 1)$. Joining (3.19) with (3.20), the algorithm retains the candidate cluster $\widehat{\mathcal{B}}(p; t)$ if:

$$\widehat{\lambda}_{\widehat{\mathcal{B}}(p;t)}(t) > \frac{\xi \kappa}{1+\kappa} \, \widehat{\lambda}_{\mathcal{N}}(t), \tag{3.21}$$

otherwise, the candidate cluster is discarded. Finally, the estimate of the entire botnet is produced by applying the union operator to the survived clusters.

It is useful to comment on the role of the hyperparameters κ and ξ. First of all, since in the overall check rule they play jointly in the single parameter $\frac{\xi\kappa}{1+\kappa}$, we have actually a single hyperparameter. We choose to keep both since they can be more easily related to "physical" considerations concerning the size and impact of the cyber-threat. Actually, the value of κ determines a safeguard level at which we judge a botnet dangerous in terms of resource saturation. Likewise, the value of ξ determines to what extent we can tolerate to miss some (small) bot cluster. Remarkably, setting these hyperparameters introduces a flexible degree of customization that is more related to the type of service and users' expectation for a particular application. For example, the typical choice of a network administrator facing a DDoS attack are: *i*) avoiding that the destination site crashes; *ii*) guaranteeing proper service to the legitimate users. Under these conditions, it is preferable not to be indulgent in cluster acceptance, so as to reduce the likelihood of banning normal users (perhaps at the price of losing some bots).

3.4 Practical Issues

In real-world settings, we cannot expect that the technical conditions used to prove our mathematical claims are perfectly met. For this reason, in this section we examine some relevant scenarios that are not covered by the previous analysis, and verify whether BotClusterBuster can deliver satisfying performance even under these perturbed conditions.

3.4.1 Violation of Condition (3.17)

The choice of the parameter θ in (3.17) can be determined by different factors. Preliminarily, we notice that the case $\omega_{ij} \approx 1$ (total overlap) is of scarce interest here, since it basically corresponds to the single cluster case already dealt with in [6]. Thus, let us assume that ω_{ij} takes on some intermediate value. On one hand, we could conservatively set the threshold parameter θ close to 1 in order to be sure that (3.17) holds true. On the other hand, if θ collapses to 1, the threshold $\gamma(t)$ collapses to the lower boundary $\widehat{\rho}_{\mathrm{tot}}(t)$, thus reducing the likelihood that the estimated MIR $\widehat{\rho}_{\{p,\tau\}}(t)$ stays below the threshold. Accordingly, there is a certain flexibility in the choice of θ that should be taken into account in practice. For this reason, it makes sense to examine whether the algorithm is robust to violation of (3.17).

In presence of this violation, it is readily seen that, among cases I–III described before, only case III changes. In fact, in cases I and II (provided that θ is not collapsing to 0 or 1), the estimated MIR $\widehat{\rho}_{\{p,\tau\}}(t)$ stays above or below the threshold, respectively. In contrast, when case III is in force, nodes p and τ belong to distinct clusters with partially overlapped emulation dictionaries, which implies that (since (3.17) is violated) the threshold $\gamma(t)$ becomes larger than the MIR associated to the botnet $\{p, \tau\}$ — see (3.14) and (3.15). Accordingly, the algorithm would classify the pair $\{p, \tau\}$ as a botnet even if p and τ belong to distinct clusters. In principle, this is not a problem, since what we need is to distinguish bots from normal users, and not to separate the bot clusters. This property is illustrated in Fig. 3.3, where we see the two possible behaviors of the algorithm, depending on whether θ is greater or smaller than the overlap degree ω_{ij}. In summary, if $\theta < \omega_{ij}$, the algorithm is able to identify membership to the (overall) botnet, while if $\theta > \omega_{ij}$ the algorithm is able to identify (the additional attribute of) membership to the individual cluster.

Unfortunately, the above analysis (which is restricted to pair of nodes) is not sufficient to establish consistency of BotClusterBuster, for the following reason. Owing to the impossibility of discriminating the individual clusters, when the algorithm progresses the intermediate pivot botnets can be *mixed*, i.e., comprising bots of different clusters. The technical analysis of this case appears to be nontrivial. For this reason, we are not in the position of claiming that BotClusterBuster learns consistently the true bot-

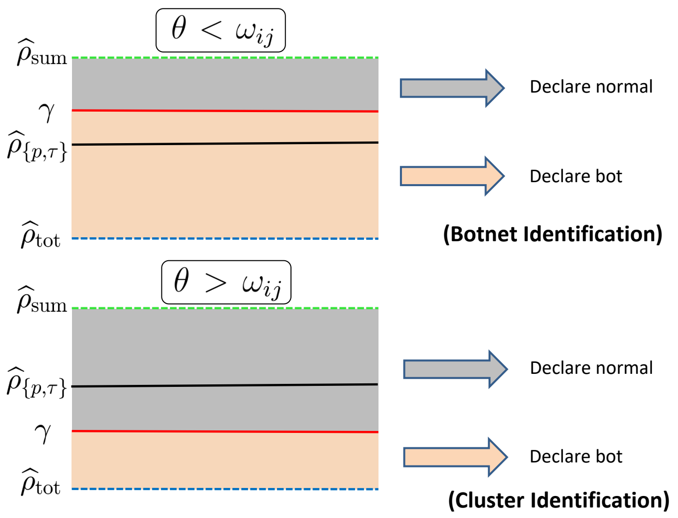

$\theta < \omega_{ij}$
$\widehat{\rho}_{\text{sum}}$
γ
$\widehat{\rho}_{\{p,\tau\}}$
$\widehat{\rho}_{\text{tot}}$
Declare normal
Declare bot
(Botnet Identification)
$\theta > \omega_{ij}$
$\widehat{\rho}_{\text{sum}}$
$\widehat{\rho}_{\{p,\tau\}}$
γ
$\widehat{\rho}_{\text{tot}}$
Declare normal
Declare bot
(Cluster Identification)

Figure 3.3: Botnet identifiability vs. cluster identifiability.

net when (3.17) is violated.

3.4.2 Clusters' Asymmetries

Another assumption adopted so far pertains to the equal distribution of the emulation dictionaries across the network. Assume instead that different clusters are assigned unbalanced emulation dictionaries, yielding different EDRs, namely, for $i = 1, 2, \dots, C$:

$$\alpha_i = \lim_{t\to\infty} \frac{|\mathscr{E}_i(t)|}{t}. \tag{3.22}$$

Likewise, we now allow for variable overlap degrees. Formally, the common intersection $\mathscr{E}_{ij}(t)$ of the dictionaries pertaining to clusters i and j has a degree of overlap with $\mathscr{E}_i(t)$ and $\mathscr{E}_j(t)$ given by, respectively:

$$\omega_{ij} = \lim_{t\to\infty} \frac{|\mathscr{E}_{ij}(t)|}{|\mathscr{E}_i(t)|}, \quad \omega_{ji} = \lim_{t\to\infty} \frac{|\mathscr{E}_{ij}(t)|}{|\mathscr{E}_j(t)|}. \tag{3.23}$$

We notice that simultaneous verification of (3.22) and (3.23) entails the following relationship:

$$\alpha_{ij} \triangleq \lim_{t\to\infty} \frac{|\mathscr{E}_{ij}(t)|}{t} = \alpha_i \omega_{ij} = \alpha_j \omega_{ji}. \tag{3.24}$$

Actually, the proof of Theorem 1 does not require any symmetry assumption. In particular, the proof reported in Appendix A is carried out under the aforementioned asymmetric scenario, and the resulting MIR referred to clusters i and j is:

$$\begin{aligned} \rho_{\mathcal{B}_i \cup \mathcal{B}_j} &= \mathscr{R}(\alpha_{ij}, \lambda_{\mathcal{B}_i}\omega_{ij} + \lambda_{\mathcal{B}_j}\omega_{ji}) \\ &+ (1-\omega_{ij})\,\mathscr{R}(\alpha_i, \lambda_{\mathcal{B}_i}) + (1-\omega_{ji})\,\mathscr{R}(\alpha_j, \lambda_{\mathcal{B}_j}). \end{aligned} \tag{3.25}$$

Despite the fact that the limiting MIR corresponding to partially overlapped dictionaries can be precisely evaluated using (3.25), in the asymmetric scenario it is hard to exploit this relationship to prove consistency of BotClusterBuster. In fact, the asymmetry introduced by the (unknown) parameters $\alpha_i, \alpha_j, \omega_{ij}, \omega_{ji}$, precludes the simple extension of the arguments used in Sec. 3.3.1 to examine cases I–III. For example, when $\alpha_i \neq \alpha_j$ we cannot claim that the reference EDR $\widehat{\alpha}(t)$ obtained from the R&R procedure converges to some known value. As a result, we are no longer in the position of establishing the limiting behavior of $\widehat{\rho}_{\mathrm{tot}}(t)$, and, hence, of the threshold $\gamma(t)$. Nevertheless, we have collected numerous numerical evidences where (for various choices of the parameters characterizing the aforementioned asymmetric scenario) the following inequality is observed:

$$\widehat{\rho}_{\mathcal{S}_a \cup \mathcal{S}_b}(t) > \widehat{\rho}_{\mathrm{tot}}(t) = \mathscr{R}(\widehat{\alpha}(t), \lambda_{\mathcal{S}_a} + \lambda_{\mathcal{S}_b}), \tag{3.26}$$

for two subnets $\mathcal{S}_1$ and $\mathcal{S}_2$, each one containing bots belonging to distinct clusters i and j. Since *i*) Eq. (3.26) implies that the empirical MIR corresponding to $\mathcal{S}_a \cup \mathcal{S}_b$ is larger than the MIR associated to the reference EDR $\widehat{\alpha}(t)$; and *ii*) the empirical MIR $\widehat{\rho}_{\mathcal{S}_a \cup \mathcal{S}_b}(t)$ is obviously upper bounded by the sum of the individual

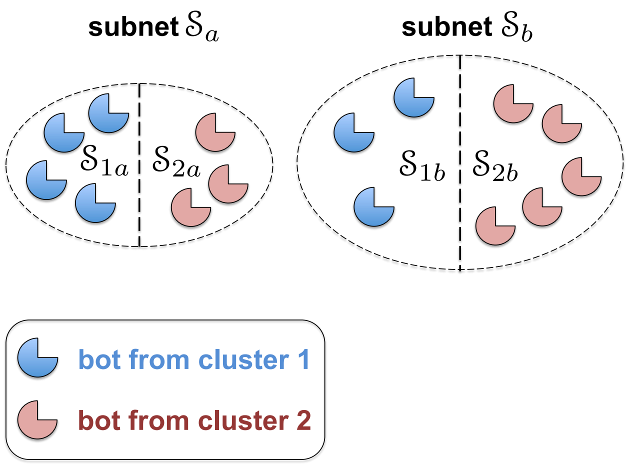


Figure 3.4: Graphical illustration of the scenario described in Sec. 3.4.3, with two subnetworks, $\mathcal{S}_a = \mathcal{S}_{1a} \cup \mathcal{S}_{2a}$ and $\mathcal{S}_b = \mathcal{S}_{1b} \cup \mathcal{S}_{2b}$, each one containing bots belonging to clusters 1 and 2.

MIRs, we conclude that under these conditions the evolution of BotClusterBuster would be basically unchanged.

3.4.3 Numerical Analysis

As remarked in Secs. 3.4.1 and 3.4.2, a rigorous treatment of the non-ideal scenarios described in these sections is definitely nontrivial. For this reason, we now focus on a numerical analysis aimed at capturing the expected behavior of BotClusterBuster under these non-ideal scenarios.

We have examined the MIR arising from pairs of subnets (of mixed type, i.e., each one containing bots from two distinct clusters), and for several sets of parameters. The considered scenario is illustrated in Fig. 3.4. We have two subnets:

$$\mathcal{S}_a = \mathcal{S}_{1a} \cup \mathcal{S}_{2a}, \quad \mathcal{S}_b = \mathcal{S}_{1b} \cup \mathcal{S}_{2b}, \tag{3.27}$$

where each subnet is made of bots from both clusters 1 and 2 (the subscripts 1 and 2 are used accordingly to denote subnets belonging to the pertinent clusters). Now, let α_1 and $\lambda_1 = \lambda_{\mathcal{S}_{1a}} + \lambda_{\mathcal{S}_{1b}}$ be, respectively, the EDR and the transmission rate of the botnet $\mathcal{S}_{1a} \cup \mathcal{S}_{1b}$ coming from cluster 1. Likewise, α_2 and $\lambda_2 = \lambda_{\mathcal{S}_{2a}} + \lambda_{\mathcal{S}_{2b}}$ will denote, respectively, the EDR and the transmission rate of the botnet $\mathcal{S}_{2a} \cup \mathcal{S}_{2b}$ aggregating bots from cluster 2. In the following analysis, we set $\alpha_1 = 1$ and let α_2 vary. We consider the limiting MIRs evaluated using the theoretical formulas, and accordingly denoted without the symbol $\widehat{}$ and without time-dependence.

In Fig. 3.5, we display $\rho_{\mathcal{S}_a \cup \mathcal{S}_b}$ and ρ_{tot} as functions of α_2, for three values of the overlap degree ω_{12} (we recall that, in view of (3.24), the value of ω_{21} is uniquely determined from the values α_1, α_2 and ω_{12}), and for balanced transmission rates $\lambda_{\mathcal{S}_{1a}} = \lambda_{\mathcal{S}_{1b}} = \lambda_{\mathcal{S}_{2a}} = \lambda_{\mathcal{S}_{2b}} = 3$. We see that the inequality in (3.26) is confirmed across the whole range of α_2 and for all the considered values of overlap. Furthermore, for very small values of α_2 all the curves tend to collapse, since they correspond to the degenerate case where cluster 2 has empty emulation dictionary.

In Fig. 3.6 we repeat the analysis by using an intermediate overlap degree $\omega_{12} = 0.5$, and by considering different values for the transmission rates, corresponding to four practical scenarios. Scenario 1 (dashed-dotted curves) accounts for the case where the activity of subnet $\mathcal{S}_a$ is predominant w.r.t. the activity of $\mathcal{S}_b$. Under scenario 2 (dashed curves), the activity of bots from cluster 2 is more intense than the activity of bots from cluster 1. Scenarios 3 (dotted curves) and 4 (continuous curves) are mixed combinations of the other two scenarios.

Remarkably, in all our experiments we observed that (3.26) was verified. While we cannot claim that BotClusterBuster is consistent when (3.17) is violated and/or under the asymmetric scenario, the conducted numerical analysis explains why BotClusterBuster is in fact able to discriminate bots from normal users under various practical conditions, as we will see in detail in the section devoted to simulations.

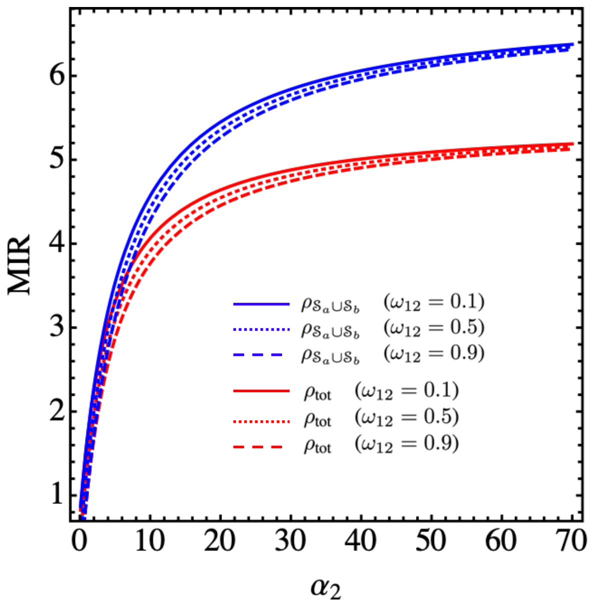


Figure 3.5: Numerical analysis aimed at testing verification of the inequality $\widehat{\rho}_{\mathcal{S}_a\cup\mathcal{S}_b} > \widehat{\rho}_{\text{tot}}$. The pertinent MIRs are displayed as functions of α_2 (with $\alpha_1 = 1$), and computed for different values of the overlap degrees.

3.5 Experimental Results

In this section we report the results of an experimental campaign that we have conducted to validate the theoretical analysis developed in the previous sections. We start with the description of the experimental setup. First of all, we constructed two different datasets containing traffic patterns from legitimate network users. In the forthcoming analysis, we will test BotClusterBuster over both datasets. The first dataset (which will be referred to as CampusDataset), was built by asking 100 people randomly polled around the Campus of the University of Salerno, to query an auction portal for about 3 minutes. The second dataset (which will be referred to as LabDataset), was built in the Co.Ri.TeL Laboratory of the University of Salerno. We asked 10 people (students and researchers) to query an e-commerce portal for about 20 min-

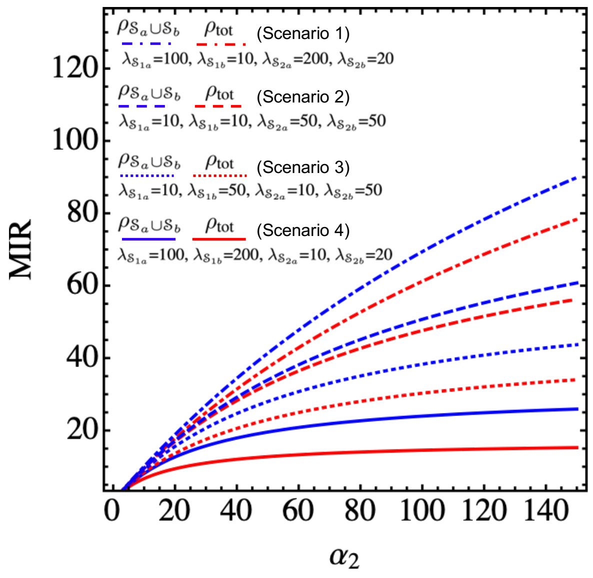


Figure 3.6: Numerical analysis aimed at testing verification of the inequality $\widehat{\rho}_{\mathcal{S}_a \cup \mathcal{S}_b} > \widehat{\rho}_{\text{tot}}$. The pertinent MIRs are displayed as functions of α_2 (with $\alpha_1 = 1$), and computed for different values of the transmission rates of the single subnetworks and clusters.

utes. With regard to this dataset, in order to obtain a larger number of legitimate users, we divided the traffic patterns into chunks lasting about 2 minutes, and considered each chunk as an independent user, obtaining a total number of 100 normal users. In both datasets, each traffic "track" is representative of a *normal* user. The application-layer patterns of these tracks have been captured by a network sniffer.

Once the normal users activity has been produced, we focus on the multi-cluster DDoS attack.[1] We will consider 3 clusters. The preliminary step is to learn an emulation dictionary that is valid for the particular target site. To this end, we join all the normal-

[1]Needless to say, we did not launch a DDoS attack against any website. We used the admissible messages taken from the activity performed by the selected real-world users to build an emulation dictionary, and then simulated the emission of requests, without effectively sending these requests to the target site.

user tracks pertaining to the particular dataset/target-site under attack (obtaining 27072 packets for both the CampusDataset and 22178 packets for the LabDataset), and consider this ensemble of messages as the overall emulation dictionary to be disseminated across the botnet. Then we divide the overall dictionary into 7 sets:

$$E_1, \quad E_2, \quad E_3, \quad E_{12}, \quad E_{13}, \quad E_{23}, \quad E_{123}, \tag{3.28}$$

where *i*) all sets are mutually disjoint; *ii*) the indices appearing as subscripts denote the cluster(s) using messages in that particular set. For example, E_1 contains messages that can be accessed only by cluster 1 and E_{123} contains messages that are available for all clusters. Then, let us focus on cluster 1 for the sake of definiteness. As initialization, we assume that at time $t = 0$ the emulation dictionary $\mathcal{E}_1(0)$ contains 100 messages chosen, e.g., from E_1. Then, at time $t > 0$ the dictionary $\mathcal{E}_1(t)$ adds to these 100 messages $\lfloor \alpha_1 t \rfloor$ messages picked from the sets relative to cluster 1, partitioned as follows:

$$\begin{aligned} \lfloor \alpha_1 \omega_{123} t \rfloor &\text{ from } E_{123}, \\ \lfloor \alpha_1 (\omega_{12} - \omega_{123}) t \rfloor &\text{ from } E_{12}, \\ \lfloor \alpha_1 (\omega_{13} - \omega_{123}) t \rfloor &\text{ from } E_{13}, \\ \lfloor \alpha_1 (1 - \omega_{12} - \omega_{13} + \omega_{123}) t \rfloor &\text{ from } E_1, \end{aligned} \tag{3.29}$$

where ω_{12} and ω_{13} are the overlap degrees introduced in (3.23), and ω_{123} is the (sub-)fraction of elements that is picked from the intersection E_{123}, common to all the three clusters.[2] The same procedure is applied to all clusters. Then it is easily verified that the dictionary construction fulfills the following conditions, for all

[2]With regard to the picking rule in (3.29), it is readily seen that not all configurations of the parameters α_i, ω_{ij} and ω_{123} are compatible with all configurations of the sets in (3.28). We have verified that the configurations used in our simulations are admissible, which roughly amounts to say that the cardinalities of the pertinent sets are sufficiently large to make the choices in (3.29) admissible for the entire duration of the observation period.

$i, j \in \{1, 2, 3\}$:

$$\lim_{t\to\infty} \frac{|\mathcal{E}_i(t)|}{t} = \alpha_i, \quad \lim_{t\to\infty} \frac{|\mathcal{E}_i(t) \cap \mathcal{E}_j(t)|}{|\mathcal{E}_i(t)|} = \omega_{ij}. \tag{3.30}$$

In order to make the bots similar to legitimate users, both the bot transmission rates and the EDR have been chosen on the same order of the corresponding attributes estimated over the normal users' traces. Finally, we will consider an overall network with 200 users, 100 of which are bots disseminated over 3 clusters of sizes 20, 30, and 50.

We are now ready to examine the performance of the BotClusterBuster algorithm. We start with the following setting:

$$\alpha_1 = \alpha_2 = \alpha_3 = 10, \tag{3.31}$$

and

$$\omega_{12} = \frac{3}{4}, \;\; \omega_{13} = \omega_{23} = \frac{1}{2}. \tag{3.32}$$

For the sake of definiteness, we report the analysis performed on the CampusDataset, with similar conclusion being obtained for the LabDataset. In Figs. 3.7 and 3.8, we illustrate the results corresponding to decreasing values of the threshold parameter θ ($\theta = 0.95$ in Fig. 3.7, $\theta \in \{0.9, 0.75, 0.5\}$ from left to right in 3.8). Let us start by examining the bottom panels, where we display an algorithm's snapshot corresponding the end of the observation window. The decisions produced by the algorithm are encoded in a graphical matrix, whose p-th row corresponds to the output of the algorithm when user p is elected as a pivot (users are ordered for clarity of visualization, but this ordering is immaterial, since in our simulations the bots were randomly spread over the network). A white pixel represents the "estimated bot presence", whereas a black pixel represents the "estimated bot absence". Thus, if pixel (p, q) is white, the algorithm is estimating (before applying the particular cluster selection rule) that user q is a bot when user p is elected as pivot.

For $\theta = 0.95$ (Fig. 3.7), condition (3.17) is met for the largest

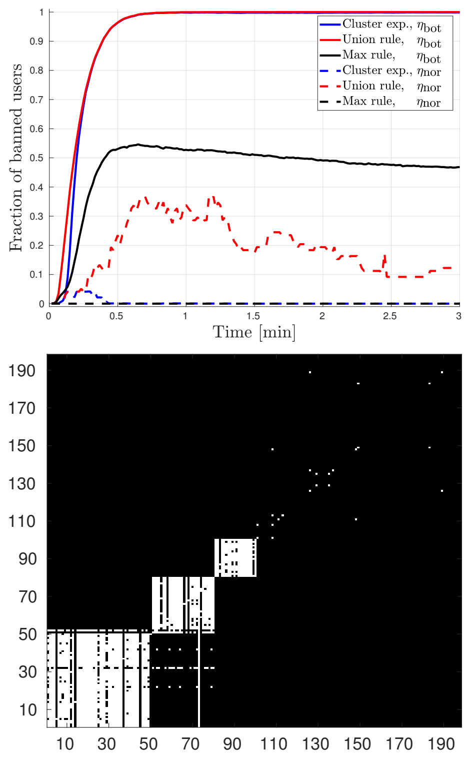


Figure 3.7: *Top.* Performance of the botnet identification algorithm. *Bottom.* Algorithm's snapshot corresponding the end of the observation window: the decisions produced by the algorithm are encoded in a graphical matrix whose p-th row corresponds to the output of the algorithm when user p is elected as a pivot. A white pixel in the location (p, q) signifies that the algorithm is estimating (before applying the particular cluster selection rule) that user q is a bot when user p is elected as pivot.

degree of overlap, i.e., $\omega_{12} = 0.75$. The presence of three distinct off-diagonal white squares (of sides 20, 30, and 50) means that the algorithm is able to identify the 3 clusters properly in agreement with the theoretical prediction. We notice also the occasional presence of some white pixels on the main diagonal. These pixels correspond to very small clusters (about 2 or 3 nodes) composed of normal users. Notably, the small size of these clusters reveals that the algorithm can occasionally form a wrong cluster, but that then this cluster is never paired with other users, i.e., it is very unlikely that more than a few users can form a wrong botnet. This property is very important, since clusters with so small cardinalities cannot have any significance in practice, and there are many rules one can conceive to discard them. In particular, we will now see whether the proposed cluster expurgation rule is able to get rid of this effect. Moreover, we notice that the white squares corresponding to the clusters have some black gaps, and are accompanied by some horizontal white bands. For what concerns gaps, a black pixel in position (p, q) signifies that the algorithm is not estimating q as member of the botnet when p is pivot. However, BotClusterBuster is on purpose designed so as to perform redundant checks (i.e., the same pairs of nodes are involved in multiple checks). In this way, a bot is missed only if it is missed by *all* checks. For this reason, we see that the occasional black gaps observed in the white squares are irrelevant if at least one time each bot is included in a cluster. In the end, the overall set of nodes that belong at least to one cluster can be found on the main (white) diagonal of the graphical matrix. In relation to the horizontal white bands, a collection of white pixels in row p means that the pivot p results paired with many nodes q, which results into the formation of a relatively large cluster. Accordingly, we see that occasionally some bots of one cluster appear paired with bots of another cluster. This effect is clearly not relevant for DDoS mitigation, as can be concluded from the fact that $\eta_{\text{bot}}(t) \approx 1$ (top panel).

For $\theta = 0.9$ (left panel in Fig. 3.8), condition (3.17) is still met, however, we are approaching the case $\theta = 0.75$ which would result into a violation of (3.17). Accordingly, we see that the images of the individual clusters starts to become a bit blurred. When we set $\theta = 0.75$ (center panel), condition (3.17) is violated, and, as predicted from our analysis in Sec. 3.4.1, the algorithm does not lose the capability of identifying the botnet, but looses

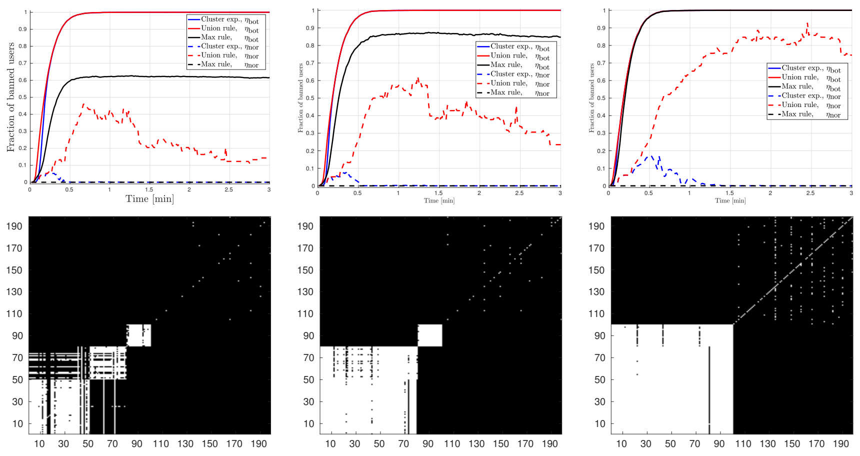


Figure 3.8: *Top.* Performance of the botnet identification algorithm. *Bottom.* Algorithm's snapshot corresponding the end of the observation window: the decisions produced by the algorithm are encoded in a graphical matrix whose p-th row corresponds to the output of the algorithm when user p is elected as a pivot. A white pixel in the location (p, q) signifies that the algorithm is estimating (before applying the particular cluster selection rule) that user q is a bot when user p is elected as pivot. We consider different values of the threshold parameter, in particular, from left to right we have $\theta = 0.9, 0.75, 0.5$. All experiments refer to the CampusDataset.

the capability of identifying the individual clusters. In particular, since in our example we have $\theta = \omega_{12} = 0.75$, but we have also $\theta > \omega_{13} = \omega_{23} = 0.5$, we see that the first two clusters are merged into a big one, whereas the third cluster is still correctly identified. Finally, this capability is lost when $\theta = 0.5$ (right panel), which corresponds to the estimation of a unique big botnet comprising all 100 bots. Moreover, in the case $\theta = 0.5$ we notice the emergence of some vertical white bands. A band in column q means that there exists a node q that results paired with many pivots p.

This corresponds to the formation of many (one for each paired pivot) small clusters. Accordingly, we see that some spurious small clusters made of normal users arise. As we will see soon, this effect will play a role in the performance of the union-rule.

Let us move on to examine the top panels of Fig. 3.7 and 3.8, where we display the performance of the algorithm expressed in terms of the indicators $\eta_{\text{bot}}(t)$ and $\eta_{\text{nor}}(t)$, which have been evaluated over 100 Monte Carlo runs. The DDoS evolution is monitored during the observation window with a time-step of 0.1 minutes. We show the results corresponding to the three rules introduced in the previous sections, namely, the max-rule, the union-rule and the cluster expurgation rule.

Let us consider the first case $\theta = 0.95$ (Fig. 3.7), which meets well condition (3.17). We start with the max-rule. From the evolution of $\eta_{\text{nor}}(t)$ (black dashed curve), we see that the max-rule is effective in rejecting the fake micro-clusters. This happens since the max-rule retains only the maximum-cardinality cluster, which has however a detrimental effect as regards the bot identification. Indeed, we see that $\eta_{\text{bot}}(t)$ (black solid curve) converges approximately to 0.5, which means that only the biggest cluster made of 50 bots is correctly identified.

We continue by illustrating the behavior of the union-rule. Here we see that the bot identification issue is remediated, since $\eta_{\text{bot}}(t)$ (red solid curve) converges to 1. This happens because the union-rule aggregates the detected clusters of *any cardinality*. On the other hand, this effect is detrimental as regards the normal users classification. Indeed, we see that $\eta_{\text{nor}}(t)$ (red dashed curve) converges approximately to 0.1. However, this effect must be expected, especially in light of the analysis of the bottom panels in Fig. 3.7. The union-rule is in fact aggregating many clusters of very small cardinality. This means that, under the union-rule, we are claiming that there exist a DDoS attack launched by several botnets comprised of 2 or 3 members, which appears to be a very uncommon scenario.

Another notable effect observed in the time-evolution of $\eta_{\text{nor}}(t)$ is the appearance of a peak at some intermediate time. The explanation of this peak sheds some further light on the detrimental effects produced by the union-rule. Examining the individual traffic patterns, we identified one specific normal user that was performing some anomalous activity. In particular, they were performing some very intense activity for a single slot during the entire observation window. As a result, during this slot of intense activity, this user was coupled to a very large number of other normal users, giving rise to several micro-clusters of size 2. The union-rule was then including *all* these clusters into the estimated botnet, producing the peak observed in $\eta_{\text{nor}}(t)$. This effect disappears progressively as time elapses, since the abnormal behavior of the particular user is not persistent over time.

Let us finally see whether the cluster expurgation rule is able to solve both issues experienced under the max-rule and the union-rule. The performance of BotClusterBuster is accordingly displayed in blue. We see that this algorithm has very good performance, since $\eta_{\text{bot}}(t)$ (blue solid curve) converges approximately to 1, and $\eta_{\text{nor}}(t)$ (blue dashed curve) converges approximately to 0. With regard to the hyperparameters of the cluster expurgation rule, in the experiments we set $\kappa = 1$ and $\xi = 0.1$. We remark that these are very "agnostic" choices. Indeed, the choice $\kappa = 1$ corresponds to say that a DDoS with attack rate barely equal to a legitimate traffic rate is deemed as dangerous. Likewise, the choice $\xi = 0.1$ corresponds to say that a cluster is deemed as meaningful if it sustains barely the 10% of the botnet activity.

From the analysis of the top plot in Fig. 3.7, we conclude that BotClusterBuster delivers good performance under the conditions used to prove our theoretical results. Let us now see how a reduction of the threshold parameter θ influences the performance. To this end, we examine the top plots in Fig. 3.8. For what concerns the max-rule, we see that $\eta_{\text{nor}}(t)$ (black dashed curve) converges approximately to 0 in all cases. In other words, the performance in terms of normal users is not affected by variations of θ. In comparison, $\eta_{\text{bot}}(t)$ (black dashed curve) is sensitive to θ. As it should be expected, when the algorithm aggregates two clusters into a big cluster of size 80 (center panel in Fig. 3.8), then $\eta_{\text{bot}}(t)$ converges approximately to 0.8, whereas when a unique cluster is produced (right panel in Fig. 3.8), then $\eta_{\text{bot}}(t)$ converges approximately to 1.

For what concerns the union-rule, we see that $\eta_{\text{bot}}(t)$ (red solid curve) converges approximately to 1, irrespectively of the value of θ, which is expected since the union rule aggregates all detected clusters and, hence, is not sensitive to the individual clusters. On the other hand, the union-rule performance degrades severely when we decrease the value of θ, since lower values of θ favor the emergence of fake micro-clusters. Accordingly, we see that $\eta_{\text{nor}}(t)$ (red dashed curve) increases (i.e., the performance degrades) as θ decreases.

So far we have shown that BotClusterBuster provides proper cluster identification when the technical conditions used to prove the theoretical results are met, and continues to provide proper botnet identification even when condition (3.17) is violated. Now we want to check what happens when we deviate further from the nominal conditions by allowing for clusters' asymmetries, as described in Sec. 3.4.2. Specifically, we set:

$$\alpha_1 = \alpha_2 = 10, \quad \alpha_3 = 30, \tag{3.33}$$

and

$$\omega_{12} = \frac{3}{4}, \quad \omega_{13} = \omega_{23} = \frac{1}{2}, \tag{3.34}$$

with the companion overlap degrees ω_{21}, ω_{31} and ω_{32} being determined by the constraints (3.24). The results of these experiments are reported in Fig. 3.9, with reference to the case $\theta = 0.95$, and for the two datasets, namely, CampusDataset (left) and LabDataset (right). As predicted by the numerical analysis carried out in Sec. 3.4.3, asymmetries in the EDRs do not lead to significant variations in the botnet identification performance. In summary, we reach the remarkable conclusion that both the asymmetries and the violation of (3.17) do not impair consistent botnet identification.

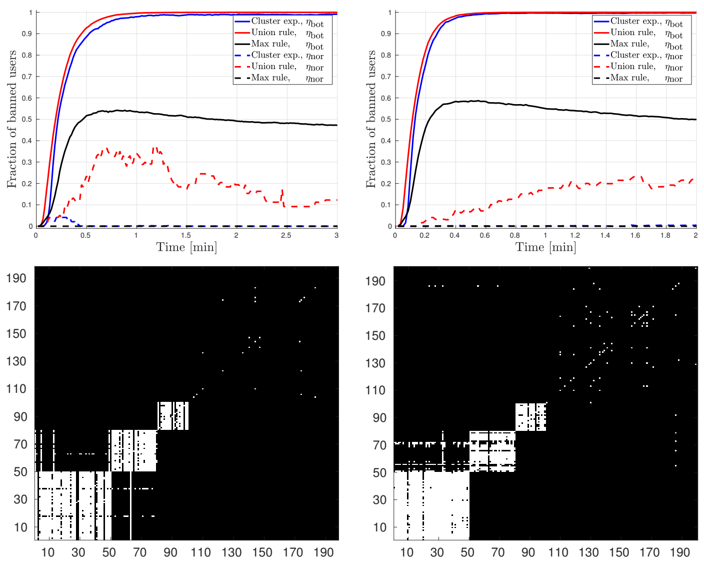


Figure 3.9: *Top.* Performance of the botnet identification algorithms. *Bottom.* Algorithm's snapshot corresponding the end of the observation window: the decisions produced by the algorithm are encoded in a graphical matrix whose p-th row corresponds to the output of the algorithm when user p is elected as a pivot. A white pixel in the location (p, q) signifies that the algorithm is estimating (before applying the particular cluster selection rule) that user q is a bot when user p is elected as pivot. Left panels refer to the CampusDataset, whereas right panels to the LabDataset.

Chapter 4

High Availability of Telecommunication Networks

In this chapter we face the general problem of network availability, with special focus on Service Function Chains (SFCs). They represent a modern way of composing network services within software-based environments, where the virtualized nodes are traversed in a specific order to provide a given service. As valuable case of SFCs, we consider the IP Multimedia Subsystem (IMS), an infrastructure aimed at managing multimedia content within 4G/5G telecommunication systems.

It is useful to split this contribution in three parts. In the first part (Section 4.1), we detail the performability (performance and availability) model of a virtualized IMS deployment, where we: *i*) exploit the SRN and RBD formalisms to derive a state-based availability model, where state transitions are supposed to be exponentially distributed; *ii*) perform a comparative analysis of multiple IMS schemes in order to evaluate the fault robustness also through a sensitivity assessment; *iii*) devise an algorithm (nicknamed OptChains+) in charge of, first, evaluating RBD/SRN models associated to IMS deployments (or settings), and, then, pinpointing the settings which satisfy, at the same time, minimum cost and high availability under specific performance criteria.

In the second part (Section 4.2) we consider a more realistic scenario where we remove the exponential assumption on state transitions. In particular, we assume that repair transition times follow a Weibull distribution, enabling us to highlight some interesting effects of the transient availability analysis.

In the final part (Section 4.3), we describe a designed-from-scratch tool that allows to automate the generation of availability models for service chains. Such a tool is developed by adhering to the Model-Driven Engineering (MDE) paradigm, and is conceived to support the network managers during the modeling availability stage.

4.1 Performability of Service Chains

The SFCs paradigm allows to easily create and deploy novel services through a series (namely, a chain) of concatenated network components [66, 67]. They are often designed by exploiting *softwarized* technologies such as virtualization, microservices, *containerized* environments, which provide a flexible and handy habitat for diverse telecommunication frameworks [68, 69]. Among such frameworks we focus on the IP Multimedia Subsystem (IMS), which, embracing the service chaining concept, has been elected both by standardization groups [70] and by the industry world [71, 72] as the ideal intermediary between legacy networks and 5G-based solutions. In this regard, being mainly focused on an architectural problem, we consider a high-level perspective of the IMS service chain, as often contemplated in the technical literature on SFCs infrastructures (e.g., [73, 74]). Moreover, a recent project named Clearwater [2] has been specifically conceived to offer an open source implementation of IMS architecture within virtualized and containerized environments, and to be exploited as a benchmark testbed for assorted performance analyses [75–77].

Inspired by this last trend, we draw up a technique for managing availability and performance of service chains, where a containerized version of IMS (cIMS for brevity) has been considered as pivotal use case. Basically, a container is a lightweight process [78, 79] that, differently from classic virtual machines (VMs), does not include a whole operating system. Container technology is normally coupled with the Network Function Virtualization (NFV) paradigm which allows to encompass the network logic (e.g. routing, firewalling, load balancing) in virtualized elements referred to as Virtual Network Functions (VNFs).

In the specific case considered in this work, containers fulfill functionalities encountered in an IMS domain. Moreover, since failure and repair events can occur at various layers of the cIMS system (container, VM, hypervisor, etc.), we first model the probabilistic behavior of the whole cIMS chain, and then solve an optimization problem to achieve cIMS configurations guaranteeing, simultaneously, high availability and minimal costs at a given performance level.

More generally, the methodology can be adapted to other instances in the field of telecommunications and hence can be useful to management organizations involved in planning/deploying/maintaining systems and services, where trade-offs among cost, availability, and performance are critical.

4.1.1 Service Availability

Service availability is a crucial parameter of Quality of Service (QoS) as specified by ITU-T E.800 [80]. The severe requirements prescribed by this standard imply a very careful network design of 5G infrastructures, where virtualized and containerized modules interact not only among them, but also with the underlying hardware infrastructure. For instance, more containers deployed on a single virtual machine can be adversely affected by a malfunctioning operating system running on the VM, or, similarly, more virtualized network functions sharing the same physical layer can be adversely influenced by a misconfiguration of the resource isolation mechanism [81]. Further issues can arise when the network elements have to be traversed in a specific order to provide a service. It is the case of SFCs, of which IMS can be considered a particular realization. Other examples may include: virtualized Evolved Packet Core (vEPC) solutions able to interoperate in a chained fashion with SDN components [82], service chains deployments across virtual data centers [83], virtual Mobility Management Elements (vMMEs) organized as SFCs interested by signaling processing flows [84].

Across such a chained scheme, the availability analysis must consider the features of each single node which, if affected by a failure event, disrupts the whole chain network flow.

Accordingly, we provide the following main contributions: *i)* we propose a detailed availability characterization of virtualized network chains by exploiting two techniques: Reliability Block Diagrams (RBDs) to describe the high-level interconnections among concatenated nodes and Stochastic Reward Networks (SRNs) to model the internal structure of each node from a probabilistic point of view; *ii)* we conduct a performability (availability plus performance) assessment of an exemplary virtualized chain constituted by an IP Multimedia Subsystem deployed in a container-based setting, namely, a containerized IMS (cIMS) where three containerized schemes have been compared and discussed in detail; moreover, through a sensitivity analysis, we evaluate the robustness of cIMS with respect to deviations of some external parameters from their nominal values, due, e.g., to designer's uncertainty on failure/repair mean times; *iii)* we devise an algorithm (nicknamed OptChains+) in charge of evaluating first RBD/SRN models associated to cIMS deployments (or settings) and then pinpointing the settings which satisfy, at the same time, minimum cost and high availability under specific performance criteria. Finally, an experimental testbed based on the Clearwater project has been deployed to execute stress tests aimed at deriving the relevant workload parameters.

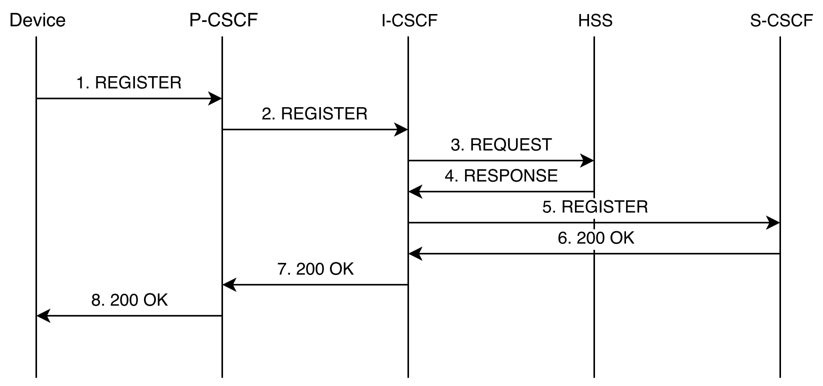


Figure 4.1: A simplified Registration procedure in the IMS domain. The *Register* message is propagated from a device to the S-CSCF. A 200 *OK* message is back-propagated to the device if the procedure ends correctly.

4.1.2 Overview of the IMS Architecture

The IP Multimedia Subsystem was born as a framework able to providing access to a plethora of multimedia IP-based services with guaranteed quality of service [85]. The IMS architecture supports a broad range of services by exploiting the flexibility of the Session Initiation Protocol (SIP) [86], to mention a few, multimedia and real-time sessions (such as phone calls), web messaging and enriched communications. The signaling flows are managed by the Call Session Control Function (CSCF) servers, which communicate by exchanging mainly SIP messages. The CSCF functionalities are distributed among three servers. The *Proxy* CSCF (P-CSCF) is a SIP proxy, and acts as an interface between the user device and the IMS network. The *Interrogating* CSCF (I-CSCF) forwards SIP requests or responses within the domain. The *Serving* CSCF (S-CSCF) is in charge of performing some core functions as session and routing control and user registration management. Another key element of the IMS infrastructure is the Home Subscriber Server (HSS), an advanced database containing users' profiles that can be queried by means of a specific protocol called Diameter.

Such nodes are interconnected to provide basic and advanced services. An example is offered in Fig. 4.1, where a (simplified) registration procedure (needed before exploiting IMS services) is depicted. Initially, a user device contacts P-CSCF via a register message (1). Such a message is passed to I-CSCF (2) that, in turns, sends it to HSS (3) in order to retrieve the address of S-CSCF in charge of current registration. Once obtained the information from HSS (4), the message is forwarded to the correct S-CSCF (5). Finally, an OK message indicating a correct device registration is back-propagated to the user device (6), (7), (8). Once completed the registration procedure, the device is ready to use IMS services as, e.g, a real-time audio/video session. It is worth noting that, in Fig. 4.1 the signaling flow is shown, but typically media flows between distinct user devices (i.e. the content of an audio/video call) traverse distinct paths.

4.1.3 Containerized IMS Infrastructure

We consider a deployment of an IMS infrastructure (described in the previous section) in a container-based framework such as the recently introduced Docker [87], RKT [88] or OpenVZ [89] frameworks.

With respect to VMs, containers exhibit some differences. First, containers share the host operating system, whereas a VM has its own guest operating system, resulting in a heavier structure (in terms of disk/memory utilization, start-up time, etc.). Second, a VM exhibits a strong isolation at the host kernel level (being the operating system not shared), thus exhibiting a stronger security w.r.t. their container counterpart. Finally, containers are more flexible in terms of portability (since they do not have a separate operating system), whereas VMs require additional efforts during porting operations especially when the hosting platforms are different.

Market leaders such as Amazon Web Services and Google Container Engine typically take advantage both from virtualization and containerization by designing infrastructures where containers run on top of VMs [90]. Actually, the possibilities of combining VMs with containers strongly depend on the specific policy adopted by the cloud provider. An useful taxonomy is provided in [62], where two common schemes stand out: the first is the *homogeneous* one, where several instances of a single kind of containers run on top of one and the same VM. Such a scheme is well suited for public cloud environments where, for security issues, it is preferable not to share VMs among different users or tenants. The second one is the *heterogeneous* scheme, where different kinds of containers are allowed to share the same VM. It is the case of private cloud environments (see for example [91]), where there are no strict security requirements, and where it is possible to design scalable redundant policies (by replicating the entire VM) to cope with failure events. Specifically, we consider Docker-based implementations of both schemes that share a common five-layer arrangement referred to as a Containerized Network Replica (CNR), consisting of (see Fig. 4.2): *i)* an infrastructure layer that generically embodies hardware components (HW) such as CPU, RAM, power supplies; *ii)* a hypervisor (HPV) acting as an interface between hardware and upper layers; *iii)* a virtual machine (VM) layer that provides a wrapper for the Docker environment; *iv)* a Docker daemon (DCK) that offers a runtime environment to handle containers; *v)* a Container (CNT) which embeds the specific software functionality to be provided (e.g. Proxy, Serving, etc.), and represents the basic element supplying/managing IMS sessions. Since a DCK can handle more than one CNT, and since a HPV can handle different VMs, different CNRs schemes are possible. As illustrated in Fig. 4.2, scheme (a) accounts for a homogeneous implementation of a CNR hosting only one kind of instance; scheme (b) represents a co-located homogeneous implementation where different containers instances are deployed over different dockers and virtual machines, although they share the same underlying infrastructure; scheme (c) accounts for a heterogeneous implementation that allows to consider a co-located deployment of different containers; finally, scheme (d) represents a mixed heterogeneous deployment with different dockers and virtual machines. We remark that schemes (b) and (c) represent

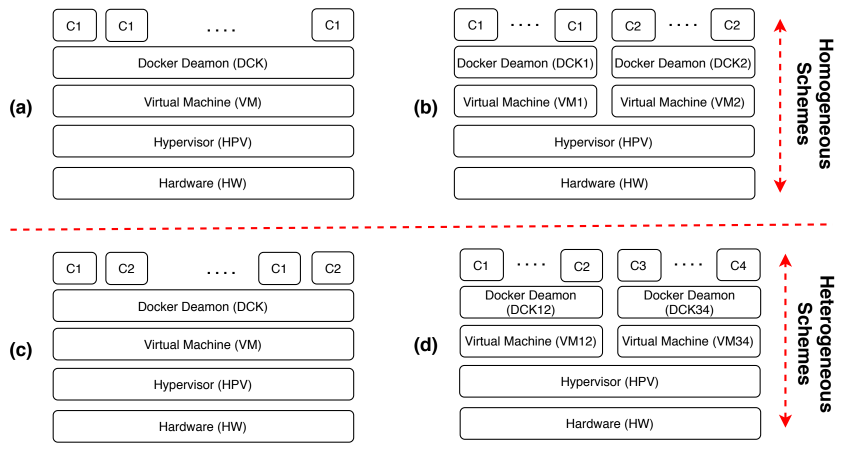


Figure 4.2: Different schemes of CNRs. Schemes (a) and (b) realize the pure and the co-located homogeneous deployments, respectively. Schemes (c) and (d) realize the co-located and the mixed heterogeneous deployments, respectively. Note that the co-location in (b) is intended at an infrastructure level, whereas in (c) is intended at a container level.

different kinds of *co-location* (that in real IMS settings typically involves HSS and I-CSCF - see [92]): the former implements co-location at infrastructure level, whereas the latter implements co-location at container level. It is useful to define a Containerized Network Function (CNF) as an ensemble of CNRs deployed to provide a specific IMS functionality. Otherwise stated, a CNF is a logical abstraction of a cIMS node composed by one or more CNRs. Thus, in the sequel, the terms CNF and cIMS node may be used interchangeably.

The introduction of the CNF representation provides some degrees of freedom. First, a single CNF can be realized by means of multiple CNRs that, in principle, can be distributed geographically. Second, different CNRs belonging to the same CNF can have a different number of containers since a CNR has a limited resource to support up to a certain number of cIMS sessions. Finally, different CNRs belonging to the same CNF can be deployed according to different schemes (see Fig. 4.2). To clarify this con-

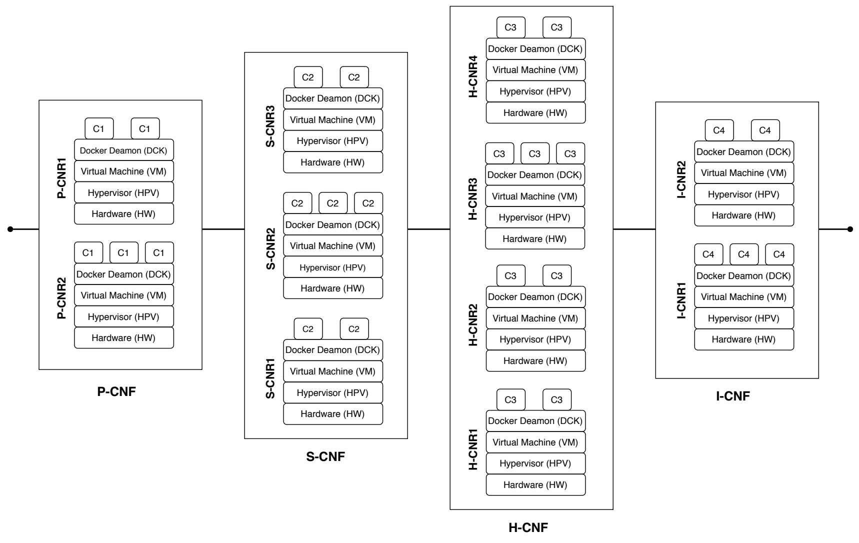


Figure 4.3: Interconnections among nodes in a (containerized) IMS infrastructure (homogeneous deployment case).

cept, Fig. 4.3 shows different CNFs (distinguished by node) each one can contain one or more CNRs. The particular arrangement in Fig. 4.3, where the four CNFs are chained in series, can be modeled using the RBD formalism.

4.1.4 Availability Model of Containerized IMS

The SRN methodology, stemming from Markov Reward Models [64], [93], allows to describe the interactions occurring among the various layers of a CNR that composes a generic cIMS node. More specifically, we adopt the usual graphical description in terms of bi-partite directed graphs where places (depicted as circles) account for specific conditions (e.g. nodes up/down), and transitions (depicted as rectangles) represent the actions (e.g. a node fails or is repaired). Inside a place, *tokens* (represented by dots or numbers) characterize holding conditions. In case of a CNT layer, one (or more) tokens lying in the "up" place indicate one or more working containers, whereas for the remaining layers (DCK, VM, HPV, HW) one token in the "up" place indicates a working layer. When a failure/repair event related to a specific CNR layer occurs (namely, a transition is *fired*), a token (or more than one token in the case of a CNT layer) is transferred from the source place to a destination place.

In an SRN, transitions times are supposed to be exponential random variables (a common assumption in reliability and availability analyses), with λ denoting the failure rate, and μ the repair rate. Solving an SRN amounts to evaluate the *reward function*, defined as a non-negative random process associated to some dependability metrics (among them, the availability).

Let $Y(t)$ be the reward function that is equal to 1 when the system is working at time t, and to 0 otherwise. The instantaneous availability can be expressed as [64]

$$A(t) = \mathbb{P}\{Y(t) = 1\} = \mathbb{E}[Y(t)] = \sum_{i \in \mathbb{I}} r_i \, p_i(t), \tag{4.1}$$

where $\mathbb{I}$ is the set of markings, namely, the set of feasible tokens distributions, r_i (commonly referred to as reward rate) is the value of $Y(t)$ in marking i, and $p_i(t)$ is the corresponding probability. The set $\mathbb{I}$ can be split in a subset of "up" states ($r_i = 1$), and a subset of "down" states ($r_i = 0$).

4.1.5 Homogeneous Scheme

Figure 4.4 (on the left) describes the SRN model of the homogeneous scheme depicted in Fig. 4.2(a) implementing a generic CNR. Places P_{upCNT} $[P_{dnCNT}]$, P_{upDCK} $[P_{dnDCK}]$, P_{upVM} $[P_{dnVM}]$, P_{upHPV} $[P_{dnHPV}]$, and P_{upHW} $[P_{dnHW}]$ take into account the working [failure] conditions of container instance, docker daemon, virtual machine, hypervisor, and hardware, respectively. Note that each place contains only one token (indicated by number 1), except for the place P_{upCNT} that contains n_k tokens, which denotes the possibility of having more replicas of a single container instance. Transitions T_{fCNT} $[T_{rCNT}]$, T_{fDCK} $[T_{rDCK}]$, T_{fVM} $[T_{rVM}]$, T_{fHPV} $[T_{rHPV}]$, and T_{fHW} $[T_{rHW}]$ denote failure [repair] activi-

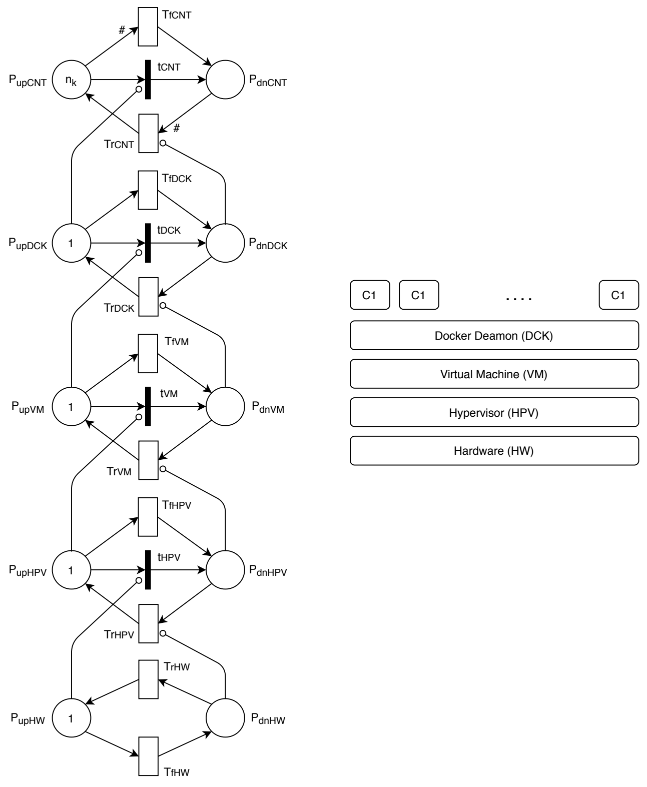


Figure 4.4: SRN-based model (on the left) representative of a generic CNR deployed according to the homogeneous scheme (on the right).

ties characterizing containers, docker, virtual machine, hypervisor, and hardware, respectively. Such transitions (depicted as unfilled rectangles) are called "timed" transitions and, as previously said, are characterized by exponentially distributed times. If a timed transition is "marking-dependent" (in Fig. 4.4 we insert the "#" symbol to denote such condition), its effective rate is multiplied by the number of tokens available in the pertinent place. Conversely, transitions t_{CNT}, t_{DCK}, t_{VM}, and t_{HPV} (represented by filled and thin rectangles) are called "immediate" transitions and account for actions occurring in a zero-length time interval.

The time-evolution of the SRN in Fig. 4.4 can be analyzed by starting from the initial working condition, where n_k tokens are located in place P_{upCNT}, while a single token is present in all the remaining "up" places. When a single container failure occurs (e.g., an uncontrolled reboot of a container instance), transition T_{fCNT} is fired and one token in P_{upCNT} is moved to place P_{dnCNT}. As a consequence, $n_k - 1$ tokens remain in P_{upCNT}. Conversely, once the container becomes again repaired, T_{rCNT} is fired and the token comes back to P_{upCNT}.

Now, let us consider the case of a docker layer failure. The transition T_{fDCK} is fired and the token is moved from P_{upDCK} to P_{dnDCK}. Notice that, when the docker layer fails, all container instances that need the underlying docker layer to be up and running become inactive. Such an issue is taken into account through an inhibitory arc (depicted as a segment between P_{upDCK} and t_{CNT} with a little circle close to the latter) that, in case of a docker failure, forces t_{CNT} to be fired. When the docker gets repaired, T_{rDCK} is fired, and two actions occur: first, the token passes from P_{dnDCK} to P_{upDCK}, then the inhibitory arc between P_{dnDCK} and T_{rCNT} is disabled and, consequently, the n_k tokens are ready to be transferred from P_{dnCNT} to P_{upCNT}.

Similar behaviors occur in case of virtual machine, hypervisor, and hardware failures/repairs. It is worth noting that the only layer without an immediate transition is the hardware layer. Indeed, being hardware the lower layer of a generic cIMS node structure, no further underlying dependencies have to be taken into account. Let us now define two quantities useful for the forthcoming performability analysis: the *demand* W, that is the required system performance in terms of concurrent IMS sessions; the *capacity* c_j, namely the maximum number of concurrent IMS sessions a container belonging to CNF j can manage. Therefore, the reward rate in marking i is

$$r_i(j) = \begin{cases} 1 & \text{if} \quad \sum_{h=1}^{k} \#P_{upCNT}^{(h)}\, c_j \geq W \text{ in marking } i, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

where h is the number of CNRs forming CNF j, and "#" refers to the number of tokens[1]. By defining $\gamma_j = W/c_j$ as the normalized performance level, Eq. (4.2) can be recast as

$$r_i(j) = \begin{cases} 1 & \text{if} \quad \sum_{h=1}^{k} \#P_{upCNT}^{(h)} \geq \gamma_j \text{ in marking } i, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

Finally, in the limit as $t \to \infty$, we get the *steady-state availability* for cIMS node j:

$$A_j = \lim_{t \to +\infty} A_j(t) = \sum_{i \in \mathbb{I}} r_i(j)\, p_{ij}, \tag{4.4}$$

where: $r_i(j)$ is derived from (4.3), and p_{ij} is the steady-state probability given by $p_{ij} = \lim_{t \to +\infty} p_{ij}(t)$ (where $p_{ij}(t)$ is the instantaneous probability of the CNF j being in marking i). It is worth noting that in (4.3) we are implicitly considering k independent CNRs replicas represented in turn by k SRNs. Accordingly, the overall marking state-space can be seen as the cartesian product of the marking state-spaces pertaining to the k composing SRNs. Thus, the probability in (4.4) is meant as the product of probabilities pertaining to the different markings of the k SRNs. Such a remark also holds true for the cases presented in the following sections (co-located, heterogeneous, mixed).

From the CNF availability in (4.4), it is possible to derive the overall steady-state availability for the homogeneous scheme as

$$A_{cIMS} = \prod_{j \in \{P,S,I,H\}} A_j. \tag{4.5}$$

The product in (4.5) stems from the RBD-like modeling of Fig. 4.3 representative of series connection among cIMS nodes. The overall cIMS steady-state availability, in fact, requires that each node must be available.

[1]In the standard SRN terminology there is a little abuse of notation, as the symbol # denotes both the number of tokens and the marking-dependent transitions in SRN graphical representations.

4.1.6 Homogeneous Co-located Scheme

Let us now consider the availability model of a CNR deployed according to a homogeneous co-located scheme as shown in Fig. 4.2(b). We recall that such a co-location is realized by sharing the infrastructural level (hypervisor/hardware). The corresponding SRN is shown in Fig. 4.5 (on the left), where, for the sake of simplicity, we consider only two different co-located containers and two different dockers and virtual machine layers. Comparing this co-located scheme against the homogeneous scheme in Fig. 4.4, we can observe that the main structure remains unaltered (on the left), whereas a new piece (on the top right) typifies the presence of co-located elements. In particular, the new piece present on the top right of Fig. 4.5 embodies: the second container layer (C2), the second docker layer (DCK2), and the second virtual machine layer (VM2). Moreover, we can notice the presence of two further inhibitory arcs connecting the two parts of the graph. The first one between P_{upHPV} and t_{VM2} accounts for the fact that, if hypervisor fails, the co-located VM (and, in turn, co-located DCK and CNT) cannot be operative, thus, t_{VM2} is fired and the only token in P_{upVM2} is moved to P_{dnVM2}. The second inhibitory arc between P_{dnHPV} and T_{rVM2} accounts for the fact that the token cannot be moved from P_{dnVM2} to P_{upVM2}, until the token in P_{dnHPV} is transferred to P_{upHPV} , since the co-located virtual machine cannot be restored until hypervisor gets repaired.

Similarly, given marking i, it is possible to define a new reward rate as

$$r'_i(j_1, j_2) = \begin{cases} 1 & \text{if} \quad \left(\sum_{h=1}^{k} \#P^{(h)}_{upCNT1} \geq \gamma_{j1}\right) \wedge \\ & \quad\quad \left(\sum_{h=1}^{k} \#P^{(h)}_{upCNT2} \geq \gamma_{j2}\right) \text{ in marking } i, \\ 0 & \text{otherwise,} \end{cases} \tag{4.6}$$

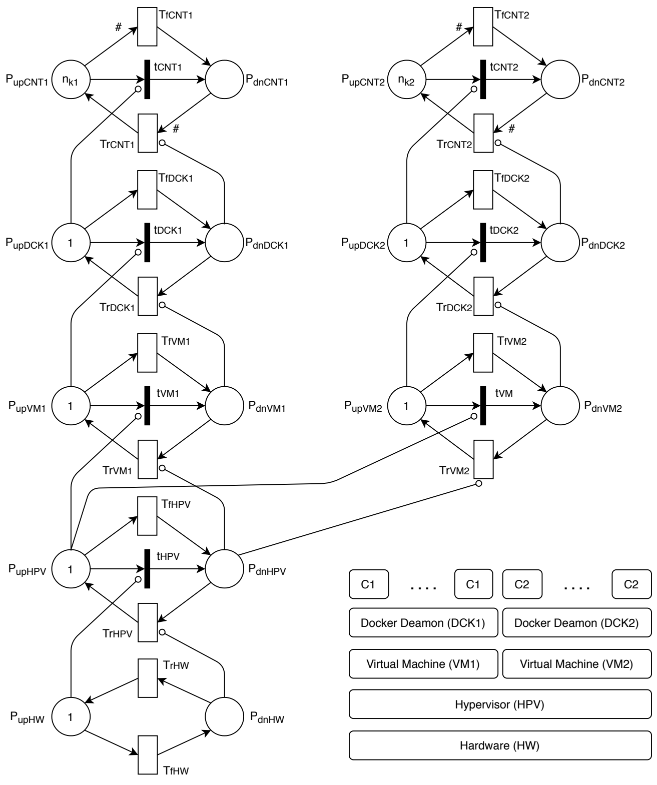


Figure 4.5: SRN-based model (on the left) representative of a generic CNR deployed according to the homogeneous co-located scheme (on the bottom-right).

where the symbol $\wedge$ denotes a logical AND operator between the two conditions. The above expression can be interpreted as a generalization of (4.3) to the case of two co-located types of containers with performance capacities c_{j1} and c_{j2} belonging to co-located CNFs j_1 and j_2 that, in our case, are I-CSCF and HSS.

Accordingly, the steady-state availability pertaining to a pair of co-located nodes can be expressed as

$$A'_{j_1 j_2} = \lim_{t \to +\infty} A'_{j_1 j_2}(t) = \sum_{i \in \mathbb{I}} r'_i(j_1, j_2)\, p'_{i j_1 j_2}, \qquad (4.7)$$

where $r'_i(j_1, j_2)$ is given by (4.6), and $p'_{ij_1j_2}$ is the corresponding steady-state probability. Considering that, in a homogeneous co-located scheme, the overall cIMS infrastructure is composed by two non-colocated nodes (typically P-CSCF and S-CSCF) and two co-located nodes (typically I-CSCF and HSS), the overall steady-state availability is:

$$A'_{cIMS} = A'_{j_1j_2} \prod_{j \neq j_1, j_2} A_j. \tag{4.8}$$

The product of A_j (derived from (4.4)) spans across the two non-colocated nodes, whereas $A'_{j_1j_2}$ (from (4.7)) takes into account the remaining co-located nodes.

4.1.7 Heterogeneous Co-located Scheme

The next model refers to a heterogeneous co-located scheme, whose pertaining SRN is depicted in Fig. 4.6 (on the left). Such a scheme represents a lightweight co-location since the whole infrastructure from docker to hardware can host different kinds of containers. In this case, just one new part represented by another container has been introduced. Similar to the previous case, the inhibitory arc from $P_{upDCK12}$ and t_{CNT2} forces the co-located container to fail in case of a docker failure, whereas the inhibitory arc from $P_{dnDCK12}$ to T_{rCNT2} prevents that co-located containers could be working again until docker gets repaired.

Since the definition of the work condition is the same as in the co-located scheme defined in the previous section, the reward rate and the steady state availability are the same defined in 4.6, 4.7 and 4.8.

4.1.8 Heterogeneous Mixed Scheme

This last SRN model pertains to a heterogeneous mixed case that is a combination of the homogeneous co-located and the heterogeneous co-located schemes; referring to Fig. 4.7 (on the left).

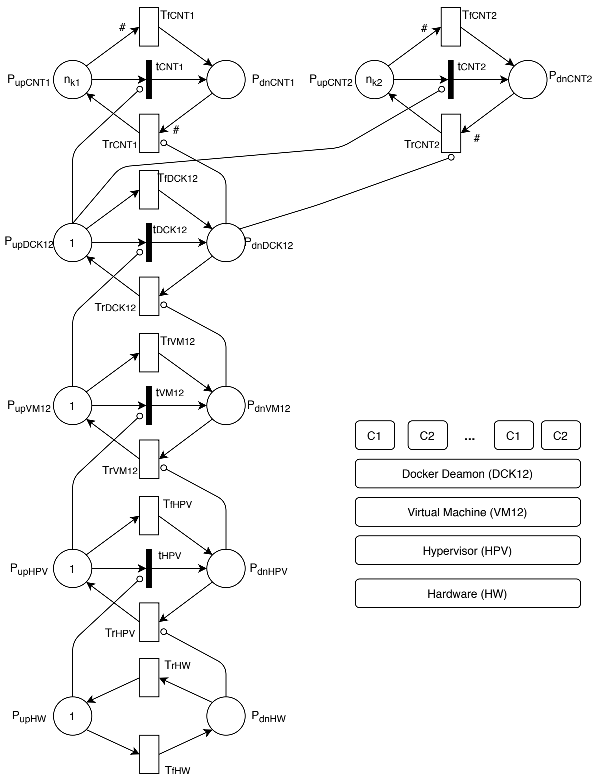


Figure 4.6: SRN-based model (on the left) representative of a generic CNR deployed according to the heterogeneous co-located scheme (on the right).

The new part allows to model the behavior of two separated substructures (each composed of container(s), docker, and VM) which share the same underlying infrastructure. The two inhibitory arcs (one from $P_{upDCK34}$ and t_{CNT4} and another from $P_{dnDCK34}$ to T_{rCNT4}) admit the same interpretation, *mutatis mutandis*, offered for the heterogeneous case. Let us now evaluate for the reward rate and the steady-state availability. Given marking i, we express the

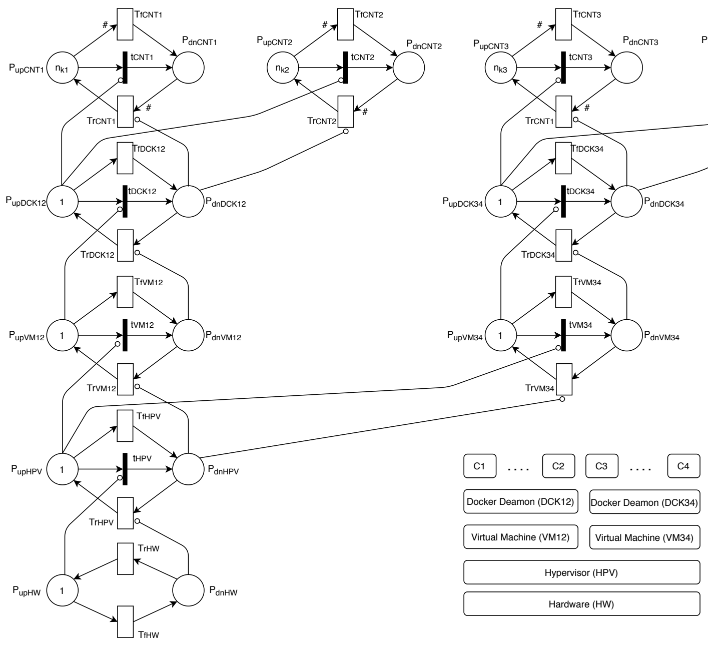


Figure 4.7: SRN-based model (on the left) representative of a generic CNR deployed according the heterogeneous co-located scheme (on the bottom-right).

reward rate as

$$r_i'''(j_1, j_2, j_3, j_4) = \begin{cases} 1 \text{ if} & \left(\sum_{h=1}^{k} \#P_{upCNT1}^{(h)} \geq \gamma_{j_1}\right) \wedge \\ & \left(\sum_{h=1}^{k} \#P_{upCNT2}^{(h)} \geq \gamma_{j_2}\right) \wedge \\ & \left(\sum_{h=1}^{k} \#P_{upCNT3}^{(h)} \geq \gamma_{j_3}\right) \wedge \\ & \left(\sum_{h=1}^{k} \#P_{upCNT4}^{(h)} \geq \gamma_{j_4}\right) \\ & \qquad\qquad \text{in marking } i, \\ & \\ 0 \text{ otherwise.} & \end{cases} \tag{4.9}$$

We notice that the above formula can be obtained from (4.3) by considering two couples of co-located containers with capacities c_{j1} (with j_1 representing I-CSCF), c_{j2} (with j_2 representing HSS), c_{j3} (with j_3 representing P-CSCF), and c_{j4} (with j_4 representing S-CSCF) in accordance to Fig. 4.2(d). Hence, the pertinent steady-state availability can be expressed as

$$A'''_{cIMS} = \lim_{t\to+\infty} A'''_{j_1j_2j_3j_4}(t) = \sum_{i\in\mathbb{I}} r'''_i(j_1,j_2,j_3,j_4)\, p'''_{ij_1j_2j_3j_4}, \quad (4.10)$$

The reward function $r'''_i(j_1,j_2,j_3,j_4)$ is given by (4.9) and $p'''_{ij_1j_2j_3j_4}$ is the corresponding steady-state probability.

4.1.9 The Availability/Cost Optimization Problem

Let setting $\mathcal{S}$ be a generic deployment of a cIMS infrastructure composed of a certain number of CNRs. Our goal is to find the settings satisfying high availability requirements at minimal cost (since CNRs can be variously combined among them and with different schemes, the optimum could be achieved by more than one setting). This optimization problem can be formalized as follows. Letting E_j be the cost (expenditure) of node j, composed of h CNRs, the overall cost of a cIMS setting is

$$E(\mathcal{S}) = \sum_{j\in\{P,S,I,H\}} E_j. \quad (4.11)$$

Letting $\mathcal{R} = \{\mathcal{S} : A_{cIMS}(\mathcal{S}) \geq A_0\}$ be the ensemble of settings satisfying a steady-state availability requirement A_0, the formal solution of the problem amounts to:

$$\mathcal{S}^* = \arg\min_{\mathcal{S}\in\mathcal{R}} E(\mathcal{S}). \quad (4.12)$$

We shall work under two assumptions. The first one concerns the cost computation/assignment of a cIMS system. Since a single cIMS node is composed of one or more CNRs, we assume that the cost of a single CNR is the sum of three dimensionless contributions: *i)* cost per container (CNT) embodying the software logic and licenses, *ii)* cost per docker and virtual machine (DCK+VM) that includes the operating system, and *iii)* cost per hypervisor and hardware (HPV+HW) representing the infrastructure cost. Each contribution is supposed to be equally priced with a normalized cost amounting to 1. Such assumption, in line with the policy pricing of top-player services such as Amazon AWS or Microsoft Azure, reflects the fact that software parts have a cost comparable with physical parts since an extra amount due to licenses must be considered. Needless to say, the proposed analysis can be generalized by customizing (4.11) and by choosing different cost contributions.

The second assumption concerns the diversity between CSCF containers (P, S, I) and the HSS, as the latter implies an additional criticality due to the underlying database structure. This issue is accounted for by imposing that HSS container provides one extra replica w.r.t. CSCF containers. In other words, we impose that $\gamma_{HSS} = \gamma_{CSCF} + 1$. To avoid overburdened notation, in the following we simply use γ in place of γ_{CSCF}.

4.1.10 OptChains+ Algorithm

In our analysis, we face a search across a huge number of possible redundancy schemes obtained by variously combining CNRs and pertinent containers.

Our analysis is assisted by TimeNET [94], a powerful tool for SRN model evaluation, whose functionalities have been further enriched by means of a purposely designed Python-based external module implementing a multi-stage procedure (sketched in Fig. 4.8) which:

- automatically builds, replicates (to achieve redundancy), and evaluates SRN models per cIMS node on the basis of some parameters such as: desired scheme (homogeneous, heterogeneous, etc.), Mean Time to Failure (MTTF) $1/\lambda$

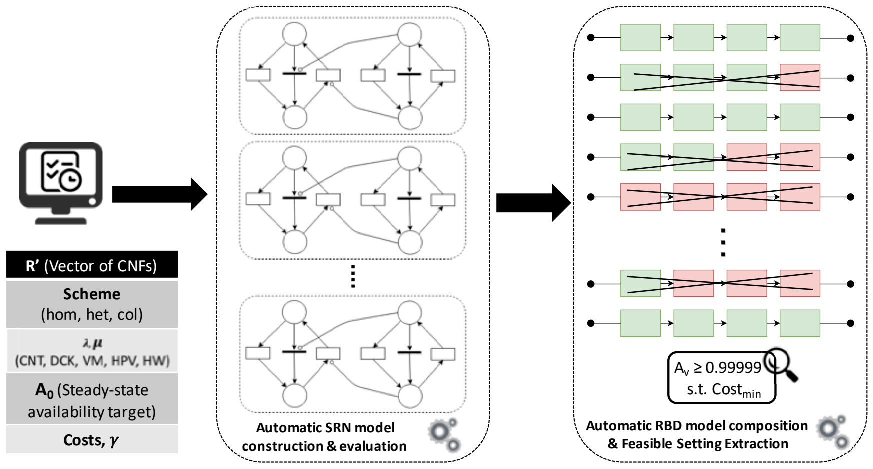

R' (Vector of CNFs)
Scheme
(hom, het, col)
λ, μ
(CNT, DCK, VM, HPV, HW)
A_0 (Steady-state availability target)
Costs, γ
Automatic SRN model construction & evaluation
$A_v \geq 0.99999$
s.t. $Cost_{min}$
Automatic RBD model composition & Feasible Setting Extraction

Figure 4.8: Big picture of the procedure implementing OptChains+ to support the performability assessment.

and Mean Time to Repair (MTTR) $1/\mu$ for various layers, desired steady-state availability target A_0 (0.99999 in our case), cost per layer, value of γ;

- automatically composes the series/parallel structures (settings) through the RBD formalism, along with the overall availability evaluation; at the same time, an extraction of feasible settings satisfying the desired constraints is performed.

The described procedure has been embedded into an algorithm dubbed OptChains+, whose pseudo-code is reported in the next page. The first part (lines $1-9$) embodies the external call to TimeNET to build and evaluate SRN models for single CNFs (made of one or more CNRs - see Fig. 4.3), by retaining only the (feasible) CNFs that cope with a given availability constraint ($A_{CNF} \geq A_0$, line 5). The rationale behind this choice is to save computational resources for the evaluation of the final availability A_{cIMS}, which is obtained as the product of A_{CNF} terms, one per node. We want to remark that such external call (line 4) is just preparatory to obtain the vector $\mathbf{R}$ of feasible CNFs, thus,

```
1  Initialize the vector R' containing all possible CNFs with various parameters (schemes, $\lambda$,
     $\mu$, $A_0$, costs, $\gamma$);
2  for CNF $\in$ R' do
3      if #Container $\geq \gamma$ then
4          SRN model evaluation (CNF)
5          if $A_{CNF} \geq A_0$ then
6              R[CNF] $\leftarrow$ $A_{CNF}$
7          end
8      end
9  end
   Intermediate Input: R, $g_1$, $g_2$, $g_3$, $G$
10 minCost $\leftarrow$ inf
11 for $p \in R_{pcscf}$ do
12     calculate $E_p$
13     if $E_p > g_1 \cdot minCost$ then
14         continue;
15     end
16     for $s \in R_{scscf}$ do
17         calculate $E_s$
18         if $E_p + E_s > g_2 \cdot minCost$
19         OR $(A_p \cdot A_s) < A_0$ then
20             continue;
21         end
22         if homogeneous then
23             for $i \in R_{icscf}$ do
24                 calculate $E_i$
25                 if $\sum_{k=p,s,i} E_k > g_3 \cdot minCost$
26                 OR $\prod_{k=p,s,i} A_k < A_0$ then
27                     continue;
28                 end
29                 for $h \in R_{hss}$ do
30                     calculate $E_h$
31                     if $\sum_{k=p,s,i,h} E_k > G \cdot minCost$
32                     OR $\prod_{k=p,s,i,h} A_k < A_0$ then
33                         continue;
34                     end
35                     minCost $\leftarrow$ min{minCost, $E_{cIMS}$}
36                     save [cIMS, $A_{cIMS}$, $E_{cIMS}$]
37                 end
38             end
39         end
40         else if colocated or heterog. then
41             for $c \in R_{col-het}$ do
42                 calculate $E_c$
43                 if $\sum_{k=p,s,c} E_k > G \cdot minCost$
44                 OR $\prod_{k=p,s,c} A_k < A_0$ then
45                     continue;
46                 end
47                 minCost $\leftarrow$ min{minCost, $E_{cIMS}$}
48                 save [cIMS, $A_{cIMS}$, $E_{cIMS}$]
49             end
50         end
51     end
52 end
```

in case a different tool is used, the rest of OptChains+ remains unaltered. The second part of the algorithm aims at achieving a reduced number of cIMS settings (matching specific costs E_{cIMS} and availability criteria A_{cIMS} at the same time) for different schemes (homogeneous (lines $22 - 39$), co-located/heterogeneous (lines $40 - 50$)). The intermediate inputs for such a second part include: $\mathbf{R}$, γ, and four weight factors (g_1, g_2, g_3, G) adopted to tune the pruning/searching process. The idea is to perform an exhaustive search with pruning, starting to cycle on the sub-vector $\mathbf{R}_{pcscf}$ that includes all feasible Proxy-type CNFs (line 11). The algorithm prunes all the settings with a cost exceeding g_1 times the cost of cIMS. The variable $minCost$ represents the whole cIMS cost calculated/updated within a cycle, and initialized at line 10. Then, when analyzing the sub-vector $\mathbf{R}_{scscf}$ (line 16), OptChains+ prunes all the settings whose total cost of Proxy and Serving-type CNFs exceeds g_2 times the cIMS cost, or whose availability product $A_p\,A_s$ is less than A_0. A similar logic holds for: I-type containers (line 23) and H-type containers (line 29). At line 31, the weight factor G indicates that an extra amount of settings (with a cost increased by $G\%$ regarding the actual cost) is retained for backup. The final output is a vector gathering: all the feasible cIMS settings along with their availability A_{cIMS} and cost E_{cIMS} (line 36 for homogenous schemes, and line 48 for colocated/heterogeneous schemes). We remark that any reasonable criterion can be pursued to select the weight factors. The practical rule we adopted is based on the assumption that each of the four nodes is worth 1/4 of the whole cIMS deployment. Hence, the algorithm starts to prune all settings whose P-CSCF cost exceeds its redoubled value, thus, $g_1 = 1/2$. Such a "conservative" reasoning is repeated further ahead in OptChains+, so as to obtain the rescaled weight factors $g_2 = 3/4$ and $g_3 = 1$. Ultimately, the value of G is set to 1.15, implying that we keep more settings than needed, with the aim of providing a broader set of cIMS combinations. Intuitively, greater values of weight factors result in a more conservative strategy since more settings are kept, but at the cost of a higher computation time. In our case study, which assumes a maximum of 6 containers to deploy per CNR, the variety of redundancy schemes to analyze produces a number of settings in the order of 10^{13} (consider combining 7 containers (0-6) deployed across 4 nodes, and, then, composing a setting of 4 elements: $(7^4)^4$).

It is worth noting that OptChains+ can be regarded as an approximation to a brute-force search with heuristic pruning rules.

Actually, relatively large weights (i.e., few pruned configurations) go in the direction of the brute-force search, reaching an approximate $O(n^4)$ complexity. In comparison, with relatively small weights the complexity can scale with the more affordable law $O(n \log n)$, which arises from a sorting operation of the pertinent cost vectors. Clearly, the amount of pruning can affect the final algorithm performance, since the more we prune, the higher is the probability of missing useful configurations. Unfortunately, there does not exist an analytical way to choose the weights so as to optimize the complexity/performance trade-off. In practice, heuristic thresholding rules are employed to set an appropriate level of pruning, as detailed before. As we will show in the experimental analysis, with typical choices of the pruning threshold the optimization routine exhibits reasonable complexity and is able to find (sub-)optimal configurations that attain satisfying performance.

With the proposed OptChains+ tuning, the number of settings to analyze decreases to 10^5, and, on a standard PC (Intel Core CPU $i5-3230$@2.60 GHz, with a RAM of 8 GB), the whole procedure requires about 450 seconds to run (neglecting the call to external tool TimeNET).

4.1.11 Numerical Evaluation

In Fig. 4.9 we sketch the deployed experimental testbed based on the Clearwater platform. On a laptop with an Intel Core CPU i7 – 3630QM@2.40GHz and with a RAM of 8 GB, we deploy two Linux-based virtual machines (1 virtual Core and 2 GB of RAM per VM): the first one serves as a containerized deployment of the whole cIMS architecture including P-CSCF (*Bono*), S/I-CSCF (*Sprout*), and HSS (*Homestead*). The second VM is a stress node that executes some routines useful to perform a load stress against the containerized platform. The test scenario considers 1000 concurrent IMS sessions with a BHCA (Busy Hour Call Attempts) equal to 2.6 per user (in line with values provided for VoLTE - see [95]). The resulting average call setup delay is 80 msec, a value

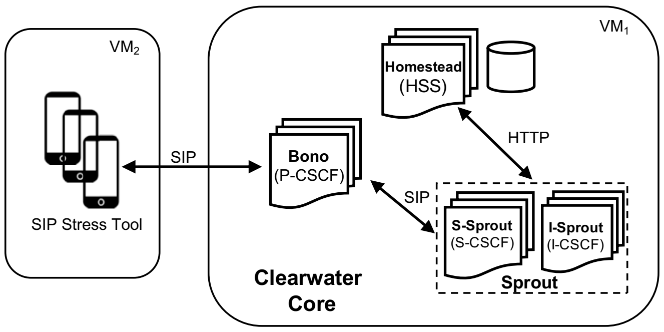


Figure 4.9: Sketch of the experimental testbed relying on the Clearwater architecture.

quite reasonable since the infrastructure is deployed on the same node, so that interconnection delays are negligible.

The values of some parameters (e.g. virtual machine and hypervisor MTTF) have been directly obtained from the technical literature [96]. The values of other parameters (e.g. container and docker MTTF and MTTR, see Table 4.1) have been chosen after consultation with experts from the ERICSSON company, within the research collaboration of the Co.Ri.TeL Laboratory of the University of Salerno.

The experiment allows for a performance demand W ranging from 2000 to 5000, assuming a performance capacity c=1000 (both in terms of IMS sessions) and assuming, for simplicity, $c_j = c$. This basically means that, if a provider needs to guarantee up to, say, 4000 concurrent IMS sessions, we get $\gamma = 4$, indicating the need for at least 4 containers. Hence, according to the W value, γ ranges from 2 to 5 (in case of non integer result, we consider the next integer value for γ).

Aimed at considering a practical case, we analyze some relevant settings (extracted among about 1000 produced by the procedure) as reported in Tables 4.2 and 4.3. The column *Scheme* indicates the type of cIMS deployment along with different values of γ. With a little abuse of notation, homogeneous (HOM.) scheme refers to a cIMS setting where all nodes are composed of homogeneous CNRs, whereas in the co-located (COL.) and

Table 4.1: Parameters values. CNT and DCK repair times must be interpreted as times spent to perform a software reboot.

Parameter	Description	Value
$1/\lambda_{CNT}$	container MTTF (hour)	500
$1/\lambda_{DCK}$	docker daemon MTTF (hour)	1000
$1/\lambda_{VM}$	virtual machine MTTF (hour)	2880
$1/\lambda_{HPV}$	hypervisor MTTF (hour)	2880
$1/\lambda_{HW}$	hardware MTTF (hour)	60000
$1/\mu_{CNT}$	container MTTR (sec)	2
$1/\mu_{DCK}$	docker daemon MTTR (sec)	5
$1/\mu_{VM}$	virtual machine MTTR (hour)	1
$1/\mu_{HPV}$	hypervisor MTTR (hour)	2
$1/\mu_{HW}$	hardware MTTR (hour)	8
W	performance demand (IMS sessions)	(2000, 5000)
c	performance capacity (IMS sessions)	1000
A_0	steady-state availability requirement	0.99999

heterogeneous (HET.) schemes, I-CSCF and HSS share the same CNR(s) of the co-located and the heterogeneous type, respectively. For each scheme, we consider four exemplary settings $(S_1, \dots, S_4)$ where a maximum of 4 CNRs per node are allowed. Let us clarify the notation adopted in Tables 4.2 and 4.3 by considering, for instance, setting S_1 in the co-located scheme with $\gamma = 2$. The notation ([2 2 0 0]) used for P-CSCF indicates that 2 out of 4 (homogeneous) CNRs are exploited and 2 containers per CNR are used. Similarly, for the S-CSCF ([1 1 1 0]), 3 out of 4 (homogeneous) CNRs are exploited and 1 container per CNR is used.

A slightly different notation is used for I-CSCF and HSS that share the same CNRs (Table 4.3). In such a case, $[2,3 \quad 2,3 \quad 0,0 \quad 0,0]$ indicates that 2 out of 4 (co-located) CNRs are exploited where 2 I-type containers and 3 H-type containers are deployed per CNR, respectively.

Such a concise notation is also helpful to quickly compute the cost E for each setting. As regards the previous example, the deployment cost E_P for P-CSCF amounts to $1 \cdot 4$ (CNT) $+1 \cdot 2$

Table 4.2: A selection of 4 exemplary settings (S_1, S_2, S_3, S_4) with different distributions of CNRs for homogeneous deployment and for values of γ ranging from 2 to 5.

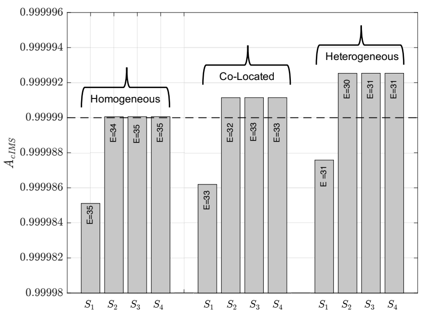
Scheme	Setting	P-CSCF	S-CSCF	I-CSCF	HSS
HOM. γ=2	S_1	[2 2 0 0]	[2 2 0 0]	[2 2 0 0]	[2 2 1 0]
	S_2	[2 2 0 0]	[2 2 0 0]	[2 2 0 0]	[3 3 0 0]
	S_3	[2 2 0 0]	[2 2 0 0]	[2 2 0 0]	[3 4 0 0]
	S_4	[2 2 0 0]	[2 2 0 0]	[2 3 0 0]	[3 3 0 0]
HOM. γ=4	S_1	[4 4 0 0]	[4 4 0 0]	[4 4 0 0]	[3 3 2 0]
	S_2	[4 4 0 0]	[4 4 0 0]	[4 4 0 0]	[5 5 0 0]
	S_3	[4 4 0 0]	[4 4 0 0]	[4 4 0 0]	[5 6 0 0]
	S_4	[4 4 0 0]	[4 4 0 0]	[4 5 0 0]	[5 5 0 0]
HOM. γ=3	S_1	[3 3 0 0]	[3 3 0 0]	[3 3 0 0]	[2 2 2 0]
	S_2	[3 3 0 0]	[3 3 0 0]	[3 3 0 0]	[4 4 0 0]
	S_3	[3 3 0 0]	[3 3 0 0]	[3 3 0 0]	[4 5 0 0]
	S_4	[3 3 0 0]	[3 3 0 0]	[3 4 0 0]	[4 4 0 0]
HOM. γ=5	S_1	[5 5 0 0]	[5 5 0 0]	[5 5 0 0]	[3 3 3 0]
	S_2	[5 5 0 0]	[5 5 0 0]	[5 5 0 0]	[6 6 0 0]
	S_3	[5 5 0 0]	[5 5 0 0]	[5 6 0 0]	[6 6 0 0]
	S_4	[5 6 0 0]	[5 5 0 0]	[5 5 0 0]	[6 6 0 0]

(DCK+VM) $+1 \cdot 2$ (HPV+HW); cost E_S for S-CSCF amounts to $1 \cdot 3$ (CNT) $+1 \cdot 3$ (DCK+VM) $+1 \cdot 3$ (HPV+HW); cost $E_{I,H}$ for the co-located I-CSCF and HSS amounts to $1 \cdot 10$ (CNT) $+1 \cdot 4$ (DCK+VM) $+1 \cdot 2$ (HPV+HW). The total cost amounts to $E = E_P + E_S + E_{I,H} = 33$.

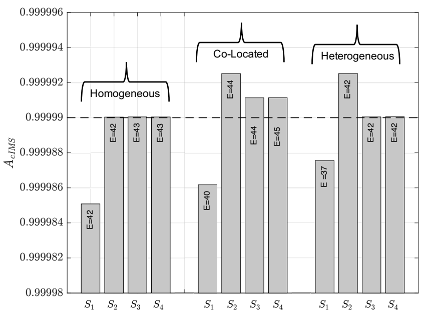
Let us now explore the results in terms of availability and costs for various settings through Fig. 4.10, where we report the steady-state availability A_{cIMS} for different values of γ. Let us consider, for instance, Fig. 4.10(b) showing the case $\gamma = 3$, where the four exemplary settings have been grouped per scheme. Each bar indicates the availability value, whereas the number inside reports the cost associated to that particular setting. The horizontal dashed line represents the "five nines" threshold that, if crossed, means that the pertinent setting does not match the availability requirement.

Table 4.3: A selection of 4 exemplary settings (S_1, S_2, S_3, S_4) with different distributions of CNRs for the co-located and the heterogeneous deployments and for values of γ ranging from 2 to 5.

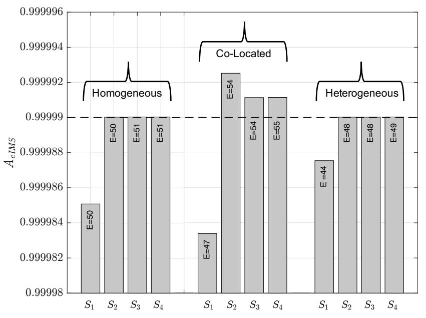
Scheme	Setting	P-CSCF	S-CSCF	I,H (CNR shared)
COL. γ=2	S1	[2 2 0 0]	[1 1 1 0]	[2,3 2,3 0,0 0,0]
	S2	[2 2 0 0]	[2 2 0 0]	[2,3 2,3 0,0 0,0]
	S3	[2 2 0 0]	[2 3 0 0]	[2,3 2,3 0,0 0,0]
	S4	[2 2 0 0]	[2 2 0 0]	[2,3 2,3 0,0 0,0]
COL. γ=3	S1	[3 3 0 0]	[3 3 0 0]	[3,2 3,2 0,2 0,0]
	S2	[3 3 0 0]	[3 3 0 0]	[3,3 3,3 0,1 0,1]
	S3	[3 3 0 0]	[3 3 0 0]	[3,2 3,2 0,2 0,2]
	S4	[3 4 0 0]	[3 3 0 0]	[3,3 3,3 0,1 0,1]
COL. γ=4	S1	[4 4 0 0]	[4 4 0 0]	[2,3 2,3 2,2 0,0]
	S2	[4 4 0 0]	[4 4 0 0]	[3,3 3,3 1,2 1,2]
	S3	[4 4 0 0]	[4 4 0 0]	[2,3 2,3 2,2 2,2]
	S4	[4 5 0 0]	[4 4 0 0]	[3,3 3,3 1,2 1,2]
COL. γ=5	S1	[5 5 0 0]	[5 5 0 0]	[3,3 3,3 2,3 0,0]
	S2	[5 5 0 0]	[2 3 3 3]	[2,3 3,3 3,3 3,0]
	S3	[2 3 3 3]	[5 5 0 0]	[2,3 3,3 3,3 3,0]
	S4	[5 5 0 0]	[2 3 3 4]	[2,3 3,3 3,3 3,0]
HET. γ=2	S1	[1 1 1 0]	[2 2 0 0]	[2,3 2,3 0,0 0,0]
	S2	[2 2 0 0]	[2 2 0 0]	[2,3 2,3 0,0 0,0]
	S3	[2 2 0 0]	[2 3 0 0]	[2,3 2,3 0,0 0,0]
	S4	[2 3 0 0]	[2 2 0 0]	[2,3 2,3 0,0 0,0]
HET. γ=3	S1	[3 3 0 0]	[3 3 0 0]	[2,2 2,2 1,2 0,0]
	S2	[3 3 0 0]	[3 3 0 0]	[3,3 3,3 0,1 0,1]
	S3	[3 3 0 0]	[3 3 0 0]	[3,1 3,2 0,2 0,3]
	S4	[3 3 0 0]	[3 3 3 0]	[2,2 2,2 1,2 0,0]
HET. γ=4	S1	[4 4 0 0]	[4 4 0 0]	[2,3 2,3 2,2 0,0]
	S2	[4 4 0 0]	[2 2 2 2]	[2,3 2,3 2,2 0,0]
	S3	[2 2 2 2]	[4 4 0 0]	[2,3 2,3 2,2 0,0]
	S4	[4 4 0 0]	[2 2 2 2]	[3,3 1,3 1,2 0,0]
HET. γ=5	S1	[5 5 0 0]	[5 6 2 0]	[3,3 3,3 2,3 0,0]
	S2	[5 5 0 0]	[2 3 3 3]	[3,3 3,3 2,3 0,0]
	S3	[2 3 3 3]	[5 5 0 0]	[3,3 3,3 2,3 0,0]
	S4	[5 5 0 0]	[2 3 3 4]	[3,3 3,3 2,3 0,0]



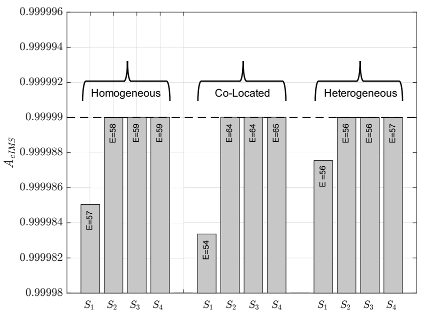
(a) Availability for $\gamma = 2$.



(b) Availability for $\gamma = 3$.



(c) Availability for $\gamma = 4$.



(d) Availability for $\gamma = 5$.

Figure 4.10: Steady-state availability considering 4 exemplary settings (S_1, S_2, S_3, S_4) per scheme for: $\gamma = 2, 3, 4, 5$.

In order to put forth some unexpected and interesting behaviors, for each case we report also a setting satisfying the "four nines" but not the "five nines" condition. This is the case of S_1, whose availability values are 0.999985, 0.999986, and 0.999987, for homogeneous, co-located, and heterogeneous schemes, respectively.

Let us now focus on the homogeneous scheme: among settings S_2, S_3, and S_4 that barely satisfy the availability constraint, S_2 has the lowest cost ($E = 42$), so we elect this setting as the best one. We notice that S_1 achieves the same cost as S_2, but with a different distribution of containers in the HSS node (see Table 4.2). For the co-located scheme, we also consider S_2 as optimal although the same cost ($E = 44$) is achieved by S_3 but at lower availability level (0.9999911 for S_3 vs. 0.9999925 for S_2). This notwithstanding, a network designer could more comfortably choose S_3, should the uniformity of replica distribution be at a premium (related, may be, to deployment flexibility). In such a case, in fact, being all CSCF nodes equal, HSS exhibits distributions $[2 \quad 2 \quad 2 \quad 2]$ for S_3 and $[3 \quad 3 \quad 1 \quad 1]$ for S_2.

Similar considerations hold true for the heterogeneous scheme where, again, S_2 emerges as the setting satisfying the best trade-off between availability and cost.

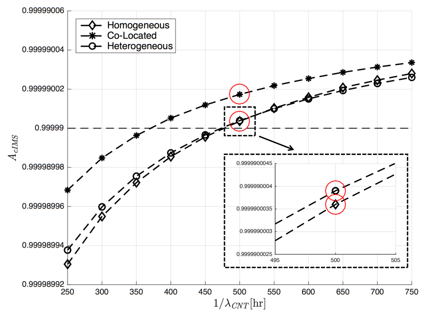
Now, consider the case $\gamma = 5$, whose availability results are shown in Fig. 4.10(d). In comparison to case $\gamma = 3$, two facts emerge: first, the availability values for settings S_2, S_3, and S_4 (those able to guarantee the "five nines") are almost equal (in each scheme), and are very close to the 0.99999 threshold. Basically, this is related to the need of achieving high availability requirements with a more challenging performance level that, in turn, implies more redundancy at a container level for all considered settings and schemes. Second, the difference in terms of costs between the homogeneous and the co-located settings becomes more pronounced. This behavior can be explained as follows. For $\gamma = 5$, the system has to manage a greater number of cIMS sessions w.r.t. the case $\gamma = 3$, which in turn implies that we need more containers. When the number of containers grows, the co-located setting is saturated more "quickly" than the homogeneous one, thus more DCK and VM levels are needed, resulting in additional costs.

In conclusion, watching from afar the availability results, two aspects should be highlighted. The first aspect concerns the monotonic increase of the cost with γ, since more resources are needed (in terms of CNRs and/or containers). The second aspect pertains to the choice of a particular scheme among the three considered: according to the performed analysis, in fact, the co-located and the heterogeneous schemes offer the best trade-off in terms of cost and availability when the performance level is not so high. Basically, this is due to the possibility of arranging containers in a more ductile way, since the homogeneous scheme forces to introduce a new CNR in case different types of containers have to be deployed. On the other hand, when high performance level is required, the redundancy at CNR level is needed also for the co-located and the heterogeneous schemes, thus, a homogeneous arrangement becomes more attractive in terms of cost reduction.

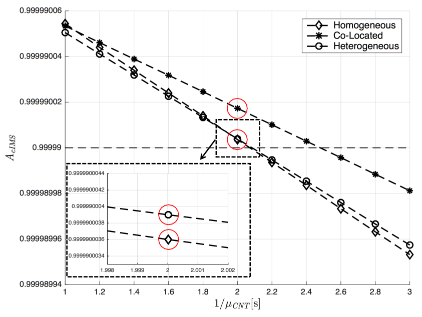
4.1.12 Sensitivity Analysis

We carry out a sensitivity analysis useful, from the designer's perspective, to cope with parameters uncertainty. Precisely, we evaluate the effects of drifts from nominal values (see Table 4.1) for six critical parameters: failure and repair times pertaining to container, docker, and virtual machine layers. The results are reported in Fig. 4.11, starting from the best settings (S_2) derived in the previous analysis for the case $\gamma = 5$. Let us first analyze the Fig. 4.11(a) showing the sensitivity analysis for the container failure time ($1/\lambda_{CNT}$). It is possible to observe that, for the case of co-located scheme, the failure time can be reduced from its nominal value (500 hours, circled in red) to about 370 hours with no side effects on the availability, since the corresponding curve remains above the horizontal dashed line (five nines limit). For the homogeneous and heterogeneous schemes, such analysis reveals a similar behavior (see the zoomed inset), but with more stringent margins since nominal values can be reduced from 500 hours to not less than 480 hours. However the improved robustness for the co-located scheme is paid in the coin of a higher cost (see Fig. 4.10(d)). Similar arguments hold for the container repair time sensitivity as shown in Fig. 4.11(b). In fact, the nominal value of $1/\mu_{CNT}$ can be relaxed from 2 seconds to about 2.5 seconds for the co-located scheme, and to about 2.15 seconds for the homogeneous and heterogeneous schemes.

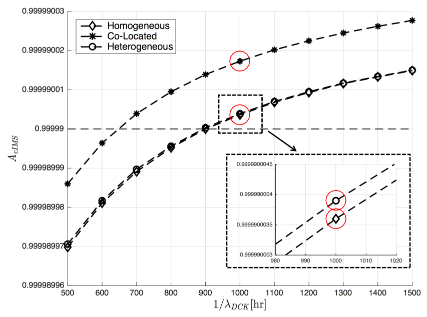
In Figs. 4.11(c) and 4.11(d), we display the results relative to docker failure and repair times, respectively. On one hand, we can see that the nominal value of $1/\lambda_{DCK}$ can be decreased from 1000 hours to about 640 hours (co-located) or to about 900 hours (homogeneous and heterogeneous cases). On the other hand, the



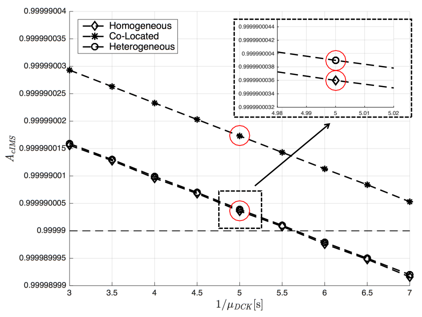
(a)



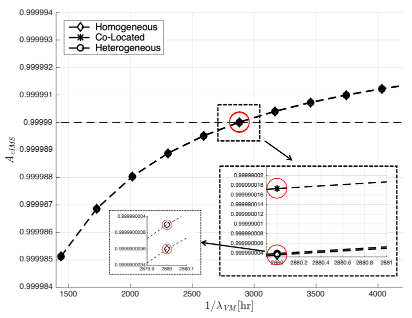
(b)



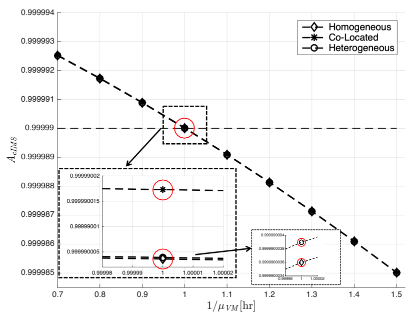
(c)



(d)



(e)



(f)

Figure 4.11: Influence on the overall cIMS infrastructure (case $\gamma = 5$) of: container failure time (a), container repair time (b), docker failure time (c), docker repair time (d), virtual machine failure time (e), virtual machine repair time (f). Nominal values (reported in Table 4.1) are circled in red.

nominal value of $1/\mu_{DCK}$ can be relaxed from 5 seconds to about 9 seconds for the co-located scheme, and to about 5.6 seconds in case of the homogeneous and heterogeneous schemes. Also in this case, the improved robustness to deviation of docker parameters is paid in terms of a higher cost of deployment.

Finally, we analyze the sensitivity for VM failure and repair times as reported in Figs. 4.11(e) and 4.11(f), respectively. The nominal value of $1/\lambda_{VM}$ can be diminished from 2880 hours to 2870 hours (co-located) and to 2878 hours (homogeneous/heterogeneous) with no side effects on the high availability requirement. On the contrary, the margin for the parameter $1/\mu_{VM}$ is even more stringent: it can be relaxed from 1 hour to 1 hour and 1 second (co-located) and to 1 hour and 7 seconds (homogeneous/heterogeneous).

In summary, the sensitivity analysis reveals that the robustness of the whole cIMS is influenced by two factors: *i)* the robustness of the individual layers, which for some cases (CNT, DCK) exhibits a reasonable margin, whereas in other cases (VM) is practically not amenable to any significant deviation; *ii)* the type of deployment where, typically, the co-located scheme offers more room for manoeuvre.

4.2 The NFV-based model of IP Multimedia Subsystem

In this section we consider a virtualized implementation of an IMS system (vIMS) in a Network Function Virtualization (NFV) environment that we characterize by an availability standpoint. NFV [97] represents one of the most innovative paradigms within the fifth generation (5G) of telecommunication systems. Basically, it has been designed to boost the deployment of new network services by exploiting the virtualization concepts. Within an NFV domain, traditional network equipments (e.g. firewalls, routers, switches, etc.) are replaced by their virtual counterparts named Virtualized Network Functions (VNFs). Differently from the containerization paradigm, NFV guarantees a more secure environment since each VNF lies within a dedicated "sandbox". Moreover, the VNF model is easier than the CNF model (see Sect. 4.1.3) since only 3 layers (in place of 5) are present: the hardware layer, the hypervisor layer, and the application layer. This latter embodies a specific IMS function (e.g. the proxy function, the interrogating function, etc.). Similarly to the CNF case, the probabilistic behavior of a VNF has been modeled through the Stochastic Reward Nets. Innovating on previous formulations, part of the analysis is carried out by adopting non-Markovian models, thus allowing for more realistic (non-exponentially distributed) times between some state transitions. As final results, we determine the optimal redundant vIMS configuration able to guarantee a steady-state availability not less than 0.99999, and we provide a sensitivity analysis useful to evaluate the system robustness to the variation of parameters from their nominal values.

4.2.1 Availability Model of virtualized IMS

We continue to follow the two-level hierarchical approach adopted in the containerized case. The first level is the RBD modeling, allowing to characterize the vIMS in terms of interconnections among nodes. Figure 4.12 shows the RBD representation derived from the Registration scenario reported in Fig. 4.1, where a series model is necessary to characterize the IMS architecture. In fact, all network functionalities must be active to guarantee the Registration service to the users. In contrast, a parallel configuration for each node (replicas) is useful to ensure a certain degree of redundancy in case of failures. Furthermore, we assume that the HSS node is deployed in a *k-out-of* configuration, where k represents the number of HSS replicas that must work to make the HSS node to work. Henceforth, we assume $k = 2$, in accordance with many actual deployments.

On the other hand, the second level of the hierarchical model relies on the SRN formalism amenable to describe the internal behavior of a single node in terms of failure/repair events. In the

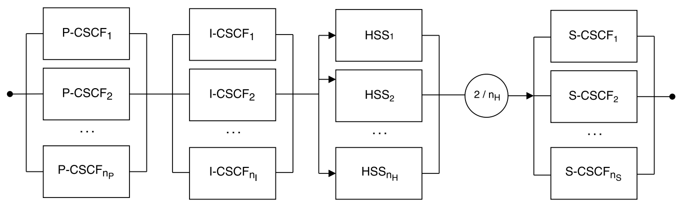


Figure 4.12: The Reliability Block Diagram representation of a virtualized IMS infrastructure, where HSS is deployed in a 2-out-of-n_H redundancy configuration.

following paragraph we start from the SRN characterization of vIMS nodes.

4.2.2 Stochastic Reward Networks for vIMS

In this section, we provide a specific model of a generic vIMS node using the SRN formalism described in Section 2.2.1.

The SRN of a generic vIMS node replica (CSCF or HSS) is shown in Fig. 4.13, while the overall vIMS infrastructure is described by the RDB model illustrated in Fig. 4.12.

By inspection of Fig. 4.13 it is possible to distinguish the following entities:

- *Places* (circles): the group of places P_{upHW}, P_{upVMM}, and P_{upAPP}, takes into account the working conditions of hardware, hypervisor (VMM subscript stands for Virtual Machine Monitor), and application layers, respectively. The numbers inside the three places (tokens) indicate the corresponding initial (working) conditions. Conversely, the places P_{dnHW}, P_{dnVMM}, and P_{dnAPP}, represent the failure conditions of hardware, hypervisor, and application layers, respectively.

- *Timed Transitions* (unfilled rectangles): such transitions take into account the various layers behavior; in particular, T_{fAPP} [T_{rAPP}], T_{fVMM} [T_{rVMM}], and T_{fHW} [T_{rHW}] denote

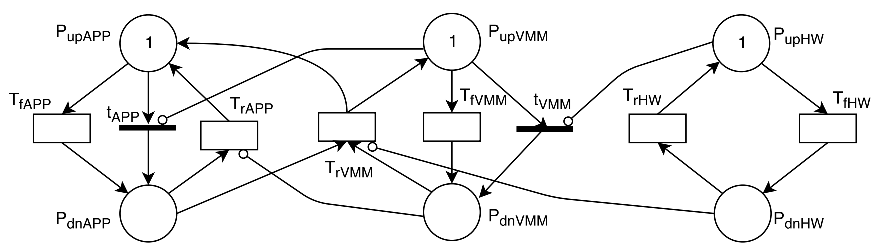


Figure 4.13: SRN representation of one generic node replica of the vIMS network infrastructure.

the failure [repair] events of the application, the hypervisor, and the hardware, respectively.

- *Immediate Transitions* (thin and filled rectangles): such transitions take into account the instantaneous actions. In the proposed SRN, two immediate transitions appear: t_{APP} and t_{VMM}.

4.2.3 Model Evolution

In this section we analyze the dynamics of the system, namely, the conditions arising when events such as failures or repairs emerge. In particular, we focus on the evolution of the SRN model of a single vIMS node. For the sake of simplicity, it is useful to consider an initial fully working condition for the node characterized by a token in each P_{up} place of the SRN. If an application failure happens, it means that the software function representing the logic of a vIMS node (a CSCF node or the HSS node) breaks. In this case, the transition T_{fAPP} is fired, and the token leaves place P_{upAPP} to enter place P_{dnAPP}. Once the application gets repaired (sometimes a reboot procedure could solve the problem), the transition T_{rAPP} is fired, and the token comes back to initial place P_{upAPP}. In case of hypervisor failure, instead, the transition T_{fVMM} is fired, and the token is moved from place P_{upVMM} to place P_{dnVMM}. It is worth noting that place P_{upVMM} is connected to immediate transition t_{APP} by an *inhibitory* arc (the segment with a small circle close to t_{APP}). Such arc forces t_{APP} to get fired in order to model an application failure. The application layer, in fact, needs a working hypervisor layer to work correctly. On the contrary, when the hypervisor gets repaired, the token is again moved from P_{dnVMM} to P_{upVMM}, and the inhibitory arc is now ineffective. A similar reasoning holds for the failure of hardware. The token passes from P_{upHW} to P_{dnHW} as transition T_{fHW} gets fired. In such a case, an inhibitory arc (between P_{upHW} and t_{VMM}) forces t_{VMM} to move the token from P_{upVMM} to P_{dnVMM} since the hypervisor layer needs an underlying functioning hardware to work properly. It is interesting to note that another inhibitory arc connects P_{dnHW} and T_{rVMM}. Such arc inhibits transition T_{rVMM} (and consequently transition T_{rAPP}), until the hardware layer gets repaired.

The reward rate $r_i(j)$ associated to marking i is given by:

$$r_i(j) = \begin{cases} 1 & \text{if } \#P_{upAPP} = 1 \quad \text{in marking } i, \\ \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

It is worth noting that such reward rate does not need to account for an "up" condition of hypervisor and hardware because this condition is automatically embedded in the SRN model in Fig. 4.12, where inhibitory arcs avoid having a working application layer coupled with failed hypervisor and/or hardware layers.

We recall that the overall vIMS system can be modeled as a series/parallel of *independent* subsystems as shown in Fig. 4.12 and the steady-state availability of the subsystems is derived from (4.4). Accordingly, the overall vIMS steady-state availability can be expressed as

$$A_{vIMS} = \left[1 - \prod_{j=1}^{n_P} \left(1 - A_j^P\right)\right] \cdot \tag{4.14}$$
$$\left[1 - \prod_{j=1}^{n_S} \left(1 - A_j^S\right)\right] \left[1 - \prod_{j=1}^{n_I} \left(1 - A_j^I\right)\right] \cdot$$
$$\sum_{j=2}^{n_H} \binom{n_H}{j} A^H \left(1 - A^H\right)^{n_H - j},$$

where A_j^P, A_j^S, and A_j^I are the steady-state availabilities of the j-th replica of nodes P-CSCF, S-CSCF and I-CSCF, respectively, while the steady-state availability of HSS node replicas is $A_j^H = A^H$, $\forall j$; the numbers of redundant node replicas of each network functionality are n_P, n_S, n_I, and n_H, respectively. Being the vIMS a series system of network functionalities, the vIMS steady-state availability (4.14) is a product of single node availabilities, where P-CSCF, S-CSCF and I-CSCF nodes are in parallel configuration and provide the first three terms in (4.14). The last factor, instead, takes into account the *k-out-of-n* configuration of the HSS node, where $k = 2$ and $n = n_H$.

4.2.4 Numerical Analysis

This section presents a numerical analysis of the proposed IMS architecture over an NFV environment by exploiting two software tools: SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) [98] and TimeNet [94]. It aims to single out the minimal-cost redundant configuration of the virtualized IMS system able to guarantee the "five nines" requirement for telecommunication systems availability. The parameters used for this analysis, representative of the mean time of failures and repairs as regards the various components (hardware, hypervisor, application) are reported in Table 4.4. For the sake of simplicity we assume that: *i)* hardware and hypervisor are supposed to be

Table 4.4: Input parameters for the pertinent subsystems

Parameter	Description	Value
$1/\lambda_{HW}$	mean time for hardware failure	60000 hours
$1/\lambda_{VMM}$	mean time for hypervisor failure	5000 hours
$1/\lambda_{CSCF}$	mean time for CSCF node failure	3000 hours
$1/\lambda_{HSS}$	mean time for HSS node failure	2000 hours
$1/\mu_{HW}$	mean time for hardware repair	8 hours
$1/\mu_{VMM}$	mean time for hypervisor repair	2 hours
$1/\mu_{CSCF}$	mean time for CSCF software repair	1 hour
$1/\mu_{HSS}$	mean time for HSS software repair	1 hour

Table 4.5: Availability results of the whole virtualized IMS

Setting	Redundancy Level	A_{vIMS}
S_1	$CSCF = [2, 2, 2]$, $HSS = 3$	0.99999416
S_2	$CSCF = [2, 2, 2]$, $HSS = 4$	0.99999756
S_3	$CSCF = [2, 2, 3]$, $HSS = 3$	0.99999497
S_4	$CSCF = [3, 3, 3]$, $HSS = 3$	0.99999659
S_5	$CSCF = [2, 3, 3]$, $HSS = 4$	0.99999918

the same for all the nodes; *ii)* all the software instances running on CSCF nodes (P-CSCF, I-CSCF, S-CSCF) are characterized by the same failure and repair times. On the contrary, the software instance running on the HSS is supposed to have a different mean time of failure, being the database a more delicate and prone to failures element. In order to characterize the stationary availability of the vIMS system over long runs, a steady-state analysis is carried out by considering some exemplary settings as shown in Table 4.5.

The first column of Table 4.5 indicates the setting identifier ($S_1, \ldots, S_5$). The second column indicates the considered redundancy level; for example, setting S_3 is characterized by two (whatever) CSCF nodes having redundancy 2, the remaining CSCF node having redundancy 3, and the HSS node having redundancy 3. The third column indicates the steady-state availability value of the whole virtualized IMS system in Fig. 4.12.

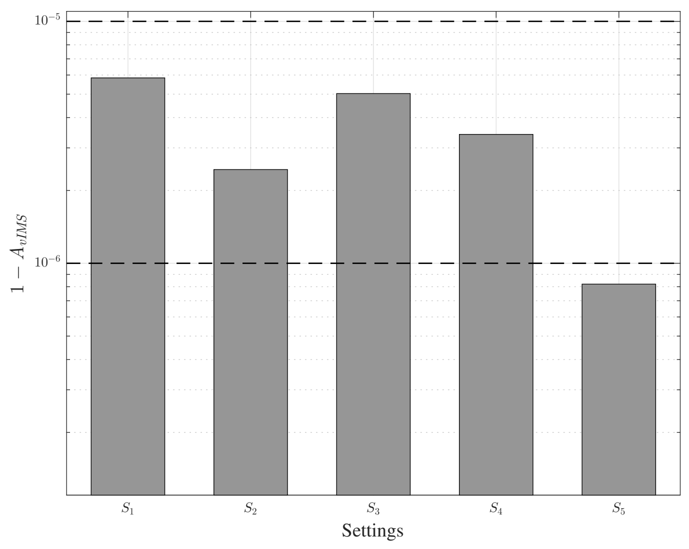


Figure 4.14: Steady-state unavailability for different settings $S_1, S_2, \ldots, S_5$. The above horizontal dashed line represents the required unavailability: $1 - A_{vIMS} = 10^{-5}$.

In order to visualize in a more comfortable manner the steady-state availability results, we show in Fig. 4.14 the unavailability of the virtualized IMS system $1 - A_{vIMS}$, and it is possible to observe that each setting S_i satisfies the "five nines" requirement, namely, each bar lies below the horizontal dashed line at $1 - A_{vIMS} = 10^{-5}$. In the case of setting S_5, the system is even able to satisfy the more challenging "six nines" requirement, being the corresponding bar lying below the horizontal dashed line at $1 - A_{vIMS} = 10^{-6}$. Among the considered settings, S_1 entails the minimum number of deployed replicas, namely, 2 replicas for each CSCF node and 3 replicas for the HSS node. Consequently, setting S_1 represents the optimal redundant configuration in terms of minimum number of deployed replicas while fulfilling the desired high availability requirement.

4.2.5 Transient non-Markovian Analysis

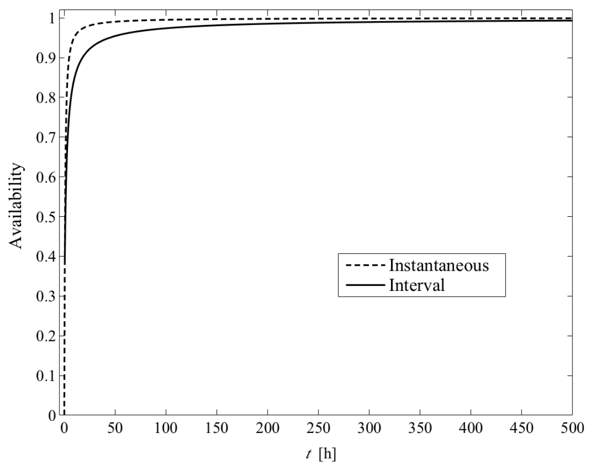
The performed regime analysis is useful to evaluate the system behavior as $t \to \infty$, but it cannot capture the dynamics of the system when, for example, some node instances are changing their states from failed to repaired. We approach this issue analyzing the dynamics of the system when software instances (namely the application parts) of both P-CSCF node replicas are down and ready to be repaired. This is the typical case of an unplanned update of the operating system that forces both software instances to be rebooted.

Actually, in order to consider a more realistic scenario, we replace the classical hypothesis of exponentially distributed software repair transition times with a Weibull-distributed one. In particular, the transient analysis evaluates the behavior of the instantaneous availability $A_{vIMS}(t)$ and the interval availability $\overline{A_{vIMS}}(t)$ of the vIMS system in $(0, t]$, which is defined as

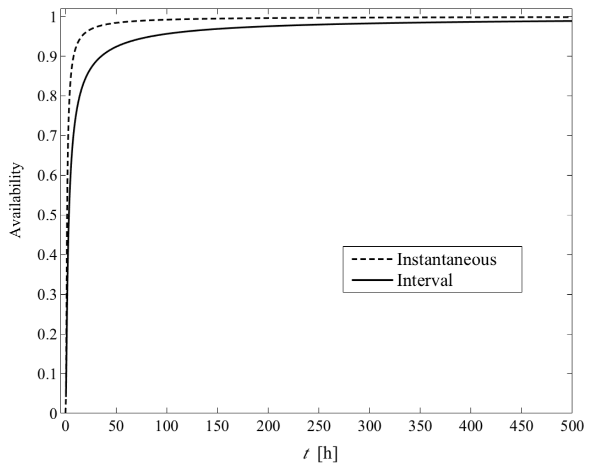
$$\overline{A_{vIMS}}(t) = \frac{1}{t}\int_0^t A_{vIMS}(u)\mathrm{d}u. \tag{4.15}$$

The interval availability represents the time average of the instantaneous availability function over the interval $(0, t]$. Figure 4.15 shows the behavior of $A_{vIMS}(t)$ and of $\overline{A_{vIMS}}(t)$ when a Markovian process (all exponential transition times) and a non-Markovian process (Weibull software repair transition times) are considered. In the case of Weibull software repair transition times, we consider the same mean repair time used in the case of exponentially distributed software repair transition. In particular, following [99] we set the shape parameter of the Weibull distribution to $\alpha = 3$, while the scale parameter β turns out to be 1.1198.

Besides, Fig. 4.16 highlights the different behavior of instantaneous availability $A_{vIMS}(t)$ during the transient, in the case of exponential and Weibull failure rate (of the application part). It is worth noting that the transient of $A_{vIMS}(t)$ in the Weibull case is slower than the transient of $A_{vIMS}(t)$ in the exponential case. This behavior is compatible with real-world effects [100]. In both



(a) Markovian process (Exponential repair time).



(b) Non-Markovian process with a Weibull repair time.

Figure 4.15: Instantaneous ($A_{vIMS}(t)$) and interval ($\overline{A_{vIMS}}(t)$) availability in case of Markovian and non-Markovian process for the repair time of the P-CSCF application part.

cases, as expected, the interval availability $\overline{A_{vIMS}}(t)$ in (4.15) converges more slowly to the steady-state availability with respect to

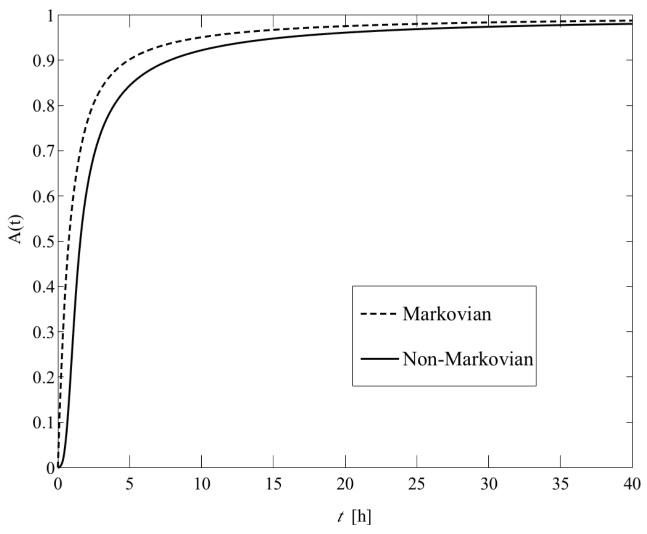


Figure 4.16: Instantaneous availability comparison in case of Markovian process (all exponential transition times) and non-Markovian process (Weibull repair time of the P-CSCF application part).

the instantaneous availability $A_{vIMS}(t)$.

4.2.6 Sensitivity Analysis

In relation to the last part of the considered experiment, we perform a sensitivity analysis with respect to deflections of two system parameters from their nominal values: λ_{P-CSCF} and λ_{HSS}. Such an analysis has been carried out by considering the minimal cost setting S_1 obtained as a result of the regime analysis. In Fig. 4.17, the influence of the P-CSCF application part failure time has been considered. The nominal value amounts to 3000 hours (see Table 4.1), but if we relax such a value to about 1500 hours, we are still able to satisfy the "five nines" requirement. At $1/\lambda_{P-CSCF} = 1500$, in fact, the value A_{vIMS} lies approximately around to 0.9999935. Figure 4.18, instead, shows the influence of the HSS application part failure time. In this case, the nominal value amounts to 2000 hours, and no side effects are notable un-

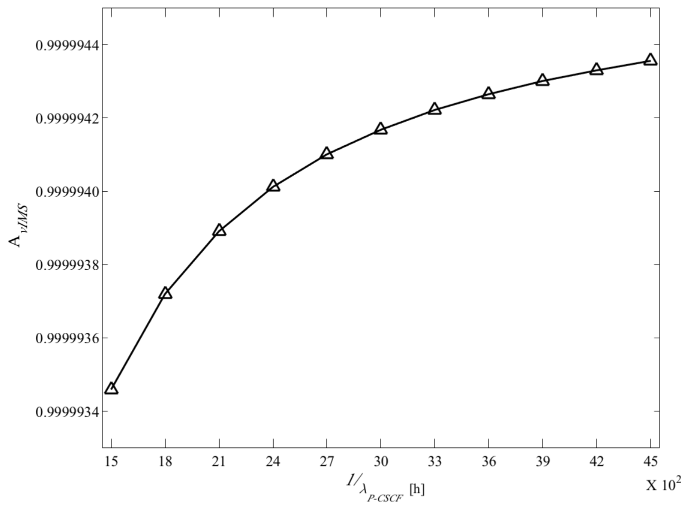


Figure 4.17: Influence of failure time $1/\lambda_{P-CSCF}$ over system availability, in case of optimal setting S_1.

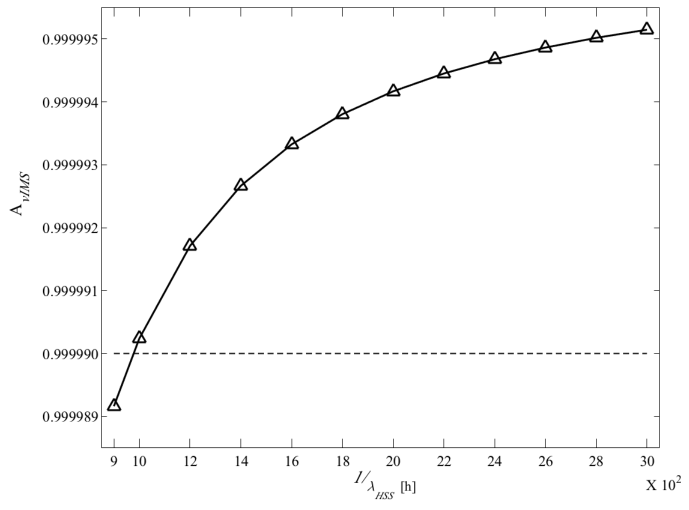


Figure 4.18: Influence of failure time $1/\lambda_{HSS}$ over system availability, in case of optimal setting S_1.

less $1/\lambda_{HSS}$ takes the value 1000 (approximately). In this case, the horizontal dashed line at 0.999990 is useful to identify the *breakpoint* of the high availability requirement.

4.3 A Tool for HA Model Generation

Network providers reputation strongly depends on their ability to guarantee high performance levels of virtualized infrastructures, and to maintain strict Quality of Service (QoS) requirements, thus, the concept of "five nines" or high availability (HA) is critical.

Aimed at facilitating the definition and analysis of availability models, we design a model-driven framework which assists a user to perform an availability assessment of modern virtualized architectures embracing the SFCs paradigm. The SFCs chained software logic is embodied in many NFV telecommunication infrastructures (e.g., the Virtual IP Multimedia Subsystem described in 4.2.1 and elected as a representative use case). The proposed framework allows to: *i)* handle an automated generation of stochastic models on the basis of some input parameters (e.g., MTTF, MTTR); *ii)* manage the transformation process in charge of translating high-level user specifications into templates feeding TimeNET, a powerful tool for Petri nets evaluation (freely downloadable from [101]); *iii)* compute the availability of the whole vIMS and of the relevant subsystems.

4.3.1 SFC model generation

The proposed tool is realized from-the-scratch, and relies on the Eclipse Modeling Framework (EMF) [102], which provides runtime support to produce Java code starting from a meta model. Moreover, a specific plug-in for invoking the TimeNET tool has been realized in order to evaluate SRN models. In Fig.4.19 we show the workflow representative of the realized tool, whose individual modules are described in the following subsections.

4.3.2 Network Service Module

The module EMF-NS allows to model, through EMF, the "skeleton" of an SFC composed of at least two Virtualized Network Functions (VNFs) connected in series. Such a series arrangement

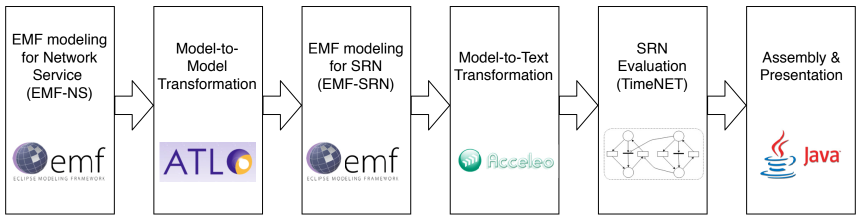

EMF modeling for Network Service (EMF-NS)
emf
Model-to-Model Transformation
ATL
EMF modeling for SRN (EMF-SRN)
emf
Model-to-Text Transformation
Acceleo
SRN Evaluation (TimeNET)
Assembly & Presentation
Java

Figure 4.19: Workflow of the realized tool.

is often known as Network Service (NS). By means of easy drag-and-drop operations, any user (basic or advanced) can setup the desired chain. At this stage, users can set system parameters such as: MTTF/MTTR, maximum number of replicas to use for redundancy, k-out-of-n values, and others.

4.3.3 Model-to-Model Transformation

In composing an NS, each single VNF needs to be mapped into a corresponding SRN. Such an operation is performed via the Atlas Transformation Language (ATL), a model-to-model transformation language often adopted in Model-Driven Engineering (MDE), which allows to produce target models from a set of source models by means of specific semantic rules that we have designed and customized for our domain.

4.3.4 SRN Module

This module (EMF-SRN) embeds a meta model needed to define the SRN described in Section 4.2.2, and is based on the definition of classes and attributes according the standard description provided by the UML class diagram, as specified in Fig. 4.20. The root class is *SRN* (top of Fig. 4.20) with an attribute *name* of type *EString*. Classes *Element* and *Function* are abstract (grey background) and can be specialized through *Arc/Node* and *RewardFunction/GuardFunction*, respectively. Classes such as *InhibitorArc* and *NormalArc* inherit functionalities from the *Arc* class.

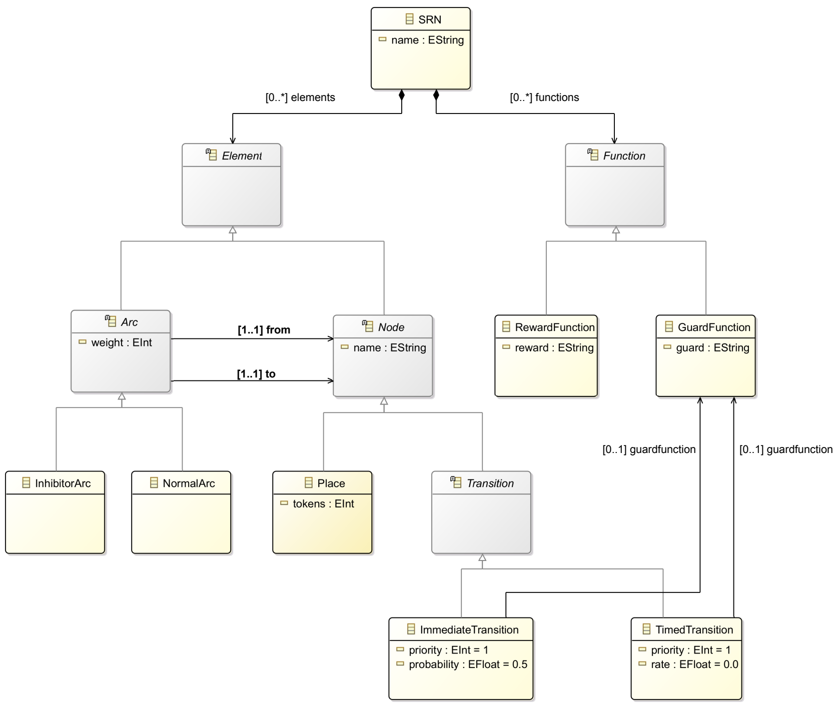


Figure 4.20: UML class diagram (EMF-based) of the SRN.

4.3.5 Model-to-Text Transformation

This module operates a transformation from a model to text by using Acceleo [103], a tool adhering to the MOFM2T (Meta-Object Facility Model-to-Text) standard. In particular, the model is translated into textual code readable by TimeNET[2].

4.3.6 SRN Evaluation

TimeNET [94] is a tool adopted for the modeling and analysis of stochastic Petri nets, including SRNs. We have designed a wrapper which allows to feed TimeNET with textual information embedding system parameters (MTTF/MTTR etc.) extracted dur-

[2]This module is fully customizable if one is interested to interact with different availability evaluation tools.

ing the EMF-SRN step. In practice, the invoked tool evaluates the steady-state availability associated to the SRN representative of a vIMS node by solving (4.4).

4.3.7 Assembly & Presentation

This module has a double function: *i)* it is in charge of evaluating the availability of the whole SFC starting from the steady state availability of a single node expressed by (4.4) (Assembly); *ii)* it gives to the final user results either in tabular and in graphical forms with the support of Java-based utilities (Presentation). As regards the Assembly sub-module, a clarification is needed. Actually, TimeNET does not provide a simple way to evaluate the availability of a system made of components arranged in series (vIMS nodes) and in parallel (redundant replicas of vIMS nodes). Accordingly, we move such functionalities into the Assembly sub-module which, starting from the availability value associated to each SRN (namely, each vIMS node), builds an RBD series/parallel representation of the whole vIMS and evaluates the final availability. We want to remark that the number of the series/parallel vIMS configurations can be very high. Unfortunately, TimeNET does not provide an automated mechanism to evaluate "one shot" the availability of all the output configurations. Thus, the Assembly module has been designed to automate such a process. Finally, the overall availability of the vIMS is quickly obtained by multiplying single availabilities derived from (4.4). Starting from the obtained configurations, the Presentation sub-module is in charge of performing a user-guided exhaustive search to propose just those vIMS configurations whose availability value satisfies a certain requirement (e.g. "five nines").

4.3.8 Graphical Appearance

The realized tool is equipped with a GUI for either basic or advanced users. Figure 4.21 shows the main screen where, through drag-and-drop operations, a user can build a three-layered node

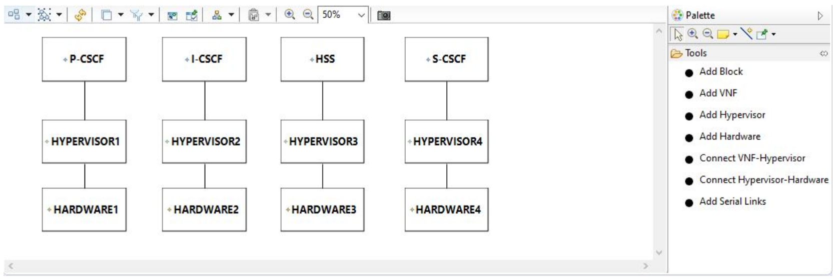
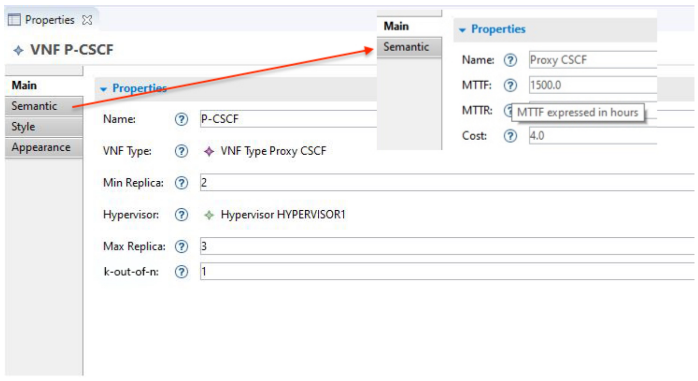


Figure 4.21: Building an SFC (virtualized IMS in our example) via drag-and-drop operations.

and connect via links the nodes to form an SFC. It is interesting to note that there is the possibility of customizing each layer by accessing the property tab (see, for instance, Fig. 4.22). Here it is possible to set/modify some parameters of each layer. A generic user is authorized to access the following fields:

- *Name*: name of the VNF;
- *VNF Type*: type of pre-charged VNFs (e.g. P-CSCF, HSS, etc.);
- *Min Replica*: minimum number of node replicas to deploy. Deploying more replicas implies more redundancy (and, hence, availability) but higher costs;
- *Max Replica*: maximum number of node replicas to deploy;
- *Hypervisor*: hypervisor type on which deploy the chosen VNF;
- *k-out-of-n*: this parameter is set when a node is required to work if and only if at least k of n replicas work.

By contrast, the VNF Type Properties Tab (see red arrow in Fig. 4.22) allows to access some layer parameters such as MTTF, MTTR, Layer Cost. It is also possible to load templates of various layers: a user can be interested in differentiating, for example,


Properties
VNF P-CSCF
Main
Semantic
Style
Appearance
Properties
Name: P-CSCF
VNF Type: VNF Type Proxy CSCF
Min Replica: 2
Hypervisor: Hypervisor HYPERVISOR1
Max Replica: 3
k-out-of-n: 1

Main
Semantic
Properties
Name: Proxy CSCF
MTTF: 1500.0
MTTR: MTTF expressed in hours
Cost: 4.0

Figure 4.22: Properties setting.

Hypervisor types (e.g. VMware, Virtualbox, etc.) on the basis of MTTF/MTTR and other specific parameters. Again, a dedicated panel (Figure 4.23) reports the underlying SRN as presented in 4.2.2, so that an advanced user can modify it. Let us now consider the results shown by the tool. Once a user has built the desired SFC (a vIMS in our example), the tool evaluates the availability of all configurations which can be very high, since each vIMS node can have a different number of replicas lying in the interval [*Min Replica*, *Max Replica*]. Figure 4.24 shows the visual outputs produced by the tool where the user decided to visualize only a limited set of configuration satisfying the HA requirement. Precisely, Fig. 4.24(a) reports a bar graph of the steady-state Unavailability (1 - Availability) of 4 exemplary configurations (S_1,...,S_4).

More details are shown in the corresponding tabular output (see Fig. 4.24(b)), where the exact value of availability is shown in the third column of the table, whereas the composition of configurations is reported in the second column (Configuration Type). For instance, the notation in the form HSS|2|3 indicates that a virtualized HSS node has been considered with $k = 2$ and $n = 3$.

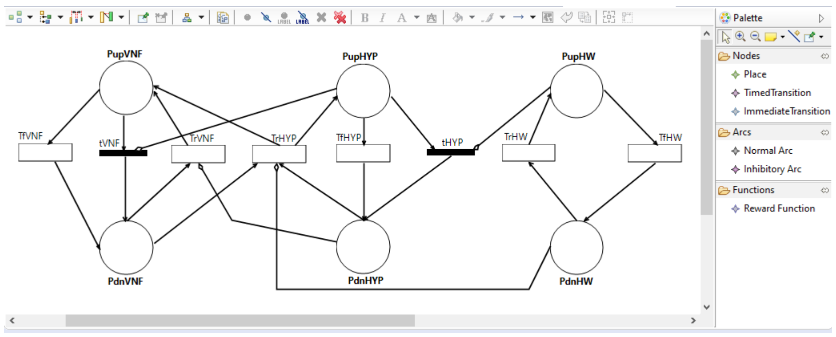
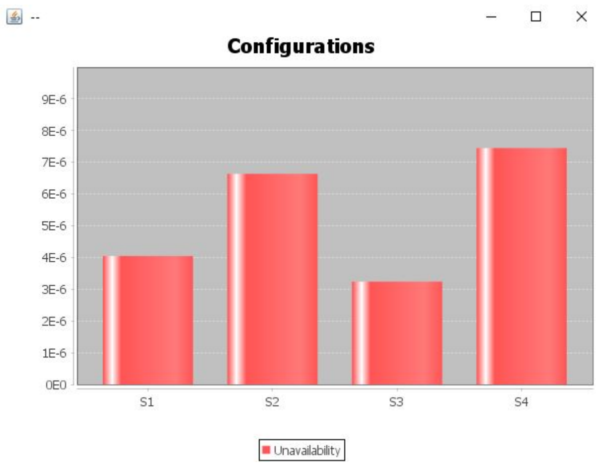
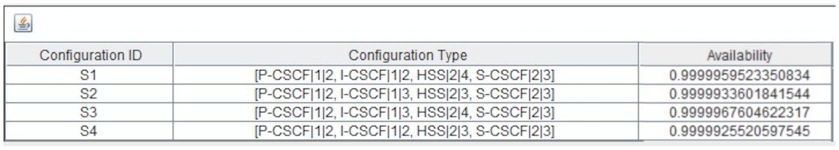


Figure 4.23: SRN building GUI.



(a) Graphical Output.



Configuration ID	Configuration Type	Availability
S1	[P-CSCF|1|2, I-CSCF|1|2, HSS|2|4, S-CSCF|2|3]	0.9999959523350834
S2	[P-CSCF|1|2, I-CSCF|1|3, HSS|2|3, S-CSCF|2|3]	0.9999933601841544
S3	[P-CSCF|1|2, I-CSCF|1|3, HSS|2|4, S-CSCF|2|3]	0.9999967604622317
S4	[P-CSCF|1|2, I-CSCF|1|2, HSS|2|3, S-CSCF|2|3]	0.9999925520597545

(b) Tabular Output.

Figure 4.24: Outputs produced by the tool.

Chapter 5

Conclusions

This thesis focused on statistical methods to characterize, mitigate, and counteract distributed network attacks. When coping with distributed network attacks, two main stages/issues emerge manifestly.

The first issue pertains to the design of suitable strategies aimed at revealing and tracking the distributed threats, which typically act sneakily to compromise the resources of a network target. The second issue concerns the precautionary countermeasures that a network manager can put in field to mitigate the damages that a threat can provoke. Accordingly, the contributions offered in this thesis moved along the two aforementioned lines as detailed in the following. The first contribution regarded the formalization of a novel class of application-layer (or L7) DDoS attacks. Under this model, a set of legitimate (thus, not suspicious) messages are collected within an emulation dictionary, and exploited by a botnet (a network of malicious nodes) to overwhelm the resources of a target (e.g. a web server). In this novel and challenging setting, we encounter different clusters (groups of bots) which share some parts of the emulation dictionary, giving to the attacker the possibility of introducing one more confusion element, thus having a greater chance of passing unnoticed through a defense system. In order to overcome this issue, this thesis provides a designed-from-scratch algorithm that, under reasonable conditions, identifies correctly the clusters of bots. Moreover, our analysis and experimental campaign identified practical scenarios where the individual clusters are not detectable, but, remarkably, our algorithm continues to guarantee (asymptotically) a correct discrimination of malicious nodes from legitimate nodes.

The second contribution pertains to the formalization of network availability and performability problems and the resulting design of suitable prevention strategies. Specifically, the focus is on the so-called "virtualized" networks, where each element can be logically split in different (and interacting) parts such as software, hypervisor, hardware. The sophisticated interconnections between such parts are captured through the Stochastic Reward Network (SRN) formalism. Such a formalism allows to model each "virtualized" element as a state-space system, and to examine its dynamical evolution under randomly distributed failure and repair events. The prevention strategy is then recast as an optimization problem, whose solution leads to the optimal trade-off between redundancy and costs.

As concerns the characterization of the novel DDoS setting, several questions are open, which could lead to useful extensions. For example, the identification algorithm can be totally parallelized (since it loops over the network users), but its complexity is quadratic in the network size. A useful extension would be to devise different algorithms to reduce the computational burden while still ensuring efficient botnet identification. Another relevant issue pertains to the collection of real-world malicious traffic traces. As it is well known, these traces are not publicly available, or are available under limited permissions that do not allow to extract sufficient information to test the pertinent algorithms (to the best of our knowledge, L7-DDoS real-world attacks with accessible message content are currently not available). Finally, the application of the Generative Adversarial Network (GAN) framework to L7-DDoS attacks with emulation dictionaries seems to be a new and challenging open problem that could be worth pursuing. In particular, relating the attacker/defender cost-performance trade-off corresponding to the GAN equilibrium, to the cost-performance trade-off pertaining to our scheme would constitute a significant research advance.

For that concerns the availability analysis, our techniques can be surely applied to other IMS deployments that, for instance, might include: co-location of multiple network nodes, more sophisticated interconnections among the involved elements, time-varying load requirements as demanded by contemporary Service Level Agreements. Other hints for future research stem from the consideration that the methods presented here can be adapted, for the benefit of service management organizations, to other infrastructures exhibiting a chained arrangement. Among modern networking systems, examples include: virtualized Evolved Packet Core (vEPC) nodes able to intervene in a service chain thanks to the SDN paradigm; dedicated SFCs (virtualized or containerized) composed, for instance, by firewalls, load balancers, IDSs built as chained resources in virtual data centers; virtualized Mobility Management Entities (vMMEs) whose signaling flow is organized in a chained fashion.

Appendix A

Appendix

In the following, the symbol $o(g_n)$ will denote a function such that $o(g_n)/g_n \to 0$ as $n \to \infty$. Also, when convenient for notational reasons, the expectation of X is denoted by $\bar{X}$.

For clarity of presentation, in the statement of Theorem 1 we deemed it useful to consider the symmetric scenario where $\alpha_i = \alpha_j = \alpha$. Actually, the MIR corresponding to pairwise cluster interactions can be characterized in the more general setting where α_i is not necessarily equal to α_j. For this reason, in this appendix we state and prove the following more general version of Theorem 1 that encompasses such asymmetric case.

Theorem 1a (Pairwise MIR with arbitrary α_i and α_j)**.** *Let $\mathcal{B}_i$ and $\mathcal{B}_j$ be subnets consisting of bots belonging to cluster i and cluster j, respectively. Assume that the transmission policies of all bots are either deterministic and synchronous (i.e., all bots transmit at regular intervals) or governed by independent Poisson processes (with rates that are allowed to vary across bots). Then, the (limiting) MIR of the joint subnet $\mathcal{B}_i \cup \mathcal{B}_j$ is (the symbol $\xrightarrow{\text{m.s.}}$ denotes mean-square convergence as $t \to \infty$):*

$$\begin{aligned}\widehat{\rho}_{\mathcal{B}_i\cup\mathcal{B}_j}(t) \quad &\xrightarrow{\text{m.s.}} \quad \rho_{\mathcal{B}_i\cup\mathcal{B}_j} = \mathscr{R}(\alpha_i\omega_{ij}, \omega_{ij}\lambda_{\mathcal{B}_i} + \omega_{ji}\lambda_{\mathcal{B}_j}) \\ &+ \quad (1-\omega_{ij})\,\mathscr{R}(\alpha_i, \lambda_{\mathcal{B}_i}) \\ &+ \quad (1-\omega_{ji})\,\mathscr{R}(\alpha_j, \lambda_{\mathcal{B}_j}). \qquad \text{(A.1)}\end{aligned}$$

Proof of Theorem 1a. Without loss of generality, we will denote the two clusters under analysis with indices 1 and 2. The emulation dictionaries of the two clusters can be decomposed as:

$$\mathscr{E}_1(t) = \widetilde{\mathscr{E}}_1(t) \cup \mathscr{E}_{12}(t),\ \mathscr{E}_2(t) = \widetilde{\mathscr{E}}_2(t) \cup \mathscr{E}_{12}(t), \tag{A.2}$$

where $\mathscr{E}_{12}(t)$ is the dictionary part shared between the two clusters, whereas $\widetilde{\mathscr{E}}_1(t)$ and $\widetilde{\mathscr{E}}_2(t)$ are the unshared parts, with:

$$\widetilde{\mathscr{E}}_1(t) \cap \widetilde{\mathscr{E}}_2(t) = \emptyset. \tag{A.3}$$

Since each bot picks messages from the dictionary pertaining to the cluster it belongs to, the empirical dictionary of the overall network $\mathcal{B}_1 \cup \mathcal{B}_2$ can be decomposed as:

$$\mathscr{D}_{\mathcal{B}_1 \cup \mathcal{B}_2}(t) = \widetilde{\mathscr{D}}_1(t) + \widetilde{\mathscr{D}}_2(t) + \mathscr{D}_{12}(t), \tag{A.4}$$

where $\widetilde{\mathscr{D}}_1(t)$ and $\widetilde{\mathscr{D}}_2(t)$ contain the (distinct) messages picked respectively from $\widetilde{\mathscr{E}}_1(t)$ and $\widetilde{\mathscr{E}}_2(t)$, while $\mathscr{D}_{12}(t)$ contains the (distinct) messages picked from $\mathscr{E}_{12}(t)$. In view of (A.4), in order to compute the limiting MIR of $\mathcal{B}_1 \cup \mathcal{B}_2$, we can examine the evolution over time of the three empirical dictionaries. The most difficult case pertains to the intersection dictionary $\mathscr{D}_{12}(t)$. For this reason, we now focus on its time evolution. The proof for the other two dictionaries follows along the same lines.

We start with the synchronous scheduling, where all bots transmit at intervals of deterministic length equal to $\tau = 1/\lambda$, where λ is the transmission rate common to all bots. The resulting scheduling corresponds to a slotted system whose time evolution can be more conveniently described with discrete time index $n \geq 0$, yielding the corresponding discrete-time quantities:

$$\begin{aligned} &e_1(n) \triangleq |\mathscr{E}_1(n\tau)|,\ e_2(n) \triangleq |\mathscr{E}_2(n\tau)|, \\ &e_{12}(n) \triangleq |\mathscr{E}_{12}(n\tau)|,\ D_{12}(n) \triangleq |\mathscr{D}_{12}(n\tau)|, \end{aligned} \tag{A.5}$$

where we used capital letters to remark that $D_{12}(n)$ is random, whereas $e_1(n)$, $e_2(n)$ and $e_{12}(n)$ are modeled as deterministic sequences. We note in passing that, from (3.23), (3.24) and (A.5) we get:

$$\begin{aligned}
e_1(n) &= \alpha_1\,\tau\,n + o(n),\\
e_2(n) &= \alpha_2\,\tau\,n + o(n),\\
e_{12}(n) &= \alpha_{12}\,\tau\,n + o(n),\\
\frac{e_{12}(n)}{e_1(n)} &= \omega_{12} + o(1),\\
\frac{e_{12}(n)}{e_2(n)} &= \omega_{21} + o(1).
\end{aligned} \tag{A.6}$$

Let us now introduce the quantity:

$$\rho_{12} \triangleq \frac{\alpha_{12}\tau(B_1\omega_{12} + B_2\omega_{21})}{\alpha_{12}\tau + (B_1\omega_{12} + B_2\omega_{21})}, \tag{A.7}$$

where B_1 and B_2 denote the cardinality of $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively. We observe preliminarily that, from the orthogonality principle, we have the following decomposition:

$$\begin{aligned}
&\mathbb{E}\left[\left(\frac{D_{12}(n)}{n} - \rho_{12}\right)^2\right]\\
&= \mathbb{E}\left[\left(\frac{D_{12}(n) - \bar{D}_{12}(n)}{n}\right)^2\right] + \left(\frac{\bar{D}_{12}(n)}{n} - \rho_{12}\right)^2,
\end{aligned} \tag{A.8}$$

where $\bar{D}_{12}(n)$ is the expectation of $D_{12}(n)$. We will now show that both terms on the RHS in (A.8) vanish as $n \to \infty$, which would in fact imply mean-square convergence of the random sequence $D_{12}(n)/n$ to the deterministic value ρ_{12} in (A.7).

Let us start by proving that the sequence of expectations converge, namely, that $\bar{D}_{12}(n)/n \to \rho_{12}$. To this end, let $i \in \{1, 2\}$ denote the specific cluster under analysis. We recall that a message picked at time n is included in the empirical dictionary $\mathscr{D}_{12}(n\tau)$ only if: *i*) the message is picked from the intersection emulation dictionary $\mathscr{E}_{12}(n\tau)$; and *ii*) the message is not already present in the empirical dictionary $\mathscr{D}_{12}((n-1)\tau)$ corresponding to the previous time epoch. According to this description, the total number of messages not contained in $\mathscr{D}_{12}((n-1)\tau)$, picked during the n-th time slot by bots belonging to cluster $i = 1, 2$, are distributed according to a binomial random variable $M_i(n)$ with B_i trials and success probability (conditioned on the cardinality $D_{12}(n-1)$):

$$p_i(n) \triangleq \frac{e_{12}(n) - D_{12}(n-1)}{e_i(n)}, \quad i = 1, 2. \tag{A.9}$$

The (conditional) expectation and variance of $M_i(n)$ are accordingly:

$$\mathbb{E}[M_i(n)|D_{12}(n-1)] = B_i\, p_i(n), \tag{A.10}$$

and

$$\mathrm{VAR}[M_i(n)|D_{12}(n-1)] = B_i\, p_i(n)[1 - p_i(n)]. \tag{A.11}$$

Furthermore, we recall that to build $\mathscr{D}_{12}(n\tau)$ we must select all the *distinct* messages among the $M_1(n)$ and $M_2(n)$ picked by the bots in clusters 1 and 2. For this reason, the effective number of new messages $D_{12}(n) - D_{12}(n-1)$ is upper bounded by the sum $M_1(n) + M_2(n)$, namely,

$$D_{12}(n) \leq D_{12}(n-1) + M_1(n) + M_2(n). \tag{A.12}$$

By taking expectations and using (A.10), Eq.(A.12) can be cast in the form:

$$\begin{aligned}
\bar{D}_{12}(n) &\leq \bar{D}_{12}(n-1) + B_1 \frac{e_{12}(n) - \bar{D}_{12}(n-1)}{e_1(n)} \\
&+ B_2 \frac{e_{12}(n) - \bar{D}_{12}(n-1)}{e_2(n)} \\
&= \bar{D}_{12}(n-1) \left[1 - \frac{B_1}{e_1(n)} - \frac{B_2}{e_2(n)}\right] \\
&+ B_1 \frac{e_{12}(n)}{e_1(n)} + B_2 \frac{e_{12}(n)}{e_2(n)} \\
&= \bar{D}_{12}(n-1) \left[1 - \frac{1}{(\alpha_1 \tau)/B_1 n + o(n)}\right. \\
&- \left. \frac{1}{(\alpha_2 \tau)/B_2 n + o(n)}\right] \\
&+ B_1 \omega_{12} + B_2 \omega_{21} + o(1) \\
&= \bar{D}_{12}(n-1) \left[1 - \frac{1}{an + o(n)}\right] \\
&+ B_1 \omega_{12} + B_2 \omega_{21} + o(1),
\end{aligned} \tag{A.13}$$

where we set:

$$a = \frac{\alpha_1 \alpha_2 \tau}{\alpha_1 B_2 + \alpha_2 B_1}, \tag{A.14}$$

and observed that:

$$\frac{1}{(\alpha_1 \tau)/B_1 n + o(n)} + \frac{1}{(\alpha_2 \tau)/B_2 n + o(n)} = \frac{1}{an + o(n)}. \tag{A.15}$$

Using Lemma 1, from the final inequality in (A.13), and applying (3.24) and the definition in (A.7), straightforward algebra yields:

$$\limsup_{n \to \infty} \frac{\bar{D}_{12}(n)}{n} \leq \rho_{12}. \tag{A.16}$$

We now manage to prove the above inequality for the lim inf, with reversed inequality sign. To this end, we consider the following (uniform, but for rounding errors) partition of $\mathscr{E}_{12}(n\tau)$ into K

subsets, namely,[1]

$$\mathscr{E}_{12}(n\tau) = \bigcup_{k=1}^{K} \mathscr{E}_{12}^{(k)}(n\tau), \quad e_{12}^{(k)}(n) \triangleq |\mathscr{E}_{12}^{(k)}(n\tau)| \tag{A.17}$$

and

$$\left\lfloor \frac{e_{12}(n)}{K} \right\rfloor \leq e_{12}^{(k)}(n) \leq \left\lfloor \frac{e_{12}(n)}{K} \right\rfloor + 1. \tag{A.18}$$

In this partition, K is an integer that can be chosen arbitrarily (we will make next a choice convenient for our purposes).

Let us now partition the empirical dictionary according to the partition of the emulation dictionary, yielding:

$$\begin{aligned} \mathscr{D}_{12}(n\tau) &= \bigcup_{k=1}^{K} \mathscr{D}_{12}^{(k)}(n\tau), \quad D_{12}^{(k)}(n) \triangleq |\mathscr{D}_{12}^{(k)}(n\tau)|, \\ D_{12}(n) &= \sum_{k=1}^{K} D_{12}^{(k)}(n). \end{aligned} \tag{A.19}$$

Then, conditionally on $D_{12}^{(k)}(n-1)$, the probability that a bot belonging to cluster $i = 1, 2$ picks a message belonging to the particular subset $\mathscr{E}_{12}^{(k)}(n\tau) \setminus \mathscr{D}_{12}^{(k)}((n-1)\tau)$ is equal to:

$$p_i^{(k)}(n) \triangleq \frac{e_{12}^{(k)}(n) - D_{12}^{(k)}(n-1)}{e_i(n)}, \quad i = 1, 2, \tag{A.20}$$

where the dependence of the probabilities upon $D_{12}^{(k)}(n-1)$ has been suppressed for ease of notation. From (A.18), we can write the loose bound (for sufficiently large n):

$$p_i^{(k)}(n) \leq \frac{2}{K}. \tag{A.21}$$

[1]Since we are interested in the limiting behavior as $n \to \infty$, we can start the analysis from an initial time n_0 where at least one message is present in every $\mathscr{E}_{12}^{(k)}(n_0)$. This is tantamount to assuming that $e_{12}(0) \geq K$.

A key observation to obtain a lower bound on the expected number of new messages is that $D_{12}^{(k)}(n-1)$ increases by *at least* 1 whenever *at least* one bot picks a new message belonging to the k-th subset, which means:

$$\begin{aligned}&\mathbb{E}[D_{12}^{(k)}(n)|D_{12}^{(k)}(n-1)] \\ &\quad\geq\; D_{12}^{(k)}(n-1) + 1 - \left[1 - p_1^{(k)}(n)\right]^{B_1}\left[1 - p_2^{(k)}(n)\right]^{B_2}.\end{aligned} \tag{A.22}$$

By exploiting the Taylor series of $(1-p)^B$ in conjunction with (A.21), we see that there exist always n_K such that, for all $n > n_K$ we can write:

$$\begin{aligned}&1 - \left[1 - p_1^{(k)}(n)\right]^{B_1}\left[1 - p_2^{(k)}(n)\right]^{B_2} \\ &\quad\geq\; B_1 p_1^{(k)}(n) + B_2 p_2^{(k)}(n) - \frac{c}{K^2},\end{aligned} \tag{A.23}$$

for a suitable constant c. In view of (A.22), the last inequality implies that:

$$\begin{aligned}&\mathbb{E}[D_{12}^{(k)}(n)|D_{12}^{(k)}(n-1)] \\ &\quad\geq\; D_{12}^{(k)}(n-1) + B_1 p_1^{(k)}(n) + B_2 p_2^{(k)}(n) - \frac{c}{K^2}.\end{aligned} \tag{A.24}$$

Averaging over $D_{12}^{(k)}(n-1)$ and summing over $k = 1, 2, \ldots, K$, from (A.20) we obtain:

$$\begin{aligned}\bar{D}_{12}(n) \;&\geq\; \bar{D}_{12}(n-1) + B_1\frac{e_{12}(n) - \bar{D}_{12}(n-1)}{e_1(n)} \\ &+\; B_2\frac{e_{12}(n) - D_{12}(n-1)}{e_2(n)} - \frac{c}{K},\end{aligned} \tag{A.25}$$

which in view of (A.6), and reasoning as done in (A.13), gives:

$$\begin{aligned}\bar{D}_{12}(n) &\geq \bar{D}_{12}(n-1)\left[1-\frac{1}{an+o(n)}\right]\\ &+ B_1\omega_{12}+B_2\omega_{21}-\frac{c}{K}+o(1).\end{aligned}\tag{A.26}$$

Using Lemma 1 along with the definition of a in (A.14) we get:

$$\liminf_{n\to\infty}\frac{\bar{D}_{12}(n)}{n}\geq\frac{a}{1+a}\Big(B_1\omega_{12}+B_2\omega_{21}-\frac{c}{K}\Big).\tag{A.27}$$

Since K is arbitrary, we conclude that:

$$\liminf_{n\to\infty}\frac{\bar{D}_{12}(n)}{n}\geq\frac{a}{1+a}\Big(B_1\omega_{12}+B_2\omega_{21}\Big),\tag{A.28}$$

which using (A.7), (A.14) and (A.16), shows finally that:

$$\lim_{n\to\infty}\frac{\bar{D}_{12}(n)}{n}=\rho_{12}.\tag{A.29}$$

We have therefore proved that the second term on the RHS in (A.8) vanishes. Let us move on to examine the first term in (A.8). Since we have just established convergence of the expectation, to prove that the variance vanishes it is sufficient to show that: $\mathbb{E}[D_{12}^2(n)]/n^2\to\rho_{12}^2$. By virtue of (A.12) we can write:

$$\begin{aligned}&E[D_{12}^2(n)|D_{12}(n-1)]\\ &\leq\ D_{12}^2(n-1)\\ &+\ \mathbb{E}[(M_1(n)+M_2(n))^2|D_{12}(n-1)]\\ &+\ 2\,D_{12}(n-1)\mathbb{E}[(M_1(n)+M_2(n))|D_{12}(n-1)],\end{aligned}\tag{A.30}$$

which, using (A.10) and (A.11), by straightforward algebra can be written in the following form:

$$\begin{aligned}
&E[D_{12}^2(n)|D_{12}(n-1)] \\
&\leq \; D_{12}^2(n-1)\left\{1-\frac{2B_1}{e_1(n)}-\frac{2B_2}{e_2(n)}+o(1/n)\right\} \\
&+ \; 2D_{12}(n-1)\left\{B_1\omega_{12}+B_2\omega_{21}+o(1)\right\}+cost.
\end{aligned} \tag{A.31}$$

By introducing the quantity $v_n \triangleq \mathbb{E}[D_{12}^2(n)]/n$, and taking expectation w.r.t. $D_{12}(n-1)$, Eq. (A.31) can be rewritten as:

$$\begin{aligned}
v_n \;&\leq\; v_{n-1}\frac{n-1}{n}\left\{1-\frac{2B_1}{e_1(n)}-\frac{2B_2}{e_2(n)}+o(1/n)\right\} \\
&+\; 2\rho_{12}\left\{B_1\omega_{12}+B_2\omega_{21}\right\}+o(1),
\end{aligned} \tag{A.32}$$

where we used the fact that, as shown before, $\bar{D}_{12}(n-1)/n \to \rho_{12}$. Now, the first term appearing on the RHS in (A.32) can be represented as (the term $1/n$ appearing in the fraction $(n-1)/n$ must be correctly embodied):

$$v_{n-1}\left(\frac{1}{1-a'n+o(n)}\right), \tag{A.33}$$

where

$$a' \triangleq \frac{\alpha_1\alpha_2\tau}{\alpha_1\alpha_2\tau+2\alpha_1B_2+2\alpha_2B_1} = \frac{\alpha_{12}\tau}{\alpha_{12}\tau+2B_1\omega_{12}+2B_2\omega_{21}}, \tag{A.34}$$

where in the last equality we used (3.24). Applying now Lemma 1 to (A.32), we get:

$$\begin{aligned}\limsup_{n\to\infty}\frac{\mathbb{E}[D_{12}^2(n)]}{n^2} &= \limsup_{n\to\infty}\frac{v_n}{n}\\ &\le 2\rho_{12}(B_1\omega_{12}+B_2\omega_{21})\frac{a'}{1+a'}\\ &= \rho_{12}\frac{\alpha_{12}\tau(B_1\omega_{12}+B_2\omega_{21})}{\alpha_{12}\tau+B_1\omega_{12}+B_2\omega_{21}}=\rho_{12}^2.\end{aligned} \tag{A.35}$$

Using now the subadditivity of limit superior we can write:

$$\begin{aligned}&\limsup_{n\to\infty}\mathbb{E}\left[\left(\frac{D_{12}(n)-\bar{D}_{12}(n)}{n}\right)^2\right]\\ &\le \limsup_{n\to\infty}\frac{\mathbb{E}[D_{12}^2(n)]}{n^2}+\limsup_{n\to\infty}\left(-\frac{\bar{D}_{12}^2(n)}{n^2}\right)\le 0,\end{aligned} \tag{A.36}$$

where the last inequality follows from (A.35) and the fact that $\bar{D}_{12}(n)/n \to \rho_{12}$. We have finally shown that $D_{12}(n)$ converges to ρ_{12} in mean-square sense. Using similar (somehow simpler) calculations we can prove that:

$$\begin{aligned}\frac{|\widetilde{\mathscr{D}}_1(n\tau)|}{n} &\xrightarrow{\text{m.s.}} (1-\omega_{12})\frac{\alpha_1\tau}{\alpha_1\tau+B_1},\\ \frac{|\widetilde{\mathscr{D}}_2(n\tau)|}{n} &\xrightarrow{\text{m.s.}} (1-\omega_{21})\frac{\alpha_2\tau}{\alpha_2\tau+B_2}.\end{aligned} \tag{A.37}$$

Combining the results in (A.37) with the convergence of $D_{12}(n)/n$ we finally get the claim for the synchronous case.

As regards the Poisson case, we consider again the slotted system in (A.5), but for the fact that τ is now an arbitrarily *small* interval. Let A_1 and A_2 denote the number of transmission attempts in a single slot pertaining to cluster 1 and 2, respectively. These variables are Poisson random variables with expectations

$\bar{A}_1 = \lambda_{\mathbb{B}_1}\tau$ and $\bar{A}_2 = \lambda_{\mathbb{B}_2}\tau$. Since the $A_1 + A_2$ transmissions correspond to independent (also across clusters) choices of messages from the pertinent emulation dictionaries, for small τ the system behaves as if we had $A_1 + A_2$ synchronous bots, where A_1 and A_2 are now *random* quantities. As a result, in the Poisson case we need to modify slightly the previous proof to embody this randomness. For example, Eq. (A.13) should be modified by considering a random numbers of bots A_1 and A_2, which after taking expectations gives:[2]

$$\begin{aligned}\bar{D}_{12}(n) \quad &\leq \quad \bar{D}_{12}(n-1)\left(1 - \frac{\bar{A}_1}{e_1(n)} - \frac{\bar{A}_2}{e_2(n)}\right) \\ &+ \quad \bar{A}_1\frac{e_{12}(n)}{e_1(n)} + \bar{A}_2\frac{e_{12}(n)}{e_2(n)}. \qquad \text{(A.38)}\end{aligned}$$

A similar development can be carried out for (A.25), since, using the definition of a Poisson random variable, we can write, for any $p \in (0,1)$:

$$\mathbb{E}[(1-p)^A] = e^{-\bar{A}p} \leq 1 - \bar{A}p + (\bar{A}p)^2. \qquad \text{(A.39)}$$

Similar arguments apply to (A.32), implying that all the equations relevant to prove the desired convergence hold true with B_1 and B_2 replaced by $\bar{A}_1$ and $\bar{A}_2$, revealing that, for example:

$$\begin{aligned}\frac{D_{12}(n)}{n\tau} \quad &\to \quad \frac{\alpha_{12}\left(\bar{A}_1\omega_{12} + \bar{A}_2\omega_{21}\right)}{\alpha_{12}\tau + \bar{A}_1\omega_{12} + \bar{A}_2\omega_{21}} \\ &= \quad \frac{\alpha_{12}(\lambda_{\mathbb{B}_1}\omega_{12} + \lambda_{\mathbb{B}_2}\omega_{21})}{\alpha_{12} + \lambda_{\mathbb{B}_1}\omega_{12} + \lambda_{\mathbb{B}_2}\omega_{21}}. \qquad \text{(A.40)}\end{aligned}$$

□

Lemma 1 (Useful Recursion)**.** *Let $a, b > 0$, $n \in \mathbb{N}$, and let f_n be*

[2]We use independence between the scheduling proccess and the message-picking process, as well as the memoryless property of the Poisson process.

a nonnegative sequence such that:

$$f_n \leq f_{n-1}\left(1 - \frac{1}{an + o(n)}\right) + b + o(1). \tag{A.41}$$

Then:

$$\limsup_{n\to\infty} \frac{f_n}{n} \leq \frac{ab}{1+a}. \tag{A.42}$$

If the inequality in (A.41) is reversed, the constant b can be relaxed to be an arbitrary real number, and:

$$\liminf_{n\to\infty} \frac{f_n}{n} \geq \frac{ab}{1+a}. \tag{A.43}$$

Proof. See Proposition 1 and its corollary in [6]. □

Bibliography

[1] CLUSIT, "2020 Clusit Security Report." https://clusit.it/, accessed: 2020-09-12.

[2] "The Clearwater Project," http://www.projectclearwater.org/, accessed: 2020-09-12.

[3] N. de Sousa, D. Perez, R. Rosa, M. Santos, and C. E. Rothenberg, "Network service orchestration: A survey," *Computer Communications*, vol. 142, pp. 69–94, 2019.

[4] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Botnet in DDoS attacks: Trends and challenges," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2242–2270, 2015.

[5] VERIZON, "Application layer DDoS mitigation in action." https://vzmediaplatform.medium.com/application-layer-ddos-mitigation-in-action-ee73e2ee4075, accessed: 2021-01-22.

[6] V. Matta, M. Di Mauro, and M. Longo, "DDoS attacks with randomized traffic innovation: Botnet identification challenges and strategies," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1844–1859, 2017.

[7] M. Cirillo, M. Di Mauro, V. Matta, and M. Tambasco, "Application-Layer DDoS attacks with multiple emulation dictionaries," *accepted to ICASSP 2021 - IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021.

[8] ——, “Botnet Identification in DDoS Attacks with Multiple Emulation Dictionaries,” *under review - IEEE Transactions on Information Forensics and Security*, 2020.

[9] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, “Availability Assessment of IP Multimedia Subsystem in an NFV-based Environment,” in *The Tenth International Conference on Advances in Future Internet (AFIN 2018)*, 2018, pp. 3–7.

[10] ——, “IP Multimedia Subsystem in an NFV environment: availability evaluation and sensitivity analysis,” in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2018, pp. 1–6.

[11] ——, “IP Multimedia Subsystem in a containerized environment: availability and sensitivity evaluation,” in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 42–47.

[12] ——, “Performability Management of Softwarized IP Multimedia Subsystem,” in *NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020.

[13] ——, “Comparative Performability Assessment of SFCs: The case of Containerized IP Multimedia Subsystem,” *IEEE Transactions on Network and Service Management*, pp. 1–1, 2020.

[14] M. Di Mauro, G. Galatro, M. Longo, A. Palma, F. Postiglione, and M. Tambasco, “Automated Generation of Availability Models for SFCs: The case of Virtualized IP Multimedia Subsystem,” in *NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020.

[15] ——, “A Tool for Automated Generation of Availability Models for Service Function Chains,” in *European Safety and Reliability Conference (ESREL 2020)*, 2020.

[16] M. Barni and B. Tondi, “The source identification game: An information-theoretic perspective,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 450–463, 2013.

[17] A. Abrardo, M. Barni, K. Kallas, and B. Tondi, "A game-theoretic framework for optimum decision fusion in the presence of Byzantines," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1333–1345, 2016.

[18] M. Barni and B. Tondi, "Source distinguishability under distortion-limited attack: An optimal transport perspective," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2145–2159, 2016.

[19] M. Mardani, G. Mateos, and G. B. Giannakis, "Dynamic anomalography: Tracking network anomalies via sparsity and low rank," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 50–66, 2013.

[20] P. Venkitasubramaniam, T. He, and L. Tong, "Anonymous networking amidst eavesdroppers," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2770–2784, 2008.

[21] J. Kim and L. Tong, "Unsupervised and nonparametric detection of information flows," *Signal processing*, vol. 92, no. 11, pp. 2577–2593, 2012.

[22] S. Marano, V. Matta, T. He, and L. Tong, "The embedding capacity of information flows under renewal traffic," *IEEE transactions on Information Theory*, vol. 59, no. 3, pp. 1724–1739, 2013.

[23] Z. Liu, H. Jin, Y.-C. Hu, and M. Bailey, "Practical proactive DDoS-attack mitigation via endpoint-driven in-network traffic control," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1948–1961, 2018.

[24] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proceedings DARPA information survivability conference and exposition*, vol. 1. IEEE, 2003, pp. 303–314.

[25] L. Li, J. Zhou, and N. Xiao, "DDoS attack detection algorithms based on entropy computing," in *International Conference on*

Information and Communications Security. Springer, 2007, pp. 452–466.

[26] Jian Yuan and K. Mills, "Monitoring the macroscopic effect of DDoS flooding attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 4, pp. 324–335, 2005.

[27] Z. Liu, Y. Cao, M. Zhu, and W. Ge, "Umbrella: Enabling ISPs to offer readily deployable and privacy-preserving DDoS prevention services," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1098–1108, 2019.

[28] Y. Xiang, K. Li, and W. Zhou, "Low-rate DDoS attacks detection and traceback by using new information metrics," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 426–437, 2011.

[29] J. Luo, X. Yang, J. Wang, J. Xu, J. Sun, and K. Long, "On a Mathematical Model for Low-Rate Shrew DDoS," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 7, pp. 1069–1083, 2014.

[30] M. E. Ahmed, S. Ullah, and H. Kim, "Statistical Application Fingerprinting for DDoS Attack Mitigation," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1471–1484, 2019.

[31] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "Delving into internet DDoS attacks by botnets: characterization and analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2843–2855, 2018.

[32] A. Praseed and P. S. Thilagam, "DDoS attacks at the application layer: Challenges and research perspectives for safeguarding Web applications," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 661–685, 2019.

[33] "Layer 7 DDoS — blocking http flood attacks." http://blog.sucuri.net/2014/02/layer-7-ddos-blocking-http-flood-attacks.html, accessed: 2020-09-12.

[34] "Taxonomy of DDoS attacks." http://www.riorey.com/types-of-ddos-attacks/#attack-15, accessed: 2020-09-12.

[35] "Global DDoS Threat Landscape." https://www.incapsula.com/blog/ddos-global-threat-landscape-report-q2-2015.html., accessed: 2020-09-12.

[36] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[37] V. Matta, M. Di Mauro, and M. Longo, "Botnet identification in randomized DDoS attacks," in *24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 2260–2264.

[38] ——, "Botnet identification in multi-clustered DDoS attacks," in *25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 2171–2175.

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2014, pp. 2672–2680.

[40] Q. Yan, M. Wang, W. Huang, X. Luo, and F. Yu, "Automatically synthesizing DoS attack traces using generative adversarial networks," *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 3387–3396, 2019.

[41] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "Enhancing network intrusion detection classifiers using supervised adversarial training," *The Journal of Supercomputing*, vol. 76, pp. 6690–6719, 2020.

[42] G. Gursun, M. Sensoy, and M. Kandemir, "On Context-Aware DDoS Attacks Using Deep Generative Networks,," in *27th International Conference on Computer Communication and Networks (ICCCN)*, 2018, pp. 1–9.

[43] J. Charlier, A. Singh, G. Ormazabal, R. State, and H. Schulzrinne, "Syngan: Towards generating synthetic network attacks using gans," *CoRR*, vol. abs/1908.09899, 2019. [Online]. Available: http://arxiv.org/abs/1908.09899

[44] R. Ghosh, F. Longo, F. Frattini, S. Russo, and K. S. Trivedi, "Scalable Analytics for IaaS Cloud Availability," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 57–70, 2014.

[45] B. Haverkort, R. Marie, G. Rubino, and K. Trivedi, *Performability Modelling: Techniques and Tools.* John Wiley & Sons, 2001.

[46] K. Nagaraja, G. Gama, R. Bianchini, R. P. Martin, W. Meira, and T. D. Nguyen, "Quantifying the performability of cluster-based services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 5, pp. 456–467, 2005.

[47] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2019.

[48] P. Nguyen and L. B. Le, "Joint Computation Offloading, SFC Placement, and Resource Allocation for Multi-Site MEC Systems," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.

[49] Z. Wang, J. Zhang, T. Huang, and Y. Liu, "Service Function Chain Composition, Placement, and Assignment in Data Centers," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1638–1650, 2019.

[50] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. Duarte, "Orchestrating Virtualized Network Functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 725–739, 2016.

[51] M. Barcelo, J. Llorca, A. M. Tulino, and N. Raman, "The cloud service distribution problem in distributed cloud networks," in

2015 IEEE International Conference on Communications (ICC), 2015, pp. 344–350.

[52] Y. Lin, T. He, S. Wang, K. Chan, and S. Pasteris, "Looking Glass of NFV: Inferring the Structure and State of NFV Network From External Observations," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1477–1490, 2020.

[53] X. Zhang, C. Wu, Z. Li, and F. C. M. Lau, "Proactive VNF provisioning with multi-timescale cloud resources: Fusing online learning and online optimization," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[54] R. Matos, J. Dantas, J. Araujo, K. S. Trivedi, and P. Maciel, "Redundant eucalyptus private clouds: Availability modeling and sensitivity analysis," *Journal of Grid Computing*, vol. 15, no. 1, pp. 1–22, 2017.

[55] M. C. Bezerra, R. Melo, J. Dantas, P. Maciel, and F. Vieira, "Availability modeling and analysis of a vod service for eucalyptus platform," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2014, pp. 3779–3784.

[56] Z. Hong, M. Shi, and Y. Wang, "CTMC-Based Availability Analysis of Multiple Cluster Systems with Common Mode Failure," in *2016 IEEE Conf. on Applied Computing and Information Technology*. IEEE, 2016, pp. 394–396.

[57] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, 2014.

[58] W. Li and A. Kanso, "Comparing containers versus virtual machines for achieving high availability," in *2015 IEEE International Conference on Cloud Engineering*. IEEE, 2015, pp. 353–358.

[59] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[60] J. Liu, Z. Jiang, N. Kato, O. Akashi, and A. Takahara, "Reliability evaluation for NFV deployment of future mobile broadband networks," *IEEE Wireless Communications*, vol. 23, no. 3, pp. 90–96, 2016.

[61] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, and J. P. Jue, "Guaranteed-availability Network Function Virtualization with Network Protection and VNF replication," in *GLOBECOM - 2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[62] S. Sebastio, R. Ghosh, and T. Mukherjee, "An availability analysis approach for deployment configurations of containers," *IEEE Transactions on Services Computing*, 2018.

[63] E. Andrade, B. Nogueira, R. Matos, G. Callou, and P. Maciel, "Availability modeling and analysis of a disaster-recovery-as-a-service solution," *Computing*, vol. 99, no. 10, pp. 929–954, 2017.

[64] J. K. Muppala, G. Ciardo, and K. S. Trivedi, "Stochastic Reward Nets for reliability prediction," in *Communications in Reliability, Maintainability and Serviceability*, 1994, pp. 9–20.

[65] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications.* New York, NY, USA: Wiley-Interscience, 1998.

[66] G. Davoli, W. Cerroni, C. Contoli, F. Foresta, and F. Callegati, "Implementation of service function chaining control plane through OpenFlow," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN).* IEEE, 2017, pp. 1–4.

[67] D. Borsatti, W. Cerroni, G. Davoli, and F. Callegati, "Intent-based Service Function Chaining on ETSI NFV Platforms," in *2019 10th International Conference on Networks of the Future (NoF).* IEEE, 2019, pp. 144–146.

[68] "Google cloud platform - container engine," https://cloud.google.com/container-engine/, accessed: 2020-09-12.

[69] "Amazon Elastic Container Service," https://aws.amazon.com/ecs, accessed: 2020-09-12.

[70] "ETSI Tech. Spec. 124 173 V15.2.0 (2018-09)," https://www.etsi.org/deliver/etsi_ts/124100_124199/124173/15.02.00_60/ts_124173v150200p.pdf, accessed: 2020-09-12.

[71] "Ericsson Techology Report: Real-time interaction in 5G - A use case example from the health care industry," https://www.ericsson.com/4a44a9/assets/local/digital-services/offerings/voice-services/health-care-case-real-time-interaction-in-5g-with/-ims-data-channel.pdf, accessed: 2020-09-12.

[72] "Huawei Techology Report: Vo5G Technical White Paper," https://www.huawei.com/it/industry-insights/technology/vo5g-technical-white-paper, accessed: 2020-09-12.

[73] J. Sun, G. Zhu, G. Sun, D. Liao, Y. Li, A. K. Sangaiah, M. Ramachandran, and V. Chang, "A Reliability-Aware Approach for Resource Efficient Virtual Network Function Deployment," *IEEE Access*, vol. 6, pp. 18 238–18 250, 2018.

[74] A. Shameli-Sendi, Y. Jarraya, M. Pourzandi, and M. Cheriet, "Efficient Provisioning of Security Service Function Chaining Using Network Security Defense Patterns," *IEEE Transactions on Services Computing*, vol. 12, no. 4, pp. 534–549, 2019.

[75] D. Cotroneo, R. Natella, and S. Rosiello, "NFV-Throttle: An Overload Control Framework for Network Function Virtualization," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 949–963, 2017.

[76] D. Cotroneo, L. De Simone, and R. Natella, "NFV-Bench: A Dependability Benchmark for Network Function Virtualization Systems," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 934–948, 2017.

[77] M. Di Mauro and A. Liotta, "Statistical Assessment of IP Multimedia Subsystem in a Softwarized Environment: A Queueing

Networks Approach," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1493–1506, 2019.

[78] C. Negus and W. Henry, *Docker Containers.* Prentice Hall Press, 2015.

[79] Y. Zhang, *Network Function Virtualization: Concepts and Applicability in 5G Networks.* John Wiley & Sons, 2018.

[80] "ITUT - Recommendation E.800," https://www.itu.int/rec/T-REC-E.800-200809-I, accessed: 2020-09-12.

[81] H. Chantre and N. L. da Fonseca, "Reliable broadcasting in 5G NFV-based networks," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 218–224, 2018.

[82] "NEC Virtualized Evolved Packet Core - vEPC," https://networkbuilders.intel.com/docs/vEPC_white_paper_w.cover_final.pdf, accessed: 2020-09-12.

[83] "Ericsson Review - Virtualizing network services - the telecom cloud," https://www.ericsson.com/4af606/assets/local/reports-papers/ericsson-technology-review/docs/2014/er-telecom-cloud.pdf, accessed: 2020-09-12.

[84] H. Jin, Y. Jin, H. Lu, C. Zhao, and M. Peng, "NFV and SFC: A Case Study of Optimization for Virtual Mobility Management," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2318–2332, 2018.

[85] G. Camarillo and M. Garcia-Martin, *The 3G IP Multimedia Subsystem*, 3rd ed. John Wiley and Sons, 2008.

[86] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Session Initiation Protocol (SIP)," http://www.ietf.org/rfc/rfc3261.txt, IETF RFC 3261.

[87] "Docker," https://www.docker.com, accessed: 2020-09-12.

[88] "CoreOS," https://coreos.com/rkt/, accessed: 2020-09-12.

[89] "OpenVZ," https://openvz.org, accessed: 2020-09-12.

[90] T. Combe, A. Martin, and R. Di Pietro, "To Docker or Not to Docker: A Security Perspective," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 54–62, 2016.

[91] "Amazon AWS Lambda," https://aws.amazon.com/lambda/, accessed: 2020-09-12.

[92] M. Ilyas and S. A. Ahson, *IP multimedia subsystem (IMS) handbook.* CRC press, 2018.

[93] A. Reibman, R. Smith, and K. Trivedi, "Markov and Markov reward model transient analysis: An overview of numerical approaches," *European Journal of Operational Research*, vol. 40, no. 2, pp. 257–267, 1989.

[94] R. German, C. Kelling, A. Zimmermann, and G. Hommel, "TimeNET: a toolkit for evaluating non-Markovian stochastic Petri nets," *Performance Evaluation*, vol. 24, no. 1–2, pp. 69–87, 1995.

[95] T. Telecom, "The LTE Data Storm in the Core of Your Network," *White Paper*, 2013.

[96] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability modeling and analysis of a virtualized system," in *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing*, 2009, pp. 365–371.

[97] ETSI, "Network Functions Virtualisation: An introduction, benefits, enablers, challenges and call for action," 2012.

[98] R. A. Sahner and K. S. Trivedi, "Reliability modeling using SHARPE," *IEEE Transactions on Reliability*, vol. 36, no. 2, pp. 186–193, 1987.

[99] M. Guida, M. Longo, F. Postiglione, K. S. Trivedi, and X. Yin, "Semi-Markov models for performance evaluation of failure-prone IP multimedia subsystem core networks," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 227, no. 3, pp. 290–301, 2013.

[100] M. L. Ayers, *Telecommunications System Reliability Engineering, theory, and practice.* Ed. John Wiley and Sons, 2012.

[101] "TimeNET project." https://timenet.tu-ilmenau.de/, accessed: 2020-09-12.

[102] "Eclipse Modeling Framework (EMF)," https://www.eclipse.org/modeling/emf/, accessed: 2020-09-12.

[103] "Acceleo project." https://www.eclipse.org/acceleo/, accessed: 2020-09-12.