Università degli Studi di Salerno

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione
Ciclo 30 – a.a 2016/2017

TESI DI DOTTORATO / PH.D. THESIS

Statistical Models for the Characterization, Identification, and Mitigation of Distributed Attacks in Data Networks

MARIO DI MAURO

SUPERVISOR: **PROF. MAURIZIO LONGO**

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

Dipartimento di Ingegneria dell'Informazione ed Elettrica
e Matematica Applicata

Contents

Chapter 1

Introduction and motivation

1.1 Threats evolution in modern networks

Nowadays, the proliferation of new network paradigms, novel communication protocols, ground-breaking technologies and infrastructures, provides new value to the digital information. Let's think of Internet of Things (IoT) paradigm allowing globe-wide disseminated objects to communicate among them. Let's think of the so-called 5G (fifth generation) platforms, such as Software Defined Networking (SDN) and Network Function Virtualization (NFV), conceived to boost the deployment of new IP-based services. Let's think of new multimedia protocols as WebRTC allowing to exploit a simple internet browser to set-up an audio/video session. All these innovations go hand in hand with security issues that become even more challenging and hard to face. As regards to the IoT ecosystem for example, the large volume of interconnected objects provides a big appeal for malicious users that could exploit the possible lack of strong security measures on IoT devices. Such devices, in fact, do not natively support advanced security mechanisms being designed with computational (and physical) constraints. It is worth noting that, at the end of 2016, a powerful and insidious malware called Mirai [1] infected a huge number of IoT devices (web cameras) by simply using telnet default passwords. As a second step, these cameras were instructed by Mirai to trigger a distributed attack against the french provider OVH that was flooded by 1.1 Tbps of network traffic, resulting in a service interruption. This kind of attack is often marked as Distributed Denial of Service (DDoS) attack that will be deepened within this work. New kinds of threats emerge also in the novel 5G platforms. The SDN infrastructure, where the data and control plane are decoupled, suffers of several vulnerabilities as the lack of authentication mechanisms in the data plane, triggerable CPU intensive operations on SDN switches, fraudulent memory space allocation on SDN devices [2]. Again, the NFV paradigm that introduces the virtualization technologies in telecommunication world, is prone to various kind of threats. In a Virtual Machine escape attack scenario, for example, a malicious user could take advantage of the common virtualization layer shared among more virtual machines (VMs); it is possible in this case to gain the access to the least protected VM and, by exploiting a weak isolation between VMs and hypervisor, to break this latter [3]. Finally, as regards to the novel multimedia protocols and applications, some classical network attacks have been conveniently tailored to tamper or interfere with VoIP/Video communications. Most dangerous threats in VoIP ecosystem are summarized in [4] where emerge: *i)* social threats, referring to vishing (namely, VoIP phishing), theft of service, unwanted spam; *ii)* eavesdropping, referring to a situation where a malicious user can unlawfully intercept (and eventually modify) the signaling or the content of a VoIP communication; *iii)* service abuse, covering the situation where service is offered in a commercial setting; *iv)* denial of service that involves VoIP infrastructures by causing resource consumption of vital nodes as SIP proxies, SIP registrars, Media Gateways. Other kinds of multimedia resources misuses concern the possibility of injecting illegal content into encrypted VoIP streams in order to circumvent any form of legal control. Accordingly, some statistical techniques

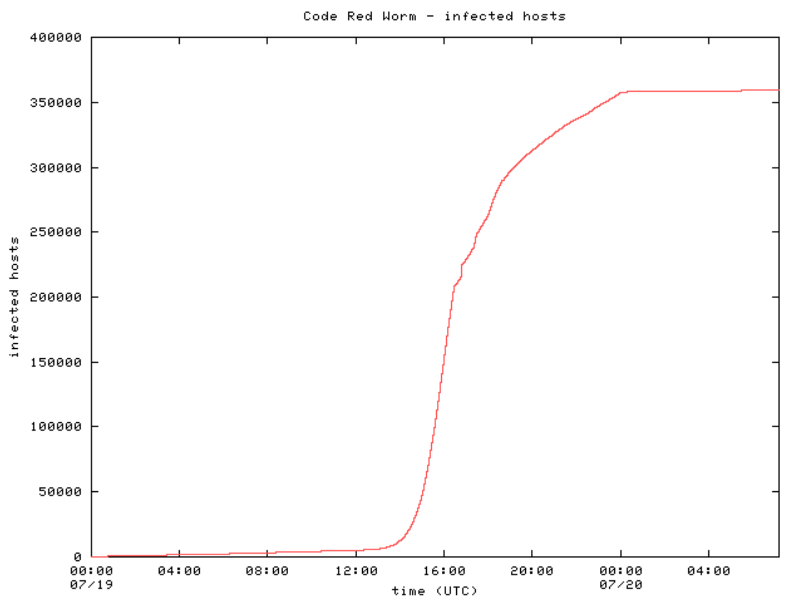


Figure 1.1: Observed Code Red propagation - number of infected hosts (from Caida.org).

have been exploited in order to reveal possibly dangerous real-time traffic [5–9]. In the following subsections I will introduce a macroscopic class of attacks that, due to its nature, is representative of one of the most evolved threats in modern communications: propagative and distributed attacks.

1.2 Taxonomy of propagative attacks

The propagation of a network threat is a common expression used in cybersecurity context to account for a particular process where a threat spreads across a network by infecting its neighborhood. The propagation (or diffusion) processes are frequently encountered in many disciplines as physics, material science, biology, chemistry, and they can emerge as naturally or artificially [10]. Such phenomena inspired a lot of hackers communities to create malwares able to mimic this kind of behavior. When propagation concepts apply to malware, they refer to diffusion processes where a malware starts to infect a source user (or a source users area) and then it propagates to other users that can become themselves a source of propagation. In the technical literature we find some interesting cases of malwares that exploit the propagation mechanisms. Let's see some examples. The worm *Code Red* [11] has been discovered in 2001 when it exploited a Windows IIS web server vulnerability. Each computer infected by Code Red started to generate one hundred of random simultaneous scans by seeking new possible victims. Routers that had to manage millions of fake scans, were prone to several disruptions as: large number of flows in NAT/PAT tables, high CPU utilization, large number of ARP requests or ARP storms in the network. A detailed study of Code Red propagation mechanism has been performed by CAIDA [1] that measured the number of infected hosts as function of time as depicted in Fig. 1.1. Few years later (2008), another worm called *Conficker* rapidly spread worldwide by exploiting a NETBIOS vulnerability in several Windows operating systems [12]. Once infected a computer, Conficker tried to install a listening http server and to wait for a client connection to download a copy of the worm in a DLL form. According to New York Times [13], this dangerous worm infected over 9 millions of computers. A very recent malware that combines the features of a ransomware and the features of a self-propagating worm, has been protagonist of worldwide attack in May 2017: *WannaCry*. Basically, the malicious software (eventually downloaded via phishing mails) is able to propagate inside a local area network by exploiting a Server Message Block (SMB) vulnerability of Microsoft systems. Once installed (and propagated inside the LAN), it behaves as a ransomware able to cipher data on computer's victims. To obtain the deciphering key, the victim has to pay about 300 dollars in bitcoins. Actually, when dealing with self-propagating threats, the users' (namely network nodes) behavior can depend on: *i)* the ability of malware of being dangerous and exploit peculiar vulnerabilities (e.g. operating system bugs) and, *ii)* the capability of user to be resilient to the at-

[1]Center for Applied Internet Data Analysis - www.caida.org

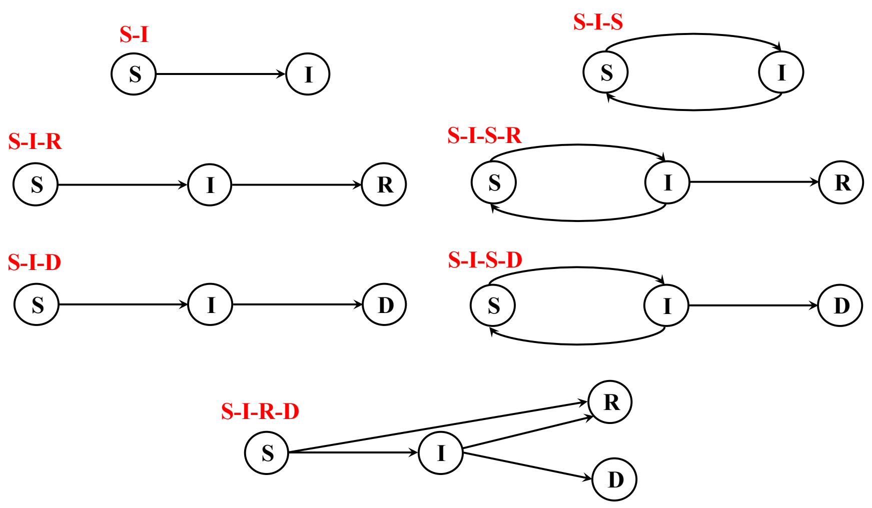


Figure 1.2: A subset of node infection models.

tack (presence of updated antivirus/firewall, advanced users skills, etc.). Inspired by epidemiological studies (and adopting the pertinent nomenclature) [14–16], it is possible to associate different states to the nodes: *susceptible* (S) indicates that a node is prone to be infected by a malware; *infected* (I) indicates that a node has been attacked by a malware; *removed* (R) indicates a successful recovery action; *dead* (D) indicates that a node has been completely broken by a malware. The four states can be variously combined in order to create different (and deterministic) infection models as represented in Fig. 1.2. The most basic combination of states generates the S-I model, where a susceptible node becomes infected and persists in the infected state up to the rest of its lifetime. When, after infection, a node gets back susceptible, we refer to a S-I-S model; in this case, there is no immunization upon recovery from infection and nodes become susceptible again. In case of a S-I-R model, instead, a node becomes immune after recovery stage. If a recovery stage is not possible, the node will die after the infection, and the resulting model is a S-I-D model. A combination of these basic configurations can generate models such as S-I-S-R or S-I-S-D. In the former case, a node can oscillate between susceptible and infected states but, at a certain time, a recovery stage will permanently cure the node. In the latter case, instead, no recovery action will be possible and the node will die. Another interesting combination is represented by S-I-R-D model. Such a model describes the behavior of a susceptible node that is infected by malware and passes to the infected state. Two transitions are possible: towards recovery state if the node receives the specific cure, and towards dead state if the node becomes destroyed. The transition between susceptible and removed state indicates a node that is conveniently patched and hence, protected. The Code Red worm is particularly suitable to be modeled with S-I-R-D paradigm.

1.3 Taxonomy of Distributed Denial of Service (DDoS) attacks

The main purpose of a traditional Denial of Service (DoS) attack is to create a service disruption. Classical examples refer to a web server overwhelmed by a huge amount of spurious requests, or an internet router that has to manage an enormous number of fake IP entries. This kind of attack is often categorized as *volumetric*, and, in many cases, it forces the target to be unavailable also for legitimate users. The distributed variant of DoS attack is called Distributed DoS (DDoS) and exploits basically the same kind of vulnerabilities and repetition schemes, except for the fact that the large request rate is now obtained by aggregating many small individual rates. Now, in the case of a classical DoS, the attacker machine generates a great bulk of traffic that allows an Intrusion Detection System (IDS) to possibly reveal the source of attack. In case of a DDoS, instead, it is practically impossible to keep track of every malicious host (often called *bot*) generating a tiny rate of fake requests. DDoS is a well structured and organized attack where a *botmaster* coordinates a set of bots called *botnet* to send attack packets to the victims. The damage depends on the botnet size that, if huge, can disrupt an entire portion of network in a very short time. Today, some darknets offer DDoS "services" allowing to rent a botnet of 100.000 bots with a total attack rate lying between 10Gbps and 100Gbps for about 100 dollars per hour. Many of (D)DoS attacks exploit the Network and the Transport levels of the TCP/IP stack, and classical examples include:

- **SYN flood attack**: a group of attackers try to break the three-way-handshake procedure characterizing the TCP protocol. In particular, the attackers send a succession of SYN requests to a victim system without waiting for the procedure were completed. The aim is to consume the victim's resources.

- **Smurf attack**: a group of attackers use the victim's IP address (IP *spoofing*) as sources to build ICMP messages sent to the broadcast address of an internet gateway. As result, the gateway forwards such ICMP messages to its network hosts that, in turn, answer to the source IP address (victim) making it flooded of unwanted responses.

- **Teardrop attack**: a group of attackers flood a target machine with malformed IP packets. More specifically, such packets are fragmented and sent with a wrong offset. During the reassembling stage, some operating systems crash when try to reassemble packets with skewed offsets.

Recently, the new class of application-layer DDoS (or L7-DDos) attacks is arising as one of the most powerful threats. Generally, L7-DDoS targets the HTTP protocol by building artificial GET and POST requests aimed at exhausting the web resources. Such attacks are not trivial to reveal for a series of reasons [17]: *i)* lower bandwidth consumption in comparison to network/transport layer DDoS attacks; *ii)* potentially infinite possibilities to build GET and POST requests with fake content; *iii)* obscurity due to the legitimate usage of TCP and UDP protocols; *iv)* facility of implementation because no tools for manipulating IP packets are needed. From a topological perspective, the DDoS attacks can

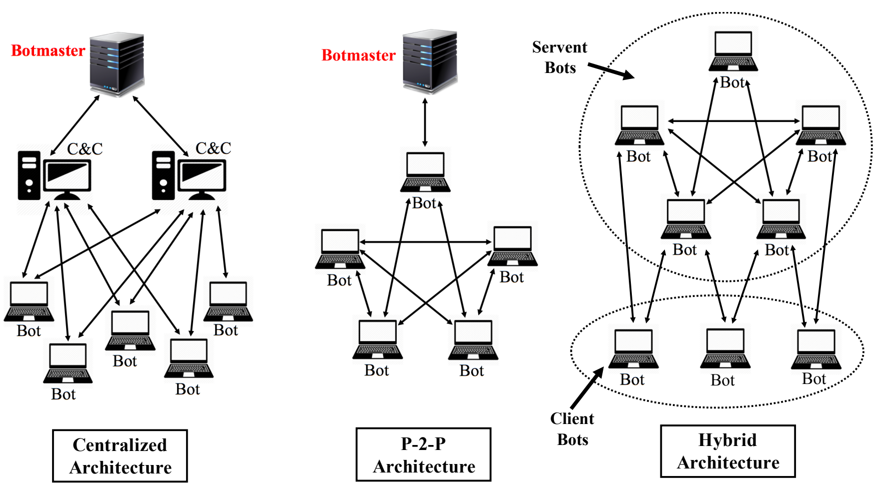


Figure 1.3: Botnet architectures and models.

be launched by relying on different botnet architectures. Some topological schemes are depicted in Fig. 1.3 where three kinds of architectures have been represented:

- **Centralized Architecture**: it is the most common of botnet architecture. Botmaster interacts (optionally) with botnet via command & control (C&C) servers, that propagate the instructions to bot agents. (C&C) servers can act either as load balancers, in case of a huge size of managed botnet, and as cluster-roots of a subset of bot agents, in case of an heterogeneous DDoS attack where different bot clusters have to be infected with different malwares.

- **P-2-P Architecture**: information about targets are shared among the bots. It provides a more resilient infrastructure because it avoids the problem of single point of failure present in the centralized solution.

- **Hybrid Architecture**: it has been designed to take advantage from both centralized and P2P architectures. Two kinds of bot agents are present: servant bots and client bots.

The former contain static and routable IP addresses, whereas the latter do the dirty work.

Due to the growing sophistication of DDoS attacks, either from a protocol point of view and from an architectural improvements perspective, the real-time detection of such attacks is becoming even more challenging. Actually, the adopted defense mechanisms try to use variously combined strategies in order to make the protection effective. Tools such as IDSs are often exploited to monitor DDoS threats by using misuse and anomaly detection approaches. According to the misuse approach, an IDS can be instructed with specific rules in order to track down suspicious combination of TCP flags and discover SYN flood or Teardrop attacks for example. On the contrary, the anomaly detection approach is useful to monitor anomalous network behaviors (e.g. traffic peaks, excessive bandwidth consumption), resulting in a deviation from regular operations. Unluckily, some DDoS attacks as L7-DDoS ones are able, in many cases, to provoke damages without possibility of being revealed.

1.4 Network resilience and availability

Propagative and distributed attacks rely on different strategies but have often the same purpose, namely, the fraudulent exhaustion of network resources. Since such massive attacks can be effective in a very short time, it is not always possible to design efficient algorithms able to counter them in a real-time fashion. Accordingly, before a security algorithm could properly intervene, the network should exhibit a certain degree of resilience in order to face (at least) the first stage of a massive attack. The network resilience refers to the capacity of a network infrastructure to resist to faults (e.g. network attacks) and to continue in providing a service. An useful metric able to capture the resilience features of a network is the availability, intended as a measure of the percentage uptime, considering the downtime due to faults. In particular, when dealing with mission critical systems, the concept of *high availability* is often introduced to ensure a certain level of performance for a higher than normal period. A measure of this condition is the number (or class) of "nines". For example, a "five nines" condition means that a system (e.g. a network) can be unavailable for only 5 minutes and 26 seconds throughout a year. The number of nines is often inserted in Service Level Agreements (SLAs) that a network provider subscribes, where planned and unplanned downtimes are considered. The former category refers to the scheduled operations of maintenance that a provider puts in field in order to guarantee a correctly working system. The latter category is obviously due to unpredictable events that can occur within a system, and it is not hard to realize that massive network attacks belong to such category. A way to ensure a certain level of high availability in a network infrastructure is to deploy some redundancy. Since redundancy policies have their own cost, good designing practices should be considered in order to guarantee a desired level of availability (often "five nines"), and to contain the operational costs at the same time.

1.5 Related research

In this section, a non-exhaustive *excursus* about relevant works concerning the characterization of propagative and distributed attacks will be exposed. Even if propagation and distribution of network threats are considered two sides of the same coin, the underlying models can be significantly different, thus, for the sake of clarity, they will be discussed separately. Therefore, a further discussion about pertinent related work concerning the availability issues of networks is proposed at the end of this section. As regards propagation and diffusion mechanisms, part of technical literature is devoting efforts to recast epidemic inspired models by applying new techniques or formalisms. For example, the authors in [18] propose an analytical framework able to catch the interactions among e-mails exchanged by infected users where the underlying scheme is represented by a Susceptible-Infected-Immunized

(S-I-I) model. Besides, such a model accounts for the problem of receiving multiple malware email copies from the same neighbor, by introducing the virtual nodes amenable to represent the n-th infection caused by users that open the n-th copy of an infected mail. A queueing-based methodology for modeling threat diffusion in complex networks has been devised in [19]. In particular, the proposed approach intends to capture the dynamics of SIS-based systems in time-varying networks, where nodes appear and disappear due to their energy reserves. An interesting patching mechanism able to immunize vulnerable nodes has been instead proposed in [20], where the intuition of the authors is to apply the propagative mechanisms to patching operations. More specifically, in a non-replicative patching setting, some nodes are preloaded with a patch that makes them immunized; in a replicative patching setting instead, each node can also forward the patch to the neighbor nodes, by relying on an action similar to the malware spreading. Again, a perspective about worm propagation in social environments is offered in [21]. Social worms are able to exploit the trusted relationships among friends, by spreading malicious code to neighbors. A new feature based on a *reinfection-notification* model seems to appear: an infected user re-transmits the malware as he/she reads the malicious message. Besides, this situation is amplified thanks to the mobility of users that can log in the same social account from different locations, causing the growth of infected nodes. In [22] the threat propagation is modeled through the Galton-Watson branching process [23], whose analytical properties are exploited to devise proper containment strategies. A generalized random Fibonacci model is instead proposed in [24], where effective strategies for tracking the infection process are devised, relying on powerful algorithms for filtering and prediction of epidemic outbreaks. As regards distributed attacks, technical literature is focusing on the novel DDoS attacks by trying to capture their multiform nature that makes them employable at each level of the TCP/IP protocol stack. The authors in [25], for example, propose to apply statistical methods based on computing entropy and frequency-sorted distributions of selected packet attributes to counter DDoS attacks. Consequently, the identification procedure is based on the detection of anomalies arising in the features of the packet attributes. In [26], two novel metrics, namely, the information distance metric and the generalized entropy metric, have been exploited to reveal low-rate DDoS attacks based on the dissimilarity between malicious and legitimate traffic. A particular DDoS variant called *shrew* distributed attack has been faced and mathematically modeled in [27]. Shrew attack tries to stealthy exploit some vulnerabilities in TCP's retransmission timeout (RTO) procedure. In particular, legitimate TCP flows are attacked by periodic high rate packet streams matching the RTO. Hence, as a TCP sender recovers from timeout, it will experiment congestion problems and will be forced to enter timeout again. More closely coupled to present research is the new class of application-layer DDoS attacks [28, 29], where the malicious traffic patterns are disguised as normal ones by leveraging the many possibilities offered at the application layer. Finally, as regards to the network availability problems, the interest of technical literature is recently growing also in accordance with newly conceived 5G infrastructures. Accordingly, the European Telecommunications Standard Institute (ETSI) has provided some guidelines about the application of reliability concepts to new generation networks [30] that have been considered in many works. In [31], some high availability concepts have been successfully applied to OpenStack platform, a cloud computing architecture implementing the Infrastructure-as-a-Service (IaaS) paradigm. Approaches based on Continuous-Time Markov Chains have been exploited in [32–34] to evaluate the system dependability. Since some serious issues arise when Markov chains are applied to complex systems due the intractable sizes of the state space, more evolved tools as Stochastic Reward Nets (SRNs) have been exploited to characterize network availability [35–37]. This last trend is in line with the research activity presented in this work.

1.6 Outline and main contribution

This thesis focuses on statistical approaches to model, mitigate, and prevent distributed network attacks. When dealing with distributed network attacks (and, more in general, with cyber-security problems), three fundamental phases/issues emerge distinctly. The first issue concerns the threat propagation across the network, which entails an "avalanche" effect, with the number of infected nodes increasing exponentially as time elapses. The second issue regards the design of proper mitigation strategies (e.g., threat detection, attacker's identification) aimed at containing the propagation phenomenon. Finally (and this is the third issue), it is also desirable to act on the system infrastructure to grant a conservative design by adding some controlled degree of redundancy, in order to face those cases where the attacker has not been yet defeated. The contributions of the present thesis address the aforementioned relevant issues, namely, propagation, mitigation and prevention of distributed network attacks. A brief summary of the main contributions is reported in the following. The current Chapter 1 is merely introductive, and offers a general overview about the topics considered in the present work.

Some mathematical and statistical tools are briefly exposed in Chapter 2, in order to help the reader to comfortably follow the dissertation. Along each proposed methodology, the candidate anticipates where it will be exploited within the document.

The first contribution is illustrated in Chapter 3, and concerns the adoption of Kendall′s birth-and-death process (1948) as an analytical model for threat propagation [38]. Such a model exhibits two main properties: *i)* it is a stochastic model (a desirable requirement to embody the complexity of real-world networks) whereas many models are purely deterministic; *ii)* it is able to capture the essential features of threat propagation through a few parameters with a clear physical meaning. By exploiting the remarkable properties of Kendall′s model, the exact solution for the optimal resource allocation problem (namely, the optimal mitigation policy) has been provided for both conditions of perfectly known parameters, and unknown parameters (with the latter case being solved through a Maximum-Likelihood estimator).

The second contribution is presented in Chapter 4, and pertains to the formalization of a novel kind of randomized Distributed Denial of Service (DDoS) attack. In particular, a botnet (a network of malicious entities) is able to emulate some normal traffic, by picking messages from a dictionary of admissible requests. Such a model allows to quantify the botnet learning ability, and to ascertain the real nature of users (normal or bot) via an indicator referred to as MIR (Message Innovation Rate). Exploiting the considered model, an algorithm that allows to identify a botnet (possibly) hidden in the network has been devised [39, 40].

Chapter 5 extends the contribution provided in the previous section, by broadening the view to a multi-clustered setting where the botnet is now spread across many clusters, each one of them having its own emulation dictionary. Although this part is intended as a further development of the one presented in Chapter 4, some original contribution emerges, such as the definition of specific rules useful to identify the malicious clusters and embodied in a new algorithm tailored to a multi-clustered scenario [41].

A third contribution is offered in Chapter 6, and concerns the formalization of the network availability problem and the consequent design of a prevention strategy. Two statistical frameworks are proposed to model the high availability requirements of network infrastructures, namely, the Stochastic Reward Network (SRN), and the Universal Generating Function (UGF) frameworks. In particular, since in the network environment dealing with multi-dimensional quantities is crucial, an extension of the classic UGF framework, called Multi-dimensional UGF (MUGF), is devised. Actually, this third contribution intends to offer a more generalized and enlarged view of the network availability issue (not exclusively intended as a prevention measure), also applied to some new generation paradigms (e.g. NFV, SDN) [5, 7, 8, 42–49].

Finally, Chapter 7 ends this thesis by drawing a brief summary along with considerations about possible future works.

Chapter 2

Classic Background

2.1 Continuous Time Markov Chains

In this section, the candidate wants to recap some classical background techniques that have been exploited in his research. Continuous Time Markov Chains (CTMCs) and their variants have been extensively used inside the present work. This representation is used to model a variety of phenomena, such as propagation and diffusion of threats (epidemics), queueing networks, reliability and availability of complex systems. A continuous-time stochastic process $\{X(t) : t \geq 0\}$ with discrete space $\mathcal{S}$ is a CTMC if for all $t \geq 0$, $s \geq 0$, $i \in \mathcal{S}$, $j \in \mathcal{S}$,

$$\begin{aligned} &P(X(s+t) = j | X(s) = i, \{X(u) : 0 \leq u < s\}) \\ &\qquad = (P(X(s+t) = j | X(s) = i) = P_{ij}(t) \end{aligned} \tag{2.1}$$

where $P_{ij}(t)$ is the transition probability from state i to state j during a time interval of t. The CTMCs have the following property: given that the process is in state i, the *sojourn time* in that state is an exponentially distributed random variable with a parameter λ_i. Since the exponential distribution is memoryless, the future outcome of the process depends only on the present state and does not depend on when the last transition occurred. Some useful properties and definitions about CTMCs follow.

- **Definition:** a probability distribution λ is a *limiting distribution* of a CTMC if

$$\lim_{t\to\infty} P_{ij}(t) = \lambda_j \qquad \forall i, j. \tag{2.2}$$

- **Definition:** a probability distribution π is a *stationary distribution* of a CTMC if

$$\pi = \pi P(t) \qquad \forall t \geq 0. \tag{2.3}$$

- **Definition:** a CTMC is *irreducible* if, given any two states i and j,

$$P(X(t) = j | X(0) = i) > 0. \tag{2.4}$$

  for some finite t.

- **Definition:** A state in a CTMC is said *recurrent* if the probability that the process will return to the same state is one. A recurrent state is said *positive-recurrent* if the expected return time is finite.

- **Stationary Transitions:** a CTMC has a *stationary transition* if

$$P_{ij}(s, s+t) = P_{ij}(0, t) = P_{ij}(t). \tag{2.5}$$

  In other words, P_{ij} only depends on the difference between $t+s$ and s, and $\{X(t)\}$ is referred to a homogeneous CTMC.

- **Ergodicity of a CTMC:** Let $\{X(t) : t \geq 0\}$ be an irreducible positive recurrent Markov chain. It follows that, if there exists a stationary distribution π it is unique, and

$$\lim_{t\to\infty} P_{ij}(t) = \pi_j \qquad \forall i, j. \tag{2.6}$$

2.2 Kendall's Birth-and-Death model

The *Birth-and-Death* (BD) process, formalized in a seminal work of Kendall [50], represents a particular case of CTMCs in which the state transitions are of two types: *i) births*, where the state variable increases by one unit, and *ii) deaths*, where the state variable decreases by one unit. The state diagram of a BD process is depicted in Fig. 2.1 where: in case of birth, we observe a transition from state n to state $n+1$, while, in case of death, we observe a transition from state n to state $n-1$. The process is characterized in terms of birth rates $\{\lambda_i\}_{i=0\ldots\infty}$ and death rates $\{\mu_i\}_{i=0\ldots\infty}$. In some cases it is useful to define a *pure* birth process, namely, a process where $\mu_i = 0$ $(\forall i \geq 0)$ and a *pure* death process, namely, a process where $\lambda_i = 0$ $(\forall i \geq 0)$. In order to formally define a BD, let $\{X(t)\}_{t\geq 0}$ a CTMC and dt a very short interval of time during which observable changes in the chain exist. Letting k the cardinality of a certain population (e.g. network nodes, particles, etc.), we want to evaluate the probability of a change occurring at time $t+dt$, by starting at time t. The probability of a birth in the interval $(t, t+dt)$, given $X(t) = k$, can be written:

$$P(X(t+dt) = k+1 | X(t) = k) = \lambda_k dt + o(dt). \tag{2.7}$$

Basically, it means that it is possible to neglect the probability of more than one birth event in a small time interval dt. Similarly, the probability of a death in the interval $(t, t+dt)$ can be written:

$$P(X(t+dt) = k-1 | X(t) = k) = \mu_k dt + o(dt), \tag{2.8}$$

with the obvious meaning that the probability of more than one death event in a small time interval dt is negligible. By grouping the two assumptions (probability of no birth or deaths occurring during $(t, t+dt)$), we can write:

$$P(X(t+dt) = k | X(t) = k) = 1 - (\lambda_k + \mu_k)dt + o(dt). \tag{2.9}$$

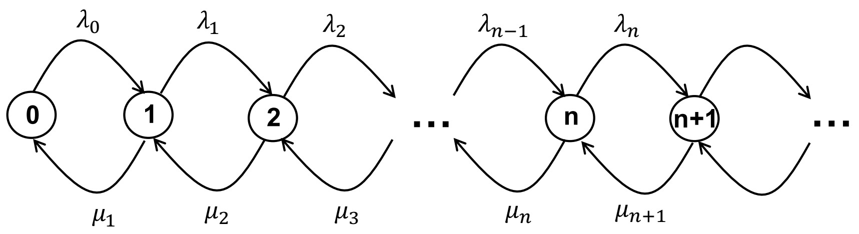


Figure 2.1: Birth-and-Death process. State diagram.

The BD process will be extensively used in Chapter 3 in order to propose a formal model of a malware propagation, where the infected nodes are interpreted in the sense of new births (malware's birth), whereas the cured nodes are interpreted in the sense of new deaths (malware's death).

2.3 The Poisson process

A homogeneous Poisson process can be defined as a pure birth process (see sec. 2.2) where $\lambda_k = \lambda$, $\mu_k = 0$ for all $k \geq 0$. We recall some useful properties of a Poisson process $\{X(t)\}_{t \geq 0}$:

- given arbitrary time instants $t_0 < t_1 < \cdots < t_n$ with $t_0 = 0$, the events occurring in disjoint intervals $X(t_1) - X(t_0), X(t_2) - X(t_1), \ldots X(t_n) - X(t_{n-1})$, are independent random variables. This Poisson process property is often called *independent increments* property;

- for $h \geq 0$ and $t > 0$, the number of events occurring in $(h, h+t)$ is represented by random variable $X(h+t) - X(h)$ that follows the Poisson distribution:

$$P(X(h+t) - X(h) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!} \tag{2.10}$$

- given a short time interval dt, from a stochastic perspective could be useful to evaluate the probability that exactly one event occurs over dt:

$$
\begin{aligned}
P(X(t+dt)-X(t)=1) &= \frac{(\lambda dt)^1 e^{-\lambda dt}}{1!} \\
= \sum_{n=0}^{\infty} \frac{(-\lambda dt)^n}{n!} &= \lambda dt + o(dt) \qquad (2.11)
\end{aligned}
$$

where $o(dt)$ indicates a term of smaller order than dt and where rate λ represents the proportionality constant in the probability that an event occurs during an arbitrary small interval dt.

The Poisson process, along with its properties, has been extensively exploited in this research: in Chapter 3, to model the counting process associated to sick nodes, in Chapters 4 and 5, to characterize the transmission scheduling of a botnet, in Chapter 6, to describe the failure and repair events associated to a multi-state performance model describing a resilient network infrastructure.

2.4 Generating Functions

The distribution of a random variable can be often characterized in terms of its Moment Generating Function (MGF). MGF can be used to comfortably derive the moments of a random variable, and, being able to uniquely determine the probability distribution of a random variable, it provides an amenable analytical tool to solve some practical problems. The MGF of a random variable X, is formally defined as:

$$M_X(t) = \mathbb{E}[e^{tX}], \qquad (2.12)$$

wherever the expectation exists. The MGF may be not defined for some real values of the argument t. If $M_X(t) < \infty$ at least in some open interval containing $t = 0$, then it is infinitely differentiable in this interval, and its n-th derivative at $t = 0$ represents the n-th moment of the random variable X. For the purposes of this work, the MGFs have been exploited in Chapter 3: for the Birth-and-Death model with immigration, in fact, it is possible to find a closed-form solution for the MGF (and, then, for the corresponding probability distribution). More specifically, it is rather straightforward to characterize the time evolution of the MGF through a first-order linear partial differential equation, by describing the dynamics of the infected nodes in a network. As a further application, an interesting extension of MGF called Universal Generating Function (UGF) has been proposed in [51], and recently resumed in [52]. UGF technique allows to find a system performance distribution based on the performance distributions of its elements, by exploiting simple algebraic procedures. From a mathematical viewpoint, the UGF of a discrete random variable X, is a polynomial-shape function $u(z)$ defined as:

$$u(z) = \sum_{j=1}^{J} p_j z^{x_j}, \tag{2.13}$$

where X has J values x_j, and $p_j = \Pr\{X = x_j\}$, for $j = 1, ..., J$. As follows from (2.13), the coefficient of z^{x_j} equals the probability that random variable X equals x_j. In other words, (2.13) represents the *pmf* (probability mass function) of the variable X. As regards the present work, UGF methodology has been used to find the best redundant, and hence, resilient network architecture as explained in Chapter 6. Besides, a multi-dimensional extension of UGF (called MUGF) useful to deal with availability of multi-operator networks has been devised as an original contribution.

2.5 Stochastic convergences

The concept of convergence is typically applied to a sequence of non-random numbers in order to analyze its asymptotic behavior. The extension of such a concept to a sequence of random variables is not so immediate because one cannot predict the asymptotic behavior of a random element. This notwithstanding, one can consider the convergence of a non-random sequence derived from the random one. Since it is possible to derive non-random sequences in many ways, it is possible to define different stochastic convergence concepts [53].

Proposition 1 (Convergence in Distribution)**.** *A sequence of random variables $\{X_n\}$ is said to converge in distribution to a random variable X if*

$$\lim_{n\to\infty} F_{X_n}(u) = F_X(u), \tag{2.14}$$

for each point u at which $F_X(u)$ is continuous, where $F_{X_n}(u)$ represents the distribution function of X_n, and $F_X(u)$ the distribution function of X. This kind of convergence is often denoted by $X_n \xrightarrow{\mathrm{d}} X$ and is the weakest form of convergence, since it is implied by all other types of convergences.

Proposition 2 (Convergence in Probability)**.** *A sequence of random variables $\{X_n\}$ is said to converge in probability to a random variable X if,* $\forall \epsilon > 0$

$$\lim_{n\to\infty} \mathbb{P}(|X_n - X| > \epsilon) = 0. \tag{2.15}$$

This kind of convergence is often denoted by $X_n \xrightarrow{\mathrm{p}} X$.

Proposition 3 (Almost sure Convergence)**.** *A sequence of random variables $\{X_n\}$ is said to converge almost surely to a random variable X if*

$$P(\lim_{n\to\infty} X_n = X) = 1. \tag{2.16}$$

This kind of convergence is often denoted by $X_n \xrightarrow{\mathrm{a.s.}} X$.

It is useful to remark that the following chain of implications holds: if $X_n \xrightarrow{\mathrm{a.s.}} X$, then $X_n \xrightarrow{\mathrm{p}} X$, then $X_n \xrightarrow{\mathrm{d}} X$.

Some stochastic convergences concepts have been exploited in Chapter 3 in order to characterize the asymptotic regime of node infection process, and in Chapter 4 when dealing with the botnet dynamics.

2.6 Stochastic Reward Networks (SRN)

Another useful application of CTMCs (and of BDs in particular) concerns the modeling of a system in terms of its reliability. More specifically, a complex system could be represented via Markov structures where a transition towards an *OFF* state regulated by a rate λ is interpreted as a failure, whereas, a transition towards an *ON* state regulated by a rate μ is interpreted as a repair. One major drawback of Markov models is that the size of chains tend to be very large when representing complex systems. To this aim, different formalisms as Stochastic Petri Nets (SPNs) have been introduced [54] to provide an high level interface for generating the underlying Markov model. Besides, SPN is useful to model some features of computer systems such as synchronization, concurrency, resource possession. An interesting variant of classical Markov models, is offered by Markov Reward Models [55] where a *reward rate* is attached to each state of Markov chain. Such reward rate is a non-negative random variable associated with certain conditions of the system, and its value is related to the particular measure (performance, dependability, availability etc.). Since MRMs (as classical CTMCs) suffer from the state space largeness, it is necessary to extend the SPN representation to a similar reward-based structure. Accordingly, the authors in [56] propose a Stochastic Reward Nets (SRNs) formalism providing a concise description of the underlying MRM.

By using SRN formalism, it is particularly straightforward to interpret the measures of interest as instances of expected reward rate. Given $X(t)$ a random variable representing the instantaneous reward rate, it is possible to write:

$$\mathbb{E}[X(t)] = \sum_{k \in \mathfrak{T}} r_k \pi_k(t), \tag{2.17}$$

where $\mathfrak{T}$ is the set of states, r_k the reward rate in state k, and π_k the probability of being in state k. The expression for the expected accumulated reward in an interval $[0, t)$ is given by:

$$\mathbb{E}[Y(t)] = \sum_{k \in \mathfrak{T}} r_k \int_0^t \pi_k(x) dx. \qquad (2.18)$$

Actually, the set of states is related to the concept of *marking* of an SRN that will be directly clarified in Chapter 6. SRNs adopt a comfortable graphical representation of a system where *places* and *transitions* account for states (e.g. off/on conditions) and events (e.g. failures, repairs) respectively. Additional details about such a graphical representation will be offered in Chapter 6 when dealing with the modeling of novel network infrastructures.

Chapter 3

Modeling network threats propagation

3.1 Problem statement

In this section, a general perspective of the faced problem is presented. In particular, the setting considered in this research is pictorially represented in Fig. 3.1. At a certain time, multiple threats crowd a data network. The network under attack is conveniently partitioned into N subnets, with each subnet being susceptible to a specific threat. Typically, sensitivity to different threats reflects into some degree of heterogeneity across distinct subnets, in terms of several types of attributes, e.g., in terms of constitutive elements (computers, laptops, mobiles), operating systems, protocols, and so on. Such heterogeneity implies, in particular, that the threat parameters must be subnet dependent. For each subnet, a security agency: *i)* gathers information (probably not in real time) regarding the number of infected nodes at successive time epochs; *ii)* aims at implementing proper countermeasures to mitigate the attack. The underlying mechanism governing threat propagation over networks follows a kind of cascade effect, where a primary source is aimed at infecting the closest nodes of other subnets. It is not hard to envisage interesting common points between such structure and the mechanism governing other prop-

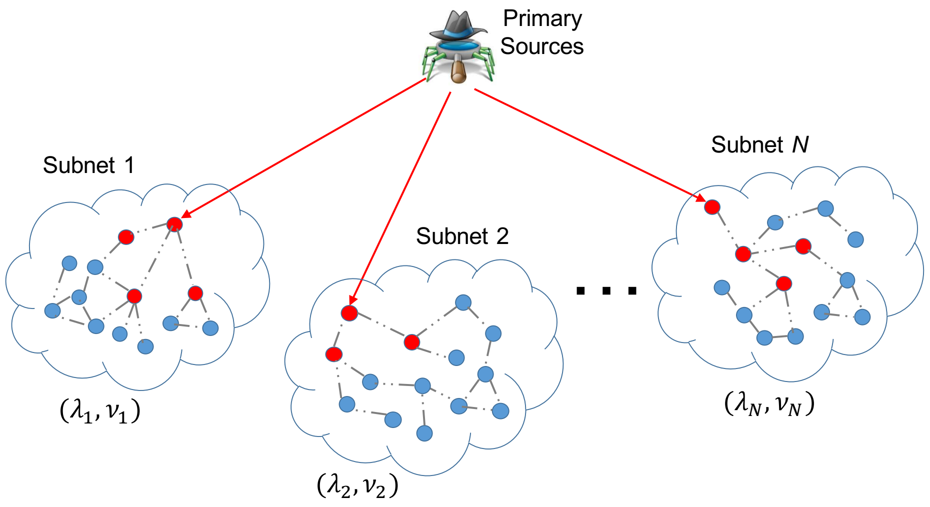


Figure 3.1: Representation of threat propagation mechanism. The *primary* source of infection continuously scans portions of network (N distinct subnets). When a vulnerable node is infected, it becomes a *secondary* source (red circle) and starts to seek for further vulnerable nodes within the pertinent subnet.

agative phenomena, such as the birth-death-immigration process ruling the growth of a population, as well as the virus propagation in epidemiological models. Accordingly, mathematical models inspired from other fields (e.g., biology, natural and/or social sciences) have been successfully employed to study the propagation of threats in the cyber-security domain [10, 14]. The earliest models were basically *deterministic*, where a system of differential equations was exploited to describe the evolution of the number of infected nodes in the observed population of individuals [15]. Then, such models have been generalized to *stochastic* models, in order to include the high variability present in typical real-world systems. One of the significant paradigms for stochastic threat propagation is the *queueing* paradigm, where, loosely speaking, nodes "arrive" (i.e., they get sick) following a certain arrival process, while nodes "depart" (i.e., they are cured) according to a certain departing process. It is very common to model the infection and the curing measures as *memoryless* processes, corresponding to say that the arrival process is a Poisson process and that the service times are exponentially distributed [57]. Even if the memoryless property could be only approximately met in real-world systems, it results into a classic and advantageous first-order approximation crucial to obtain tractable results and to capture the essential scaling laws of the system. Based on the particular features of the system under analysis (local as opposed to large, wireless as opposed to wired, and so on), several models allow representing the various characteristics of the system. There is one main conclusion that has been established as regards the threat diffusion over data networks. Several works [10, 58, 59], have confirmed the exponential growth of the infected nodes grows in the early stage of infection, namely, in the stage before the vulnerable nodes are consumed. One of the most elegant stochastic models that allows capture this essential behaviors is the birth-and-death model with immigration proposed by Kendall in 1948 [50] which is the object of the next section. Before enter the characteristic of Kendall model, it is worth to remark the three contributions provided in this section. The first contribution concerns the adoption of the Kendall's birth-and-death process with immigration as an analytical model for threat propagation with and without countermeasures. In particular, the appropriate model is a *pure-birth* model with immigration, characterized by two parameters: the "immigration" rate ν, representing the number of hosts per unit time, infected by the original source of attack (here the meaning of the term "immigration" resides in the fact that the attacker is seen as an external entity that injects the threat into the system); and the "birth" rate λ, representing the rate of hosts infected by another infected host. Accordingly, ν is an *external* infection rate, whereas λ is an *internal* infection rate. Besides, the model becomes the birth-*and-death* model with immigration, where a third parameter comes up, namely, the "death" rate μ, which takes on the meaning of the number of cured hosts per unit time. Letting $(\lambda_\ell, \mu_\ell, \nu_\ell)$ be the parameters referring to the ℓ-th subnet, the adopted countermeasures correspond to choose the curing rate

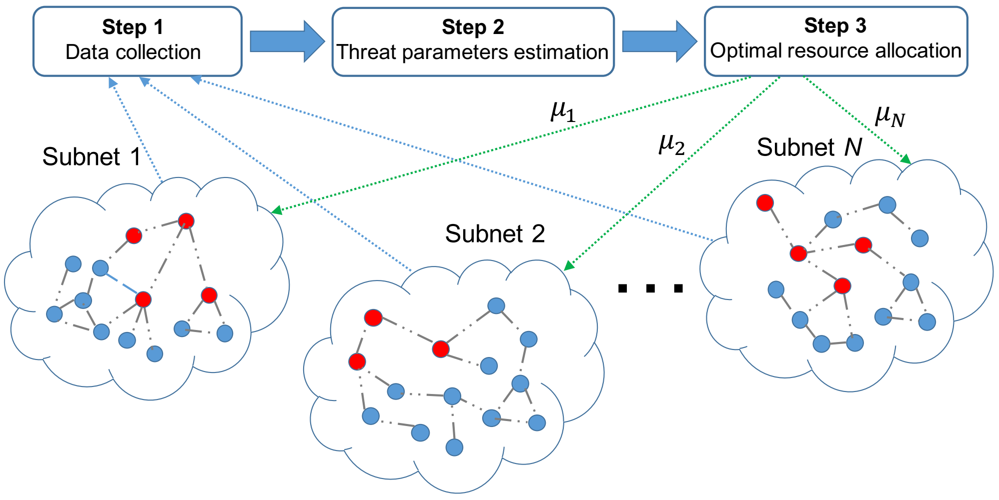


Figure 3.2: Representation of the curing process divided in three phases: collection of parameters (*phase 1*), estimation of attacks parameters (*phase 2*), optimal resource allocation (*phase 3*).

vector, $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)$, in order to minimize the threat diffusion. Due to inevitable resource limitations, a total-rate constraint must be enforced, such that the overall curing capacity must fulfill: $\sum_{\ell=1}^{N} \mu_\ell \leq C$. Such model exhibits two interesting properties. It is a *stochastic* model, an advisable requirement for complex data networks, whereas deterministic models are expected to be oversimplified. Besides, the proposed model is able to catch the main characteristics of threat propagation that appeared in the topical literature, which are conveniently summarized in a few parameters with a clear physical meaning. The second contribution consists in exploiting some useful properties of Kendall's model to provide the exact solution for the optimal resource allocation problem. Such exact solution is derived for the case where the infection parameter vectors, $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$, are perfectly known. The third contribution concerns in removing the assumption of perfect knowledge, and in providing Maximum Likelihood (ML) estimators of the unknown parameters. In summary, it is possible to sketch a pipeline for threat mitigation that works as follows: i) observe the N subnets under attack for a certain time; *ii*) based on this observation, estimate the infection parameter vectors, $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$; *iii*) implement the countermeasures allocating the curing capacity according to the optimal vector $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_N)$. This pipeline has been represented in Fig. 3.2.

3.2 Birth and Death models with Immigration (B-D-I)

Let's start to describe the birth-and-death model in connection to the specific network application considered in this research track. A group of hackers disseminates a certain threat (e.g., a malware) across a network, acting as *primary* source of infection. In order to achieve the goal, the hackers seek for vulnerable nodes by probing, progressively as time elapses, new zones of the network. When a *vulnerable* node is found, it is attacked and gets "sick" after a certain amount of time. It is important to highlight that the primary source disseminates the infection across the network *intentionally and continually.* As a result, the primary source cannot be cured, and runs forever. The global rate of infection related to the primary source(s) is denoted by ν. The times of infection will be conveniently considered as *random* variables. Accordingly, the counting process associated to the number of sick nodes will be modeled by a Poisson counting process with rate ν. Thus, the arrival times of the Poisson process coincide to the times at which the nodes are effectively infected. Once a node gets sick, it becomes a *secondary* source of infection. As soon as a new node is infected, it starts probing vulnerable nodes all across the network, giving rise to a new Poisson process assumed, for simplicity, identical across nodes, and equal to λ. Independence across nodes is assumed, implying the mutual independence of Poisson processes corresponding to distinct nodes. When a node gets sick, it is assumed to be cured after a certain time, which will be assumed as *random* time. More specifically, the curing time will be modeled as an exponential random variable of rate μ, and curing times of distinct nodes will be assumed as statistically independent. It is useful to remark that the Poisson infection process that starts from a certain sick node ends as soon as such node is cured. An important assumption is that the primary source of infection cannot be cured, namely, the primary infection process is a *never-ending* process. From the aforementioned description, it is possible to note that the propagation model exhibits a tree-type complexity, since each infected node acts as a new source of infection. Thus, the model is open to an exponential growth, which could give rise to an *unstable regime* of operation as time elapses. On the other hand, it is expected that the infection growth is related to the balance and interplay between the infection and curing processes. Accordingly, it is legitimate to ask whether some *stable* regime exists, where such (undesired) growth can be avoided. I shall comment exhaustively on these aspects in the forthcoming sections. As we shall see, the analytical characterization that will be given later on, sheds some light on these important questions, and identifies precisely three distinct regimes of operation. Before proceeding, it is useful to introduce some notation. Capital letters will denote random variable and boldface letters will refer to vectors. The symbols $\mathbb{P}$ and $\mathbb{E}$ will denote the probability and expectation operators respectively. The symbols

$$\xrightarrow[t\to\infty]{\mathrm{d}}, \qquad \xrightarrow[t\to\infty]{\mathrm{p}}, \qquad \xrightarrow[t\to\infty]{\mathrm{a.s.}}, \tag{3.1}$$

denote respectively convergence in distribution, convergence in probability, and almost sure convergence as $t \to \infty$. Let $I(t)$ be the number of infected nodes at time t, and let the probability of having n infected nodes at time t be:

$$p(n;t) \triangleq \mathbb{P}[I(t) = n]. \tag{3.2}$$

I introduce also the Moment Generating Function (MGF) of the number of infected nodes at time t:

$$\Psi(x;t) \triangleq \mathbb{E}[e^{xI(t)}], \tag{3.3}$$

and, to avoid confusion, I stress that x is the argument of the MGF, while t is the time variable.

For the birth-and-death model with immigration, it is possible to find a closed-form solution for the MGF (and, then, for the corresponding probability distribution). More specifically, it is rather straightforward to describe the time evolution of the MGF through a first-order linear partial differential equation, namely, through Eq. (3.13) further ahead. I now outline briefly how such time evolution can be obtained. Let us consider a *vanishing* time interval of size ϵ. Given that there are $n-1$ nodes in the system, the infection (i.e., arrival) process is a Poisson process of global rate

$$\lambda(n) \triangleq (n-1)\lambda + \nu, \tag{3.4}$$

which aggregates the internal (i.e., $(n-1)\lambda$) as well as the external (i.e., ν) infection rate components. Using the known properties of Poisson processes, the probability of reaching the state n is approximately given by $\lambda(n)\epsilon$. Likewise, given that there are $n+1$ nodes in the system, and using the known properties of the exponential distribution, the probability of reaching the state n is approximately given by $\mu(n)\epsilon$, with $\mu(n) \triangleq (n+1)\mu$. For the same reasons, reaching the state n from state $n \pm k$, with $k > 1$, is infinitesimal of higher order, and is accordingly neglected. We can accordingly write, by the law of total probability:

$$\begin{aligned} p(n; t+\epsilon) &= \lambda(n)\epsilon\, p(n-1; t) + \mu(n)\epsilon\, p(n+1; t) \\ &+ [1 - \lambda(n) - \mu(n)]\epsilon\, p(n; t). \end{aligned} \tag{3.5}$$

Dividing by ϵ, and taking the limit as $\epsilon \to 0$, we get an infinite system of differential-difference equations, namely, the classic birth-and-death *master equations* [50]:

$$\begin{aligned} \frac{dp(n;t)}{dt} &= \lambda(n)\, p(n-1;t) - [\lambda(n) + \mu(n)]\, p(n;t) \\ &+ \mu(n)\, p(n+1;t). \end{aligned} \tag{3.6}$$

If we now multiply both sides of the above equation by e^{nx}, and sum over n, after simple algebraic manipulations we obtain:

$$\begin{aligned}
&\sum_{n=0}^{\infty} \frac{dp(n;t)}{dt} e^{nx} \\
&= \lambda e^{x} \sum_{n=0}^{\infty} np(n;t)e^{nx} + \nu e^{x} \sum_{n=0}^{\infty} p(n;t)e^{nx} \\
&- (\lambda+\mu) \sum_{n=0}^{\infty} np(n;t)e^{nx} - \nu \sum_{n=0}^{\infty} p(n;t)e^{nx} \\
&+ \mu e^{-x} \sum_{n=0}^{\infty} np(n;t)e^{nx}.
\end{aligned} \tag{3.7}$$

Exchanging differentiation and infinite summation yields:

$$\begin{aligned}
\sum_{n=0}^{\infty} \frac{dp(n;t)}{dt} e^{nx} &= \frac{d}{dt} \sum_{n=0}^{\infty} p(n;t)e^{nx} = \frac{\partial \Psi(x;t)}{\partial t} \\
&= \lambda e^{x} \frac{\partial \Psi(x;t)}{\partial x} + \nu e^{x} \Psi(x;t) \\
&- (\lambda+\mu) \frac{\partial \Psi(x;t)}{\partial x} - \nu \Psi(x;t) \\
&+ \mu e^{-x} \frac{\partial \Psi(x;t)}{\partial x},
\end{aligned} \tag{3.8}$$

which, grouping suitably the terms, corresponds to (3.11).

We are now ready to state some known results regarding the statistical characterization of the BDI process. At certain points, it will be convenient to work in terms of the following normalized quantities:

$$\Delta \triangleq \lambda - \mu, \qquad \rho \triangleq \lambda/\mu, \qquad \eta \triangleq \frac{\nu}{\lambda} \tag{3.9}$$

Property 1. *(Statistical characterization of I(t))*

1. *Let:*

$$a(x) \triangleq [\lambda(1-e^{x}) + \mu(1-e^{-x})], \quad b(x) \triangleq \nu(e^{x}-1). \tag{3.10}$$

Then, the moment generating function of $I(t)$ obeys the following first-order linear partial differential equation.

$$\frac{\partial \Psi}{\partial t} + a(x)\,\frac{\partial \Psi}{\partial x} = b(x)\,\Psi \tag{3.11}$$

2. *Let also:*

$$\pi_t \triangleq \frac{e^{\Delta t} - 1}{e^{\Delta t} - 1/\rho}, \qquad q_t \triangleq \frac{e^{\Delta t} - \rho}{e^{\Delta t} - 1}. \tag{3.12}$$

Then, the moment generating function that solves (3.11) is defined in the range $x < \ln(1/\pi_t)$, and is equal to:

$$\Psi(x;t) = \left(\frac{1 - \pi_t}{1 - \pi_t\, e^x}\right)^{\eta + n_0} \left(\frac{1 - q_t\, e^x}{1 - q_t}\right)^{n_0} \tag{3.13}$$

where n_0 is the initial number of infected nodes.

The probability distribution of the number of infected nodes can be expressed in the following closed form, for $n = 0, 1, \ldots$:

$$\begin{aligned} p(n;t) &= (1/\rho)^{n_0} (1 - \pi_t)^{\eta} \pi_t^{n + n_0} \sum_{i=0}^{\min(n_0, n)} (\alpha_t - 1)^i \\ &\times \binom{n_0}{i} \binom{n_0 + n + \eta - i - 1}{n - i}. \end{aligned} \tag{3.14}$$

where

$$\alpha_t = \frac{(1 - q_t)(1 - \pi_t)}{\pi_t} \Delta e^{\Delta t}. \tag{3.15}$$

I remark that, in the special case $\rho = 1$, Eqs. (3.13), (3.14) and (3.15) hold true, provided that one sets:

$$\pi_t = \frac{\lambda t}{\lambda t + 1}, \qquad q_t = \frac{\lambda t - 1}{\lambda t}, \tag{3.16}$$

which can be obtained from (3.12) by setting (see 3.9) $\rho = \lambda/(\lambda -$

Δ), and then taking the limit as $\Delta \to 0$.

Proof. See [14, 60]. $\square$

The statistical characterization presented in Property 1 enables a detailed asymptotic study of the system. In the two following propositions, some known results about the asymptotic characterization of the BDI process are presented, which will be useful to face the proposed setting.

3.2.1 Regimes of operation

It will possible to see that the system exhibits three distinct regimes of operations. It is useful to preliminarily introduce some notations. A negative binomial random variable of parameters $r > 0$ and $0 < p < 1$ will be denoted by $\mathscr{N}_b(r, p)$, featuring a probability mass function equal to [61]:

$$p_{\rm nb}(n) = \binom{n+r-1}{n}(1-p)^r p^n, \qquad (n = 0, 1, \dots). \tag{3.17}$$

A unit-scale gamma random variable with scale parameter $r > 0$ will be denoted by $\mathscr{G}(r)$, featuring a probability density function equal to:

$$f_{\rm G}(z) = \frac{1}{\Gamma(r)} z^{r-1}\, e^{-z}, \qquad z > 0 \tag{3.18}$$

where $\Gamma(\cdot)$ is the (complete) gamma function [62]. Finally, let $\mathscr{Y}(r, s, m)$ with $r > 0$, $s > 1$, and $m \in \{0, 1, \dots\}$ be a random variable with moment generating function equal to:

$$\Psi_{\rm Y}(x) = \left(\frac{1}{1 - \frac{xs}{s-1}}\right)^{r+m} \left(1 - \frac{x}{s-1}\right)^{m}, \tag{3.19}$$

defined for $x < 1 - 1/s$ We have the following result.

Property 2. *(Asymptotic regimes of operations)*

$$\boxed{\begin{array}{ll} I(t) \xrightarrow[t\to\infty]{d} \mathscr{N}_b(\eta,\rho) & \text{if } \rho < 1, \\ \\ \dfrac{I(t)}{\lambda t} \xrightarrow[t\to\infty]{d} \mathscr{G}(\eta) & \text{if } \rho = 1\ , \\ \\ I(t)\, e^{-\Delta t} \xrightarrow[t\to\infty]{d} \mathscr{Y}(\eta,\rho,n_0) & \text{if } \rho > 1 \end{array}} \tag{3.20}$$

Proof. See Appendix A.1. □

Property 2 provides a helpful description of the possible modes of operation in the threat propagation model. There are three fundamental regimes of operations. The first regime, corresponding to the case $\rho < 1$, is a *stable* regime of operation, where the probability distribution of the number of infected nodes approaches, as time elapses, a negative binomial distribution of parameters ρ and η. The second regime, corresponding to the case $\rho = 1$, is an *unstable* regime of operation, where the probability distribution of the number of infected nodes, *scaled by* λt, converges to an exponential distribution. We see that, on average, the number of infected nodes increases *linearly* with time. The third regime, corresponding to the case $\rho > 1$, is a *strongly unstable* regime of operation, where the probability distribution of the number of infected nodes, *scaled by* $e^{\Delta t}$, converges to an exponential distribution. We see that, on average, the number of infected nodes increases *exponentially* with time. Remarkably, the transition between the three regimes of operation occurs in a sharp, non-smooth way with respect to variations of the system parameter ρ. The strongly unstable regime (for $\rho > 1$) is of interest whenever the curing rate is smaller than the infection rate, a situation that is frequently encountered when facing threat propagation. This observation applies especially in the context of large networks and at the early stages of the infection spreading, which are the focus of the present work. Moreover, we observe that the regime of operation of the system is necessarily unstable in the initial phase of threat propagation, namely, before that any countermeasure has been implemented. The exponential divergence of birth-and-death models for $\rho > 1$ and, more in general, of branching processes operating in such a regime, has been studied extensively in the literature [23, 63, 64]. In fact, it is possible to sharpen the weak convergence result presented in Property 2 by establishing that exponential divergence holds over the sample paths (i.e., almost surely).

Property 3. *(Almost sure exponential divergence of $I(t)$ when $\rho > 1$) Under the unstable regime with $\rho > 1$, the number of infected nodes, $I(t)$, diverges exponentially with probability one. More specifically, the scaled process, $I(t)e^{-\Delta t}$, converges almost surely to a limiting random variable $I^\star$, namely,*

$$I(t)e^{-\Delta t} \xrightarrow[t\to\infty]{a.s.} I^\star \sim \mathscr{Y}(\eta, \rho, n_0), \quad (\rho > 1) \tag{3.21}$$

Proof. From Lemma 1 in the Appendix A.1, we know that the random process $I(t)e^{-\Delta t}$ admits almost surely a (finite) limit, which is denoted by $I^\star$. On the other hand, from Proposition 1 we know that $I(t)e^{-\Delta t}$ converges weakly to $\mathscr{Y}(\eta, \rho, n_0)$. Since almost sure convergence implies weak convergence, we conclude that $I^\star$ is distributed as $\mathscr{Y}(\eta, \rho, n_0)$. ☐

It is interesting to note that, in the context of branching processes, the exponent Δ is usually referred to as *Malthusian* parameter, stemming from the fact that T.R. Malthus is credited to have conjectured the exponential growth of populations for the first time [65]. For later use, it is important to specialize (3.21) to the case $\mu = 0$, which yields:

$$I(t)e^{-\lambda t} \xrightarrow[t\to\infty]{\text{a.s.}} I^\star \sim \mathscr{G}(\eta + n_0), \quad (\mu = 0) \tag{3.22}$$

The result in Property 3 is a strong result that deserves special attention in our setting, for the following reasons. Since almost sure convergence is a convergence over the sample paths, Property 3 reveals that, even if the process $I(t)$ is random, (almost) all realizations of $I(t)$ will share the same behavior:

$$I(t) \approx I^\star e^{\Delta t} \qquad \text{for large } t, \tag{3.23}$$

i.e., they eventually increase exponentially fast with rate Δ. On the other hand, the randomness of the limiting variable $I^\star$ gives rise to a *non-ergodic* behavior of the process, because different realizations will feature a different constant multiplying the exponential factor $e^{\Delta t}$. Such non-ergodic behavior will play an important role in the estimation problem addressed in the next section. The above remarks are more clearly visible in Fig. 3.3, where are depicted (log-scale on the vertical axis) three realizations of the process, under the unstable regime, along with the theoretical exponential curve $e^{\Delta t}$ (dashed line). It is possible to see that all the realizations stay nearly parallel to the theoretical exponential curve, which matches perfectly (3.21) and (3.23). The different heights of the curves correspond to the different realization of the *random* multiplying constant, $I^\star$. The latter behavior is magnified in the inset of Fig. 3.3, where the normalized process, $I(t)e^{-\Delta t}$, is depicted. As predicted by (3.21), the normalized curves converge, over the sample paths, to some limit points. The different limit points correspond to the different realizations of the limiting random variable, $I^\star$.

3.3 Optimal resource allocation for threat mitigation

In this section the optimization of resource allocation with perfect knowledge of the infection parameter vectors, $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ is considered. Later on, a methodology to estimate such parameters when they are unknown will be presented. Ideally, one would like to solve the following optimization problem:

$$\min_{\boldsymbol{\mu}} \sum_{\ell=1}^{N} I_\ell(t) \quad \text{s.t.} \quad \sum_{\ell=1}^{N} \mu_\ell \leq C \tag{3.24}$$

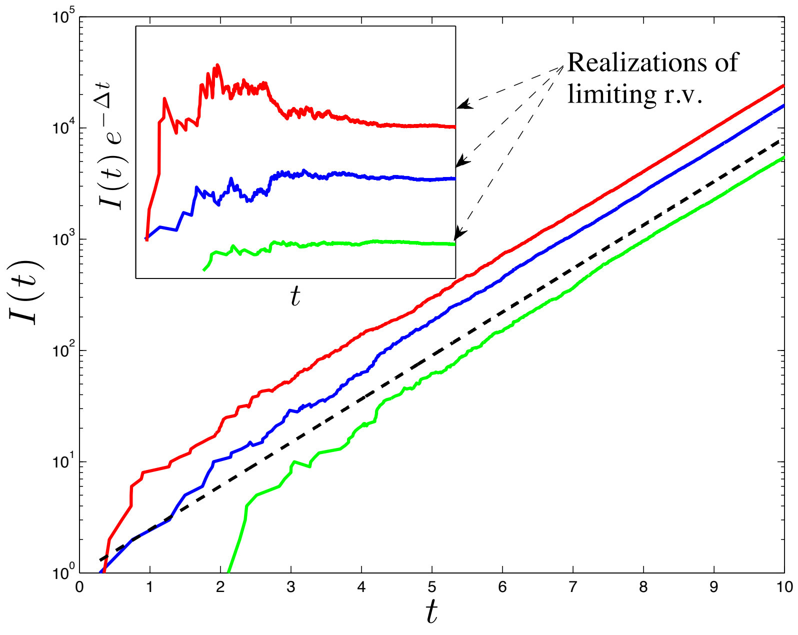


Figure 3.3: Semi-log representation of three realizations (red, blue, and green lines) of random process I(t), and the theoretical exponential curve $e^{\Delta t}$ (black dashed line) under the unstable regime. In the inset, the normalized process $I(t)e^{-\Delta t}$ is depicted where the three distinct limit points correspond to the distinct realizations of the limiting random variable $I^{\star}$.

However, the functions $I_\ell(t)$ are *random* processes, and, hence, we must choose a proper cost function that is amenable to optimization. We must distinguish two regimes for the optimization, which are determined by the available system capacity, C.

The first regime corresponds to the case that the global infection rate is greater than the available capacity, namely, $\sum_{\ell=1}^{N} \lambda_\ell > C$. It is possible to see that, no matter what allocation is employed, at least for one ℓ one should have $\lambda_\ell > \mu_\ell$. Let us accordingly define:

$$\Delta_{\max} \triangleq \max_{\ell=1,2,\ldots,N} (\lambda_\ell - \mu_\ell) > 0. \tag{3.25}$$

The following proposition ascertains the asymptotic behavior of the overall number of infected nodes under the regime $\sum_{\ell=1}^{N} \lambda_\ell > \sum_{\ell=1}^{N} \mu_\ell$.

Proposition 1 (Almost sure exponential divergence of the overall number of infected nodes). *If $\sum_{\ell=1}^{N} \lambda_\ell > \sum_{\ell=1}^{N} \mu_\ell$, then, almost surely, the overall number of infected nodes diverges exponentially with exponent equal to Δ_{max}, namely, we have that:*

$$\sum_{\ell=1}^{N} I_\ell(t) e^{-\Delta_{\max} t} \xrightarrow[t\to\infty]{a.s.} I^\star_{sum}, \tag{3.26}$$

where random variable $I^\star_{sum}$ has MGF equal to:

$$\prod_{\ell\in\mathcal{S}} \left(\frac{1}{1-\frac{x\rho_\ell}{\rho_\ell-1}} \right)^{\eta_\ell+n_{0,\ell}} \left(1-\frac{x}{\rho_\ell-1}\right)^{n_{0,\ell}}, \; x < 1-\frac{1}{\rho_{\min}}, \tag{3.27}$$

with x being the argument of MGF, $\rho_{\min} \triangleq \min_{\ell\in\mathcal{S}} \rho_\ell$, where the subscript ℓ has been appended to all system parameters to denote dependence on the particular subnet, and where:

$$\mathcal{S} \triangleq \{\ell \in [1,N] : \lambda_\ell - \mu_\ell = \Delta_{max}\}, \tag{3.28}$$

Proof. In view of (3.28), $\lambda_\ell - \mu_\ell = \Delta_{max}$ for all $\ell \in \mathcal{S}$. Thus, straight application of Property 3 yields:

$$I_\ell(t) e^{-\Delta_{\max} t} \xrightarrow[t\to\infty]{\text{a.s.}} I^\star \sim \mathscr{Y}(\eta_\ell, \rho_\ell, n_{0,l}), \quad [\ell \in \mathcal{S}]. \tag{3.29}$$

It is now possible to show that $I_\ell(t)e^{-\Delta_{max}(t)}$ converges to zero almost surely when $\ell \notin \mathcal{S}$. As first, it is possible to show that the results holds distribution, namely, that:

$$I_\ell(t) e^{-\Delta_{\max} t} \xrightarrow[t\to\infty]{\text{p}} 0 \quad [\ell \notin \mathcal{S}]. \tag{3.30}$$

Now, it is possible to note that [66]: if $Y(t)$ converges to Y in distribution, and the (deterministic) function $f(t)$ converges to 0, then the product $f(t)Y(t)$ converges to 0 in probability. Let's apply this result to the three possible regimes characterizing the considered system. For the case $\rho_\ell < 1$, $I_\ell(t)$ converges in distribution for the Property 2. Setting $Y(t) = I_\ell(t)$ and $f(t) = e^{-\Delta_{max} t}$, (3.30) holds true because $e^{-\Delta_{max} t}$ vanishes as t goes to infinity. In the case that $\rho_\ell = 1$, $(\lambda_\ell t)^{-1} I_\ell(t)$ converges in distribution in view of Property 2. Setting $Y(t) = (\lambda_\ell t)^{-1} I_\ell(t)$ and $f(t) = (\lambda_\ell t) e^{-\Delta_{max} t}$, (3.30) holds true because $(\lambda_\ell t) e^{-\Delta_{max} t}$ vanishes. Finally, in the case that $\rho_\ell > 1$, $I_\ell(t) e^{-\Delta_\ell t}$ converges in distribution for the Property 2. Setting $Y(t) = I_\ell(t) e^{-\Delta_\ell t}$ and $f(t) = e^{(\Delta_\ell - \Delta_{max})t}$, (3.30) holds true because $e^{(\Delta_\ell - \Delta_{max})t}$ vanishes for all $\ell \notin \mathcal{S}$. Moreover, since for $\ell \notin \mathcal{S}$ we have $\Delta_{\max} > \Delta_\ell$, from Lemma 1 in the Appendix A.1 it is possible to conclude that the convergence in (3.30) actually holds in the almost sure sense. Merging this result with (3.29) we get the claim of the proposition, provided that we set:

$$I^\star_{sum} = \sum_{\ell \in \mathcal{S}} I^\star_\ell. \tag{3.31}$$

Claim 3.27 follows because: the processes corresponding to different subnets are statistically independent; the MGF of the sum of independent variables is given by the product of the MGFs of the variables; in view of (3.29), the MGF of $I^\star_\ell$ is given by (3.19), by choosing: $r = \eta_\ell, s = \rho_\ell$, and $m = n_{0,\ell}$. $\square$

According to Proposition 1, in the case $\sum_{\ell=1}^N \lambda_\ell > C$, we can therefore conclude that, for t large enough, the behavior of the overall number of infected nodes across the N subnets will be determined by an exponential function featuring the largest exponent, namely,

$$\sum_{\ell=1}^{N} I_\ell(t) \approx e^{\Delta_{\max} t} \tag{3.32}$$

where the symbol "$\approx$" here means "scales as". As a consequence, the optimization in (3.24) can be meaningfully reformulated as:

$$\min_{\boldsymbol{\mu}} \max_{\ell \in [1,N]} (\lambda_\ell - \mu_\ell) \quad \text{s.t.} \sum_{\ell=1}^{N} \mu_\ell \leq C, \tag{3.33}$$

namely, we focus on the minimization *at the exponent.* It is easy to show that the minimizer is obtained through the so-called reverse water-filling solution:

$$\mu_\ell^\star = \max(0, \lambda_\ell - \gamma) \qquad (\ell = 1, 2, \dots, N) \tag{3.34}$$

where γ is set to met the constraint $\sum_{\ell=1} \mu_\ell = C$. Let us move on examining the most favorable case where the system capacity is larger than the global infection rate, namely, $\sum_{\ell=1}^N \lambda_\ell < C$. In this case it will be certainly possible to reach the desirable condition $\lambda_\ell < \mu_\ell$ for all $\ell = 1, 2, \dots, N$, which prevents from exponential divergence in all subnets. According to Property 2, in this case we cannot rely on almost sure convergence, and, hence, we cannot consider a cost function directly on the sample paths. A comfortable and meaningful choice is that of minimizing the expected number of infected nodes. To this aim, it is useful to remark that the weak convergence expressed by the first equation in (3.20) can be obtained by proving the stronger result that the MGF of $I(t)$ converges to the MGF of $\mathscr{N}_b(\eta, \rho)$ (see the proof of Property 2 in the Appendix A.1). Since convergence of the MGF implies converges of moments [67], and the expectation of $\mathscr{N}_b(\eta, \rho)$ is equal to $\frac{\eta\rho}{1-\rho} = \frac{\nu}{\mu-\lambda}$ [61] we can write:

$$\mathbb{E}\left[\sum_{\ell=1}^N I_\ell(t)\right] \xrightarrow[t\to\infty]{} \sum_{\ell=1}^N \frac{\nu_\ell}{\mu_\ell - \lambda_\ell}, \tag{3.35}$$

and accordingly focus on the optimization problem:

$$\min_{\boldsymbol{\mu}} \sum_{\ell=1}^N \frac{\nu_\ell}{\mu_\ell - \lambda_\ell} \quad \text{s.t.} \quad \sum_{\ell=1}^N \mu_\ell \le C \tag{3.36}$$

with $\mu_\ell > \lambda_\ell$ for all $\ell = 1, 2, \dots, N$. We will solve this problem by the Lagrange multipliers method. Let us introduce the Lagrangian:

$$J(\boldsymbol{\mu}) = \sum_{\ell=1}^N \frac{\nu_\ell}{\mu_\ell - \lambda_\ell} + \beta \sum_{\ell=1}^N \mu_\ell \tag{3.37}$$

where β is the Lagrange multiplier. Taking the partial derivative with respect to the k-th component, we get:

$$\begin{aligned}\frac{\partial J}{\partial \mu_k} &= -\frac{\nu_k}{(\mu_k-\lambda_k)^2}+\beta=0 \\ &\Rightarrow \quad \mu_k=\lambda_k+\beta\sqrt{\nu_k}. \qquad (3.38)\end{aligned}$$

Imposing the constraint with equality yields:

$$\mu_k^\star=\lambda_k+\delta C\frac{\sqrt{\nu_k}}{\sum_{\ell=1}^{N}\sqrt{\nu_\ell}} \qquad (k=1,2,\ldots,N) \qquad (3.39)$$

where we defined

$$\delta C = C-\sum_{\ell=1}^{N}\lambda_\ell. \qquad (3.40)$$

3.4 Maximum Likelihood estimation of B-D-I parameters

When the infection parameters are unknown, it is necessary to estimate them before choosing the optimal allocation detailed in the previous section. Clearly, such estimation takes place before the implementation of the countermeasures, which, according to our model, corresponds to the case that $\mu=0$. We denote the process observed until time epoch t as:

$$X_t \triangleq \{I(\tau)\}_{\tau\le t}. \qquad (3.41)$$

The pertinent log-likelihood function can be written as:

$$\mathscr{L}(X_t;\lambda,\nu)=\sum_{n=0}^{I(t)-1}\ln(n\lambda+\nu)-\lambda\int_0^t I(\tau)d\tau-\nu t \qquad (3.42)$$

which is obtained by exploiting the properties of exponential inter-arrivals characterizing the system (and corresponding to specialize the log-likelihood function used in [68] to the case $\mu=0$). We shall examine the maximum likelihood solution:

$$(\hat{\lambda}, \hat{\nu}) = \arg\max_{\lambda,\nu} \mathscr{L}(X_t; \lambda, \nu). \tag{3.43}$$

In order to compute the ML estimators, let us evaluate the partial derivatives:

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \lambda} &= \sum_{n=0}^{I(t)-1} \frac{n}{n\lambda + \nu} - \int_0^t I(\tau) d\tau, & (3.44) \\ \frac{\partial \mathscr{L}}{\partial \nu} &= \sum_{n=0}^{I(t)-1} \frac{1}{n\lambda + \nu} - t & (3.45) \end{aligned}$$

We start looking for the stationary point $(\hat{\lambda}, \hat{\nu})$ where the above two derivatives nullify, yielding:

$$\begin{aligned} \sum_{n=0}^{I(t)-1} \frac{n}{n\hat{\lambda} + \hat{\nu}} &= \int_0^t I(\tau) d\tau, & (3.46) \\ \sum_{n=0}^{I(t)-1} \frac{n}{n\hat{\lambda} + \hat{\nu}} &= t. & (3.47) \end{aligned}$$

Since we can write:

$$\begin{aligned} \sum_{n=0}^{I(t)-1} \frac{n}{n\lambda + \nu} &= \sum_{n=0}^{I(t)-1} \frac{n + \nu/\lambda}{n\lambda + \nu} \\ &- \sum_{n=0}^{I(t)-1} \frac{\nu/\lambda}{n\lambda + \nu} \\ &= \frac{I(t)}{\lambda} - \frac{\nu}{\lambda} \sum_{n=0}^{I(t)-1} \frac{1}{n\lambda + \nu} & (3.48) \end{aligned}$$

using (3.47) in (3.48) we get:

$$\sum_{n=0}^{I(t)-1} \frac{n}{n\hat{\lambda} + \hat{\nu}} = \frac{I(t)}{\hat{\lambda}} - \frac{\hat{\nu}}{\hat{\lambda}} t. \tag{3.49}$$

Substituting (3.49) in (3.46), we obtain:

$$\hat{\lambda} = \frac{I(t)}{\int_0^t I(\tau) d\tau}, \tag{3.50}$$

which corresponds exactly to the ML solution for the pure-birth model without immigration [69]. With this approximate choice, one has obtained the decoupled estimation of λ from estimation of ν. Besides, it is possible to observe that the integral $\int_0^t I(\tau)d\tau$ fulfills the following condition:

$$\int_0^t I(\tau) d\tau e^{-\lambda t} \xrightarrow[t\to\infty]{\text{a.s.}} \frac{I^\star}{\lambda} \tag{3.51}$$

Almost sure exponential divergence of integral and other functionals of branching processes have been addressed in the literature see, e.g., [63]. In particular, in our case, the convergence in (3.51) can be obtained in a straightforward manner from the convergence in (3.22), by using the same procedure applied in [[69], Th. 2.2]. Using (3.22) and (3.51), it is possible to conclude that $\hat{\lambda}$ converges almost surely to λ as t goes to infinity, namely, $\hat{\lambda}$ is a.s. consistent [66]. Let's now switch to examining (3.47). The summation appearing in (3.45), can be expressed as follows:

$$\sum_{n=0}^{I(t)-1} \frac{1}{n\lambda + \nu} = \frac{\psi(I(t) + \nu/\lambda) - \psi(\eta)}{\lambda}, \tag{3.52}$$

where $\psi(\cdot)$ is the digamma function [62]. For large values of its argument, the digamma function can be approximated by the function $ln(x-1/2)$ and it can be verified that such approximation is excellent in the range $x \geq 1$, which is important in our case. By examining the (3.52), it is possible to see that the pertinent arguments of the digamma function are typically greater than 1, because: *i)* for meaningful values of t on has that $I(t) \gg 1$; *ii)* typically one has $\nu/\lambda > 1$, since the infection rate sustained by the primary source is expected to be larger than the infection rate sustained by a generic infected host. Accordingly, by employing the approximation $\psi(x) \approx ln(x - 1/2)$ in (3.52) and by using (3.47) it is possible to obtain the following approximate condition:

$$ln\frac{I(t)+\hat{\nu}/\hat{\lambda}-1/2}{\hat{\nu}/\hat{\lambda}-1/2} \approx \hat{\lambda}t \to \hat{\nu} \approx \hat{\lambda}\left(\frac{I(t)}{e^{\hat{\lambda}t}}+\frac{1}{2}\right). \tag{3.53}$$

We know that the estimator $\hat{\lambda}$ converges almost surely to the true value λ. In order to assert the role of the term 1/2, let us roughly replace $\hat{\lambda}$ with λ in 3.53, yielding:

$$\hat{\nu} \approx \lambda\left(I^{\star}+\frac{1}{2}\right) \triangleq \hat{\nu}. \tag{3.54}$$

Recalling from Property 3 that $I^{\star}$ is a unit-scale gamma random variable with shape parameter $\eta = \nu/\lambda$, one has that [61]:

$$\mathbb{E}[\hat{\nu}] = \nu + \frac{\lambda}{2}, \qquad \mathbb{E}[(\hat{\nu}-\nu)^2] = \lambda\nu + \frac{\lambda^2}{4} \tag{3.55}$$

suggesting that the term 1/2 in (3.54), does not bring any benefit to the mean-square estimator's performance. Thus, it is possible to safely remove such constant term, yielding the final pair of estimators:

$$\hat{\lambda} = \frac{I(t)}{\int_0^t I(\tau)d\tau}, \qquad \hat{\nu} = \lambda\frac{I(t)}{e^{\hat{\lambda}t}} \tag{3.56}$$

The consistency holds as regards $\hat{\lambda}$, but the situation for $\hat{\nu}$ is different and a non-vanishing error is expected as t gets large – see (3.55). This behavior is perfectly in accordance with the non-ergodicity remarked in the comment of Fig.3.3. Indeed, such non-ergodicity manifests itself in the fact that $I^{\star}$, which contains information about ν, is still random. This behavior might be ascribed to the fact that the effect of the primary source is somehow obfuscated by the effect of the infected nodes, which produce a global (internal) infection rate, $n\lambda$, scaling exponentially with time (because the number of infected nodes grows exponentially).

3.5 Experimental results

Some experimental results are now discussed concerning the application of the aforementioned optimization procedure for the two possible situations regarding: *i)* the case that the available capacity, C, is smaller than the overall infection rate, $\sum_{\ell=1}^{N} \lambda_\ell$, and *ii)* the opposite case that C is larger. Figure 3.4 refers to the former scenario, depicting the overall number of infected nodes spreading across $N = 3$ subnets, as a function of time. The internal infection rates are $\boldsymbol{\lambda} = [5, 4, 3.9]$, whereas the external (scaled) infection rates are $\boldsymbol{\nu} = [2.5, 4, 7.8]$. The available capacity is set to $C = 0.8 \sum_{\ell=1}^{N} \lambda_\ell$. In the considered example, at one fifth of the observation window the network manager estimates the infection parameters and implements the optimization detailed in (3.34), replacing the actual values of $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ with the estimates provided in (3.56). For comparison purposes, the system that performs the optimization with the actual values of $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ is displayed. It is evident from Fig. 3.4 that, starting from the origin, the number of infected nodes increases exponentially (in the displayed logarithmic scale), and a marked change of slope occurs approximately at one fifth of the time window, namely, after the implementation of the countermeasures following the optimization procedure. The broken lines refer to the slopes (i.e., exponents) predicted by the theoretical analysis. We note that, since the available capacity is smaller than the infection rate, the countermeasures are not sufficient to contain the exponential spread of the threat, while they are effective in mitigating its growth rate. In contrast, in Fig. 3.5 we address the most favorable scenario where $C = 1.1 \sum_{\ell=1}^{N} \lambda_\ell$. In this case, optimization is performed (still after estimating $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$ at one fifth of the observation window) by employing the optimization in (3.39). We see that the optimization is now successful in forcing the stability of the threat growth, which is no longer exponential.

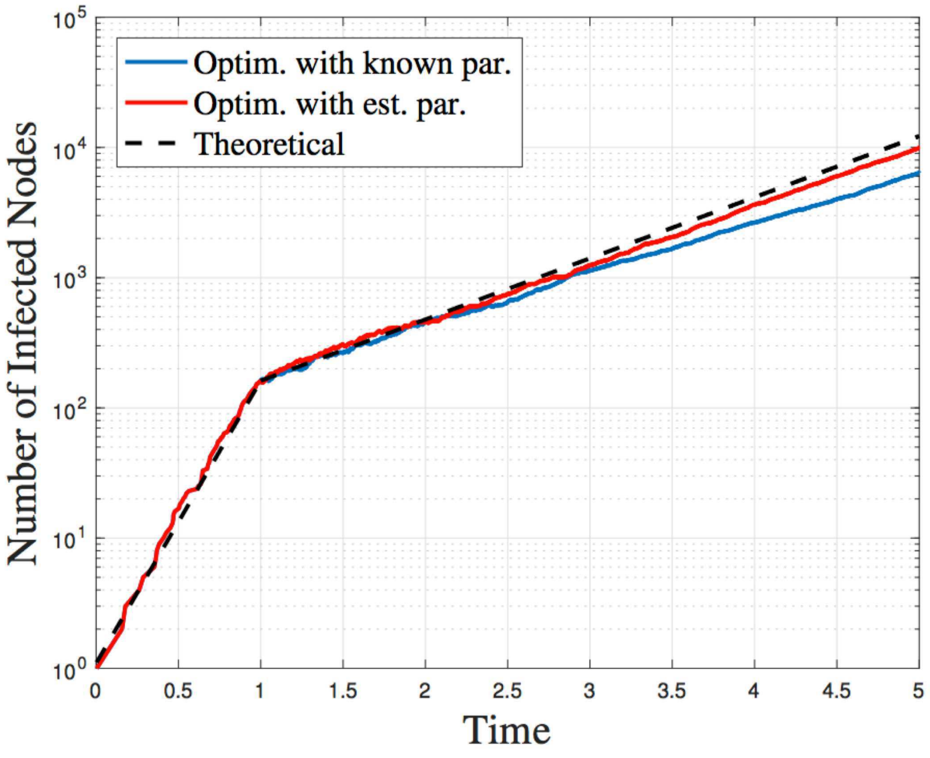


Figure 3.4: Semi-log representation of the number of infected nodes spreading across $N = 3$ subnets. The internal and external infection rates are $\boldsymbol{\lambda} = [5, 4, 3.9]$ and $\boldsymbol{\nu} = [2.5, 4, 7.8]$ respectively, whereas the available capacity is set to $C = 0.8 \sum_{\ell=1}^{N} \lambda_\ell$. The (broken) black dashed lines indicate the slope predicted by theoretical analysis, whereas blue and red curves refer to the optimization procedures in case of known and unknown (estimated via ML criterion) parameters respectively. Being $C > \lambda$, the optimization procedure weakly mitigates the threat growth rate since the number of infected nodes increases exponentially up about one fifth of time window before changing its slope.

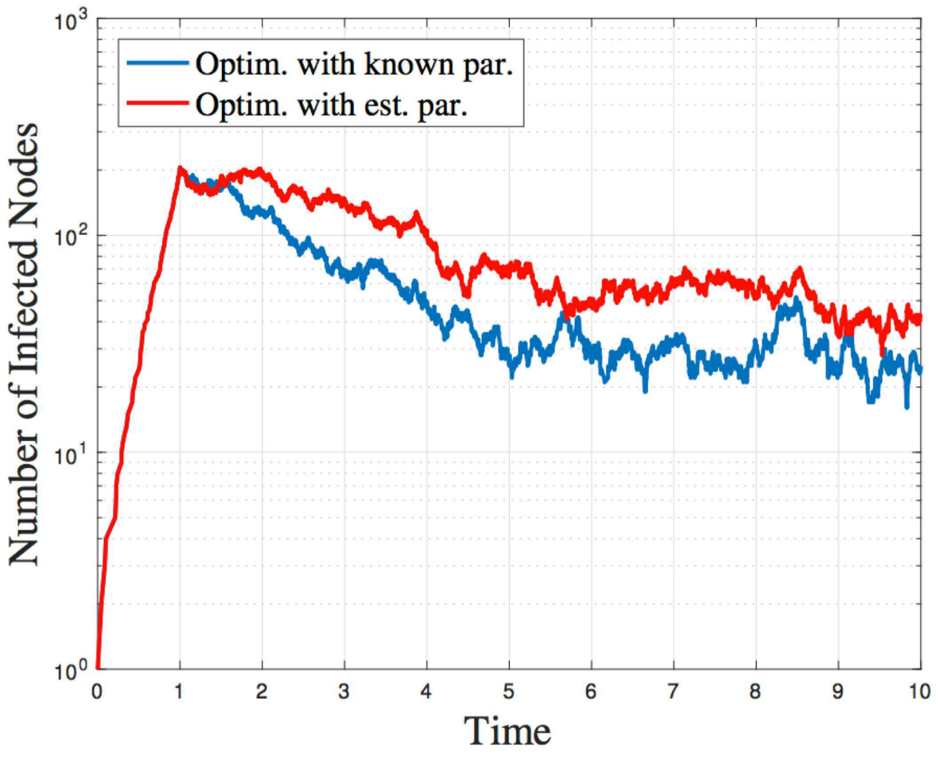


Figure 3.5: Semi-log representation of the number of infected nodes spreading across $N = 3$ subnets. The internal and external infection rates are $\boldsymbol{\lambda} = [5, 4, 3.9]$ and $\boldsymbol{\nu} = [2.5, 4, 7.8]$ respectively, whereas the available capacity is set to $C = 0.8 \sum_{\ell=1}^{N} \lambda_\ell$. Blue and red curves refer to the optimization procedures in case of known and unknown (estimated via ML criterion) parameters respectively. Being $C < \lambda$, the optimization procedure is now able to guarantee the stability of threat growth by avoiding exponential behaviors.

Chapter 4

Randomized DDoS: formal model and performance evaluation

4.1 Main setting and inference strategies

This section focuses on a reinforced DDoS attack that takes inspiration from powerful L7-DDoS. For our purposes, in fact, the botnet is given the strong power of learning an *emulation dictionary* that becomes richer and richer as time elapses. One can think such emulation dictionary as a database built (and kept updated) by a botmaster. The botmaster is able to scan the network (by using network probes or sniffers), and to collect messages originating from legitimate users. Then, such messages are made available to the bots that continually and periodically pick them from the emulation dictionary as pictorially depicted in Fig. 4.1. Given the aforementioned setting, two important questions arise: *i)* is it possible to discriminate (and then to identify) bots from legitimate users? *ii)* is it possible to devise pertinent inferential strategies by guaranteeing a good trade-off between the botnet learning ability and the inference performance? By starting from

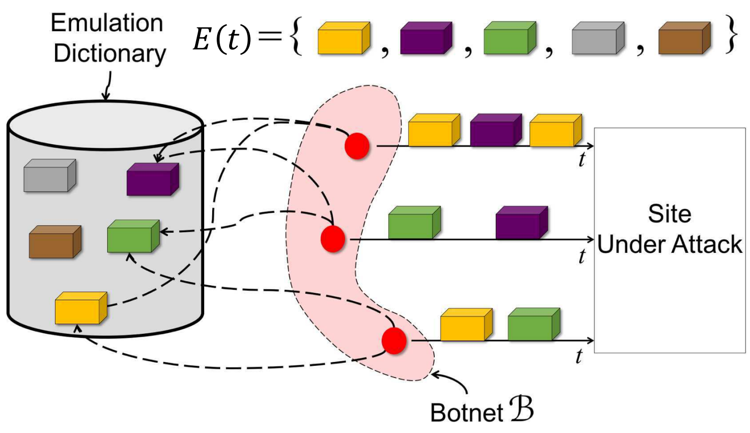


Figure 4.1: The Emulation Dictionary contains a set of legitimate messages. Bots continually pick such messages in order to build apparently legitimate requests exploited to attack a target.

the latter question, the classical inferential strategies derived from technical literature are not properly conceived to manage this class of DDoS attacks with an increasing emulation dictionary. Traditional parametric methods (e.g. Neyman-Pearson tests, maximum likelihood) offer a good degree of tractability in terms of analytical results and performance guarantees, but typically require a detailed knowledge of the underlying model [70, 71]. Such a condition is far from being met in the considered setting. On the contrary, data-driven methods (e.g. machine learning techniques) that operate in a black box environment with a great degree of flexibility, often suffer of lack of analytical results and heavy tuning of algorithms. The approach followed in this research lies in the middle, and takes inspiration from emerging trends of signal processing techniques applied to the cyber-security context [72]. Accordingly, the following guidelines will be kept in mind: *i)* focus on minimal-but-realistic physical assumptions; *ii)* conceive significative indicators arising from the modeling assumptions; *iii)* devise an inference strategy.

4.2 Randomized DDoS model

A formal model for randomized DDoS attacks with growing emulation dictionary, is defined by the following main features: *i*) the botnet emulates the legitimate traffic patterns by gleaning admissible messages from an emulation dictionary; *ii*) the botnet is given the great power of learning an emulation dictionary that becomes richer and richer as time progresses, so as to guarantee a sufficient variability across messages. In order to quantify the botnet *learning ability*, the Emulation Dictionary Rate (EDR) is introduced, accounting for the increase of dictionary cardinality per unit time. Remarkably, the considered class of DDoS attacks is more general and powerful than many attacks documented in the literature. The assumption of such great power in the attacker's hands might perhaps look overly pessimistic. At the same time, a worst-case analysis is perfectly suited to security applications, and allows getting important insights as regards the botnet identifiability under challenging operational conditions.

The fundamental descriptive indicator employed in this work to ascertain the nature of network users is the Message Innovation Rate (MIR), namely, the number of distinct messages per unit time, transmitted by a given group of users. The relevance of the MIR for botnet identification purposes arises since, in view of the coordination in the DDoS attack, the users belonging to a botnet are expected to exhibit a smaller degree of innovation than normal users, which act by their own nature independently one each other.

The first contribution determines the MIR for a botnet $\mathcal{B}$, with either deterministic or Poisson transmission scheduling. Denoting by $\lambda_{\mathcal{B}}$ the transmission rate corresponding to the overall transmission activity in $\mathcal{B}$, and by α the EDR, we show that the MIR converges in probability to the following innovation rate (Theorem 1):

$$\mathscr{R}(\alpha, \lambda_{\mathcal{B}}) = \frac{\alpha\, \lambda_{\mathcal{B}}}{\alpha + \lambda_{\mathcal{B}}} \tag{4.1}$$

The second contribution consists in devising an algorithm that, under a suitable Botnet Identification Condition (BIC), guarantees that the botnet hidden in the network is correctly identified as time elapses.

Finally, as a third contribution, all of the aforementioned theoretical results are tested and validated on a testbed environment; the experimental outcomes are definitely encouraging.

Notation. $\mathbb{P}[\cdot]$ and $\mathbb{E}[\cdot]$ denote the probability and the expectation operators, respectively. Given an ensemble of random variables X_t (with either continuous or discrete index t), the notation $X_t \xrightarrow{\text{p}} X$ means that X_t converges in probability to X as $t \to \infty$.

4.3 Main network indicators

It is useful to introduce the basic quantities that will be used to describe the network activity. The first quantity relates to the *transmission* activity of the network users. Each user employs a certain scheduling, which is identified by the transmission epochs of its own messages. More in general, for any given subnet $\mathcal{S}$ of the network, it is possible to define the *aggregate* pattern that comprises all (ordered) transmission epochs of the users belonging to $\mathcal{S}$, formally: $T_{\mathcal{S}}(1), T_{\mathcal{S}}(2), \dots,$ where $T_{\mathcal{S}}(i)$ is the i-th (random) transmission epoch of users belonging to $\mathcal{S}$. Likewise, the pattern of an individual user u becomes: $T_u(1), T_u(2), \dots,$ where, with a slight abuse of notation (which will be used throughout the work), we have written u in lieu of $\{u\}$. The total number of transmissions occurred in $\mathcal{S}$, up to a given (deterministic) time t is denoted by $N_{\mathcal{S}}(t) \triangleq |\{i : T_{\mathcal{S}}(i) \leq t\}|$.

As an indicator of the *transmission* activity, it is possible to introduce the *empirical* transmission rate at time t, namely,

$$\hat{\lambda}_{\mathcal{S}}(t) \triangleq \frac{N_{\mathcal{S}}(t)}{t} \tag{4.2}$$

Whenever a limiting rate (as t goes to infinity) is meaningfully defined, it will be denoted by $\lambda_{\mathcal{S}}$, which will be simply referred to as the transmission rate of subnet $\mathcal{S}$.

Two examples of transmission schedulings which are relevant for our DDoS application, and which admit a limiting rate, are the synchronous, constant-rate transmission scheduling, and the independent Poisson scheduling. In the former case, all users transmit synchronously, and the (constant) interval between two transmissions has duration $1/\lambda$. The empirical transmission rate clearly obeys: $\hat{\lambda}_{\mathcal{S}}(t) \to \lambda\,|\mathcal{S}|$ as $t \to \infty$. In the latter case, the transmission pattern of user u is a Poisson process with rate λ_u, and the processes are mutually independent. Since the aggregate of independent Poisson processes is still a Poisson process, as a straightforward application of the (weak) law of large numbers, it is possible to write [73]: $\hat{\lambda}_{\mathcal{S}}(t) \xrightarrow{\mathrm{p}} \sum_{u\in\mathcal{S}} \lambda_u$. As a second indicator of the network activity, it is possible to define a quantity that relates to the *content* of the messages sent by network users. We are interested in the *new* messages that are incrementally produced by the users during their activities, namely, in a Message Innovation Rate (MIR). In order to obtain a formal definition of the MIR, let $\mathscr{D}_{\mathcal{S}}(t)$ denote the empirical dictionary composed by the *distinct* messages sent, up to time t, by users within $\mathcal{S}$. For the sake of clarity, I remark that, if the same message is sent, e.g., twice, from users belonging to $\mathcal{S}$, it appears only once in the dictionary $\mathscr{D}_{\mathcal{S}}(t)$. The *empirical* Message Innovation Rate (MIR) is:

$$\hat{\rho}_{\mathcal{S}}(t) \triangleq \frac{|\mathscr{D}_{\mathcal{S}}(t)|}{t} \tag{4.3}$$

In particular, if $\hat{\rho}_{\mathcal{S}}(t) \xrightarrow{\mathrm{p}} \rho_{\mathcal{S}}$, the limiting value $\rho_{\mathcal{S}}$ will be simply referred to as the MIR of subnet $\mathcal{S}$. The transmission activity and MIR have been summarized in Figg. 4.2 and 4.3 respectively.

4.4 Randomized DDoS with Emulation Dictionary

A botnet $\mathcal{B}_{\mathrm{tot}}$, composed by B_{tot} malicious nodes, sends messages to the destination under attack in order to saturate its resources. The botnet mimics normal patterns by picking messages from an

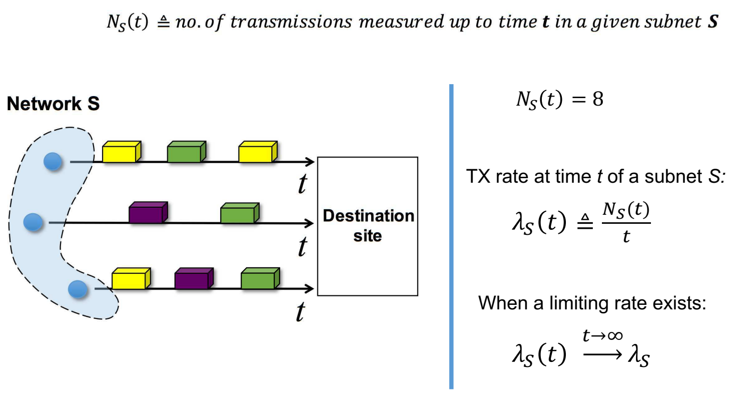

$N_S(t) \triangleq$ no. of transmissions measured up to time t in a given subnet S
Network S
t
Destination site
$N_S(t) = 8$
TX rate at time t of a subnet S:
$\lambda_S(t) \triangleq \frac{N_S(t)}{t}$
When a limiting rate exists:
$\lambda_S(t) \xrightarrow{t\to\infty} \lambda_S$

Figure 4.2: Transmission Rate as an indicator of users transmission activity.

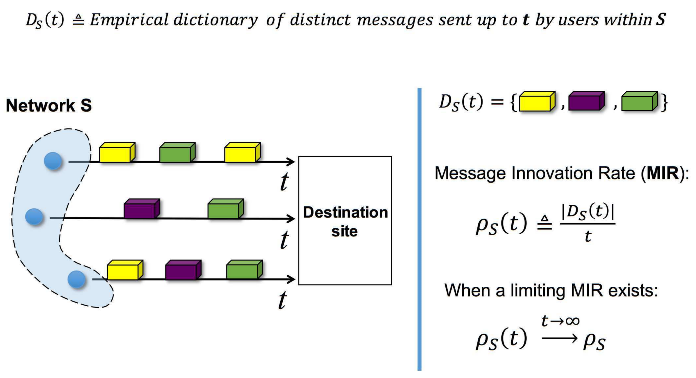

$D_S(t) \triangleq$ Empirical dictionary of distinct messages sent up to t by users within S
Network S
t
Destination site
$D_S(t) = \{$ 🟨, 🟪, 🟩 $\}$
Message Innovation Rate (MIR):
$\rho_S(t) \triangleq \frac{|D_S(t)|}{t}$
When a limiting MIR exists:
$\rho_S(t) \xrightarrow{t\to\infty} \rho_S$

Figure 4.3: Message Innovation Rate as an indicator of capacity of users to produce incrementally new messages.

emulation dictionary, which is learned *continually* (i.e., its cardinality increases with time), in order to ensure that a reasonable innovation rate can be sustained. Such a dictionary construction can occur in many different ways. For instance, by means of one or more powerful botmasters, the botnet might be able to perform an on-line monitoring of normal activities from across the network. From such a monitoring, sequences of messages corresponding to normal patterns of activity are collected, allowing the construction of a dictionary of admissible messages.

Let $\mathscr{E}(t)$ be the (common) dictionary available at time t to all botnet members. It is assumed that the number of messages available for emulation grows, asymptotically, in a linear fashion. Therefore, it makes sense to introduce the Emulation Dictionary Rate (EDR) as:

$$\alpha \triangleq \lim_{t \to \infty} \frac{|\mathscr{E}(t)|}{t} \tag{4.4}$$

Given the emulation dictionary, the botnet has clearly many ways to build the traffic patterns. At one extreme, the botmaster disseminates B_{tot} *disjoint* (say, equal-size) portions of $\mathscr{E}(t)$ through the botnet. Then, each bot builds its traffic pattern by scanning, in a sequential fashion, its portion of the emulation dictionary. Such a scheme would clearly maximize the independence among the bots. With this policy, the problem would become equivalent to the case that each bot owns a distinct emulation dictionary with EDR equal to α/B_{tot}. However, since B_{tot} must be large, it is unrealistic to assume that a botmaster can learn so many patterns to build B_{tot} distinct dictionaries that are in turn so rich to guarantee a credible emulation. Therefore, in the case of disjoint dictionaries, the number of distinct messages available to a single bot would be typically small, implying a suspiciously high degree of replication, which would make the bots easily identifiable by single-user inspection.

At the other extreme, each bot might simply use *all* messages contained in $\mathscr{E}(t)$. Clearly, such scheme maximizes the innovation of each individual bot, but also maximizes the dependence inside the botnet. By inspection of the messages sequentially sent by two or more bots, a traffic analyst would recognize an anomalous behavior.

Hence, it is assumed that the attacker has devised some intermediate strategy to circumvent the aforementioned issues. I introduce a class of *randomized* DDoS attacks, where a bot that intends to transmit at time t picks a message from the available emulation dictionary $\mathscr{E}(t)$, and sends such a message to the destination. The message is chosen uniformly at random, so that the probability of a particular message is simply $1/|\mathscr{E}(t)|$.

The corresponding evolution of the empirical dictionaries, for any subnet $\mathcal{B}$ of $\mathcal{B}_{\rm tot}$, is easily obtained as follows. Given the empirical dictionary $\mathscr{D}_{\mathcal{B}}(t)$, the empirical dictionary $\mathscr{D}_{\mathcal{B}}(t+\tau)$ is obtained by adding the *distinct* messages not contained in $\mathscr{D}_{\mathcal{B}}(t)$, which have been selected during the interval τ by the bots belonging to $\mathcal{B}$.

It is now useful to derive a formal characterization of the Botnet Message Innovation Rate, by introducing the following function:

$$\mathscr{R}(\alpha,\lambda) \triangleq \frac{\alpha\,\lambda}{\alpha+\lambda} \tag{4.5}$$

A first result provides a closed-form expression for the MIR of a botnet.

Theorem 1 (Botnet MIR)**.** *Consider a botnet $\mathcal{B}_{\rm tot}$ launching a DDoS attack, where the node transmission policies are either synchronous with constant transmission rate, or independent Poisson processes, with rates λ_u, for $u \in \mathcal{B}_{\rm tot}$. Consider a subset of the botnet $\mathcal{B} \subseteq \mathcal{B}_{\rm tot}$. Let $\mathscr{E}(t)$ be the emulation dictionary available to the botnet, with emulation dictionary rate α, and let $\mathscr{D}_{\mathcal{B}}(t)$ be the empirical dictionary of the subnet $\mathcal{B}$ at time t. Then, the message innovation rate of $\mathcal{B}$ is:*

$$\frac{|\mathscr{D}_{\mathcal{B}}(t)|}{t} \xrightarrow{\rm p} \rho_{\mathcal{B}} = \mathscr{R}(\alpha,\lambda_{\mathcal{B}}) \tag{4.6}$$

where $\lambda_{\mathcal{B}} = \sum_{u\in\mathcal{B}} \lambda_u$ is the aggregate transmission rate of the considered botnet subset.

Proof. See Appendix A.2. □

Remark I. From (4.5) and (4.6) we see that increasing the EDR α and/or the transmission rate λ corresponds to increasing the MIR. Besides, the MIR is always smaller[1] than $\min(\alpha,\lambda)$, which makes sense, since the number of new messages can exceed neither the number of messages in the emulation dictionary ($\mathscr{R}(\alpha,\lambda) \leq \alpha$), nor the overall number of transmitted messages ($\mathscr{R}(\alpha,\lambda) \leq$

[1]For $x > 0$ and $y > 0$, one has $x/(x+y) \leq 1$.

λ). Notably, the quantity $\min(\alpha, \lambda)$ is the MIR corresponding to a practical scheme where the patterns are obtained by taking sequentially (in a deterministic way) the messages of the emulation dictionary. With such a scheme, if $\alpha > \lambda$, a new message can be always found in $\mathscr{E}(t)$, and the maximum rate of distinct messages is λ. Likewise, if $\lambda > \alpha$, all messages in $\mathscr{E}(t)$ can be selected, along with some unavoidable repetitions, and the maximum rate of distinct messages is α.

Remark II. As α goes to infinity, the MIR converges to λ. In fact, as the number of messages in the emulation dictionary goes to infinity, each transmission would correspond with high probability to a new message, and the MIR will eventually reach the maximum allowable value λ. Likewise, as λ goes to infinity, we see that the MIR converges to α. In fact, as the number of sent messages goes to infinity, the emulation dictionary is completely spanned, and the MIR will eventually saturate to its maximum allowable value α.

Remark III. The MIR is symmetric in α and λ, implying that both quantities, even if they have a completely different practical meaning, play the same role as regards their effect on the MIR. In particular, we can write $\mathscr{R}(\alpha, \lambda) = (1/\alpha + 1/\lambda)^{-1}$, which reveals that the rate $\mathscr{R}(\alpha, \lambda)$ can be represented as the inverse of a time interval given by the sum of the average time between two messages available in the emulation dictionary, $1/\alpha$, and the average time between two transmissions, $1/\lambda$.

Remark IV. For strictly positive α and λ we have:

$$\mathscr{R}(\alpha, \lambda_1) + \mathscr{R}(\alpha, \lambda_2) > \mathscr{R}(\alpha, \lambda_1 + \lambda_2) \tag{4.7}$$

The latter inequality can be straightforwardly checked by exploiting the definition of $\mathscr{R}(\alpha, \lambda)$ in (4.5). Such inequality can be explained in the light of the physical interpretation of Theorem 1. In fact, the LHS in (4.7) corresponds to the MIR of a botnet made of two subnets: *i*) featuring transmission rates λ_1 and λ_2, respectively, and *ii*) picking messages from two *disjoint* dictionaries, each one with EDR equal to α. In contrast, the RHS corresponds to the

MIR of a botnet made of two subnets, still featuring transmission rates λ_1 and λ_2, but picking messages from a *common* dictionary with EDR α. Hence, the lower bound follows.

Remark V. The focus is on *genuinely-distributed* DoS attacks where the number of bots is large, *and* the transmission rate of each bot is not anomalous. For comparison purposes, let us consider another DDoS strategy, where B_{tot} *disjoint* (say, equal-size) portions of the emulation dictionary are disseminated through the botnet. Assuming for simplicity that all bots have unitary transmission rates, the MIR of user u, and the MIR of the whole botnet will be, respectively,

$$\rho_u = \frac{\alpha}{\alpha + B_{\mathrm{tot}}}, \quad \rho_{\mathcal{B}_{\mathrm{tot}}} = \sum_{u \in \mathcal{B}_{\mathrm{tot}}} \rho_u = \frac{\alpha\, B_{\mathrm{tot}}}{\alpha + B_{\mathrm{tot}}}, \tag{4.8}$$

where the first relationship follows from Theorem 1, while the second relationship follows from disjointness of the emulation (and, hence, of the empirical) dictionaries.

On the other hand, for our *coordinated* DDoS with *common* emulation dictionary, Theorem 1 gives:

$$\rho_u = \frac{\alpha}{\alpha + 1}, \quad \rho_{\mathcal{B}_{\mathrm{tot}}} = \frac{\alpha\, B_{\mathrm{tot}}}{\alpha + B_{\mathrm{tot}}}. \tag{4.9}$$

Notably, the rightmost formulas in (4.8) and (4.9) reveal that the MIR for the case of disjoint dictionaries *is the same as the MIR of a botnet using a common emulation dictionary.* On the other hand, the leftmost formulas in (4.8) and (4.9) reveal that the MIR of a single bot for the case of disjoint dictionaries *is approximately* B_{tot} *times smaller than the MIR of a single bot for the case of a common emulation dictionary.* Such a reduced degree of innovation matches the observations reported below (5.1), concerning the flaws of deterministic DDoS attacks based on disjoint emulation dictionaries.

Remark VI. Assume that the traffic analyst must estimate α based on the patterns collected from a certain subnet $\mathcal{S}$. From (4.5) and (4.6), we have $\alpha = \lambda_{\mathcal{S}}\, \rho_{\mathcal{S}} / (\lambda_{\mathcal{S}} - \rho_{\mathcal{S}})$. Accordingly, a reasonable estimator of α can be obtained by replacing ρ and λ with their empirical counterparts, yielding:

$$\hat{\alpha}_{\mathcal{S}}(t) \triangleq \frac{\hat{\lambda}_{\mathcal{S}}(t)\, \hat{\rho}_{\mathcal{S}}(t)}{\hat{\lambda}_{\mathcal{S}}(t) - \hat{\rho}_{\mathcal{S}}(t)} \tag{4.10}$$

In view of Theorem 1, such estimator converges in probability to α as t goes to infinity, for any $\mathcal{S} \subseteq \mathcal{B}_{\text{tot}}$.

In contrast, when dealing with normal users, such an interpretation fails in general, since: *i*) a limiting value α does not necessarily exist, and *ii*) the generative mechanism of normal patterns is not necessarily interpreted in terms of random picking from an emulation dictionary. Nevertheless, the quantity $\hat{\alpha}_{\mathcal{S}}(t)$ can be meaningfully defined also for arbitrary subnets (i.e., composed also, or even exclusively, by normal users), since it represents the ratio between the empirical rate of "distinct" messages $\hat{\rho}_{\mathcal{S}}(t)$, and the empirical rate of "repeated" messages $\hat{\lambda}_{\mathcal{S}}(t) - \hat{\rho}_{\mathcal{S}}(t)$, scaled[2] by the empirical transmission rate $\hat{\lambda}_{\mathcal{S}}(t)$. Such an interpretation is useful since it is now independent from the particular model adopted (transmission scheduling, botnet or normal behavior, etc.). In the following, even when dealing with arbitrary subnets, I shall loosely refer to $\hat{\alpha}_{\mathcal{S}}(t)$ as the *empirical*, or *estimated* EDR.

Finally, exploiting (4.5) and (4.10), the empirical MIR $\hat{\rho}_{\mathcal{S}}(t)$, for an *arbitrary* subnet $\mathcal{S}$, can be expressed as:

$$\hat{\rho}_{\mathcal{S}}(t) = \mathscr{R}(\hat{\alpha}_{\mathcal{S}}(t), \hat{\lambda}_{\mathcal{S}}(t)) \tag{4.11}$$

4.5 The Botnet Identification Condition (BIC)

The coordination implied in the *distributed* DoS attack introduces a certain amount of correlation between the empirical dictionaries of the bots, due to the common emulation dictionary where mes-

[2]The scaling simply corresponds to expressing the result on a per-time-unit basis, rather than on a per-transmission basis.

sages are selected. In contrast, the empirical dictionaries of two normal users are expected to be weakly correlated, due to independence among their activities. Likewise, the empirical dictionaries of a bot and of a normal user are expected to be weakly correlated, since the network employed by the botmaster to acquire the emulation dictionary is usually not part of the network monitored by the traffic analyst. On the other hand, even in the presence of normal (thus, *independent*) users, it is realistic to assume a certain degree of *physiological* correlation among the users' activities. Distinct users can reasonably share parts of their dictionaries, e.g., their surfing activities might partly overlap, due to common interests, popular web-pages, peculiar structure of the destination of interest, etc. Similar considerations apply when dealing with a subset of the botnet and a subnet made only of normal users. Given the very limited amount of information and assumptions made, and according to the above discussion, any meaningful strategy to discriminate a normal from a malicious behavior, cannot but be based on the degree of dependence among the users. In the considered setting, a convenient way to measure the degree of dependence is provided by the empirical message innovation rate in (4.3). However, the mere availability of a good network indicator does not provide a *quantitative* way to discriminate normal users from bots. In order to design an algorithm for botnet identification, we need to define a proper *identification threshold*. To this aim, it is possible to use as reference case for a malicious behavior, the MIR corresponding to the activity performed by a botnet. In order to understand how such operation can be implemented, let us start by considering the case that we must decide whether users 1 and 2 belong to a botnet. Assume for now that the empirical EDRs of the two users obtained through (4.10) are comparable (the explicit dependence on t being suppressed, for ease of notation, here and in the forthcoming discussion):

$$\hat{\alpha}_1 \approx \hat{\alpha}_2 \approx \hat{\alpha}. \tag{4.12}$$

When both users belong to a botnet, in view of Theorem 1, for t large enough we can write:

$$\hat{\rho}_{\{1,2\}} \approx \mathscr{R}(\hat{\alpha}, \hat{\lambda}_1 + \hat{\lambda}_2) \triangleq \hat{\rho}_{\text{bot}}. \tag{4.13}$$

Moreover, *irrespectively of the users' nature*, the empirical MIR of the aggregate subnet $\{1,2\}$ can be upper bounded by the MIR corresponding to disjoint dictionaries, namely,

$$\begin{aligned} \hat{\rho}_{\{1,2\}} &\leq \hat{\rho}_1 + \hat{\rho}_2 = \mathscr{R}(\hat{\alpha}_1, \hat{\lambda}_1) + \mathscr{R}(\hat{\alpha}_2, \hat{\lambda}_2) \triangleq \hat{\rho}_{\text{sum}} \\ &\approx \mathscr{R}(\hat{\alpha}, \hat{\lambda}_1) + \mathscr{R}(\hat{\alpha}, \hat{\lambda}_2), \end{aligned} \tag{4.14}$$

where the second equality follows from (4.11), while the approximate equality follows from (4.12). Since from (4.7) we know that $\hat{\rho}_{\text{bot}} < \hat{\rho}_{\text{sum}}$, it makes sense to introduce a threshold lying between the two points $\hat{\rho}_{\text{bot}}$ and $\hat{\rho}_{\text{sum}}$, formally, for $\epsilon \in (0,1)$:

$$\hat{\rho}_{\text{bot}} < \gamma = \hat{\rho}_{\text{bot}} + \epsilon(\hat{\rho}_{\text{sum}} - \hat{\rho}_{\text{bot}}) < \hat{\rho}_{\text{sum}}. \tag{4.15}$$

When the two users belong to a botnet, from (4.13) it is possible to see that, for large t, the empirical MIR $\hat{\rho}_{\{1,2\}}$ converges to the value $\hat{\rho}_{\text{bot}}$. On the other hand, using Theorem 1, it is easy to verify that $\hat{\rho}_{\text{sum}} - \hat{\rho}_{\text{bot}}$ converges in probability to a positive quantity, which implies that, for *any* $\epsilon > 0$, as time elapses, the empirical MIR will stay sooner (higher ϵ) or later (lower ϵ) *below* the threshold γ, yielding:

$$1 \text{ AND } 2 \text{ are bots} \Rightarrow \hat{\rho}_{\{1,2\}} < \gamma \tag{4.16}$$

Consider now the case that at least one user is normal. Were the dictionaries of the two users perfectly disjoint, we would clearly observe, for any $\epsilon \in (0,1)$, that $\hat{\rho}_{\{1,2\}} \approx \hat{\rho}_{\text{sum}} > \gamma$. However, we already noticed that some correlation is expected to exist even among normal users, or among normal users and bots. It is also natural to assume that such a correlation is weaker than the correlation exhibited by groups of bots, since the latter are choosing their messages from one and the same underlying dictionary.[3] Accordingly, one might expect that, when at least one user is normal, for sufficiently small ϵ, the empirical MIR still stays above the threshold, namely:

$$1 \text{ OR } 2 \text{ are normal} \Rightarrow \hat{\rho}_{\{1,2\}} > \gamma \tag{4.17}$$

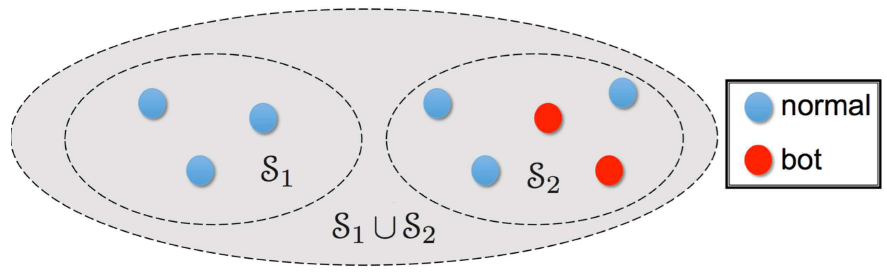
In summary, if the empirical MIR stays below γ, it is possible to declare that the two users form a botnet, otherwise, we can declare that at least one user is normal.

Two main points emerge. First, the essential feature enabling a successful discrimination is the assumption in (4.17), which accordingly plays the role of a Botnet Identification Condition (BIC). Second, the determination of the threshold γ relies on a tuning parameter ϵ, which is in principle related to the intrinsic (and unknown) properties of the normal traffic patterns. Remarkably, the experimental study conducted in the forthcoming section will show clearly that: *i*) the BIC can be safely used, and *ii*) the choice of ϵ is by no means critical, even in the non-parametric scenario where no prior information about the normal users' behavior is available.

Unfortunately, not all that glitters is gold. There is an important complication that has been deliberately overlooked so far. According to the above explanation, we need to compare the empirical MIR to the MIR of a *reference* botnet. However, a botnet is characterized by a *common* underlying EDR α, while in practice we shall typically have $\hat{\alpha}_1 \neq \hat{\alpha}_2$ (especially when at least one user is normal), implying that the approximation in (4.12) is unsupported. One approach could be that of discarding *ab initio* the botnet hypothesis whenever $\hat{\alpha}_1$ and $\hat{\alpha}_2$ are too dissimilar. The qualification of being "too dissimilar" translates into the appearance of some extra tuning parameter, possibly depending on time, which we want definitely to avoid.

Another possibility is clearly that of choosing as reference EDR

[3]In making such assumption, it is assumed that the specific mechanism used to build normal patterns has a minor influence.



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

Figure 4.4: The Botnet Identification Condition and relationships among ρ parameters.

some intermediate value comprised between $\hat{\alpha}_1$ and $\hat{\alpha}_2$. In this connection, it is useful to remark that the naïve choice of the arithmetic average does not work for the following reason. It can be simply verified that, in general, there exist values of $\lambda_1, \lambda_2, \alpha_1, \alpha_2 \in \mathbb{R}^+$ for which $\mathscr{R}(\lambda_1, \alpha_1) + \mathscr{R}(\lambda_2, \alpha_2) < \mathscr{R}(\lambda_1 + \lambda_2, 1/2(\alpha_1 + \alpha_2))$, implying that the empirical MIR, *even for the case of disjoint dictionaries*, is not necessarily greater than the MIR of a botnet with reference EDR given by the arithmetic average of $\hat{\alpha}_1$ and $\hat{\alpha}_2$. In Fig. 4.4 an example is offered, aimed at understanding how to build the BIC across a network made of two subnets (S_1 and S_2), through the various ρ parameters. A systematic way to select a proper intermediate value is substantially more involved, and is the object of the following subsection.

4.5.1 Replacement and Reassignment procedure

Let us consider two (disjoint) subnets $\mathcal{S}_1$ and $\mathcal{S}_2$, with focus on the case that at least one of them is composed only by normal users, with $\hat{\alpha}_{\mathcal{S}_1} \neq \hat{\alpha}_{\mathcal{S}_2}$. Recall that we are considering a fixed time t, and that the explicit dependence of all quantities upon t is suppressed for ease of notation.

Since a botnet has *common* underlying EDR, and since we want to compare the behavior of $\mathcal{S}_1 \cup \mathcal{S}_2$ to that of a botnet, it would be useful to envisage a new pair of traffic patterns for $\mathcal{S}_1$ and $\mathcal{S}_2$ possessing the following characteristics:
i) The individual EDRs of $\mathcal{S}_1$ and $\mathcal{S}_2$ are equal, namely (superscript $'$ refers to the "new" patterns),

$$\hat{\alpha}'_{\mathcal{S}_1} = \hat{\alpha}'_{\mathcal{S}_2} = \hat{\alpha}'. \tag{4.18}$$

ii) The transmission rate and the MIR of the network $\mathcal{S}_1 \cup \mathcal{S}_2$ coincide with those of the original traffic patterns.

It is now useful to illustrate a Replacement and Reassignment (RR) procedure, which finds such a new pair starting from the original pattern configuration. Such a procedure relies on the intuitive consideration that, if some messages are reassigned from the subnet with highest EDR to the other subnet, the resulting EDRs tend to keep each other closer. In order to avoid misunderstandings, it is worth remarking that *the RR procedure does not correspond to any real/physical operations made on the traffic patterns. The RR procedure is a conceptual experiment used to demonstrate that it is possible to construct two patterns possessing the aforementioned requirements i) and ii).*

The RR procedure goes as follows — see Fig. 4.5 for a pictorial illustration.

1. Replacement of repeated messages. The traffic pattern of a subnet $\mathcal{S}$ contains $|\mathscr{D}_{\mathcal{S}}|$ distinct messages, the remaining $N_{\mathcal{S}} - |\mathscr{D}_{\mathcal{S}}|$ ones being repetitions of messages contained in $\mathscr{D}_{\mathcal{S}}$. The first step of the procedure amounts to replacing such $N_{\mathcal{S}} - |\mathscr{D}_{\mathcal{S}}|$ messages by one and the same message, say it m^*, contained in $\mathscr{D}_{\mathcal{S}}$. The replacement is applied to both subnets $\mathcal{S}_1$ and $\mathcal{S}_2$, with the corresponding replacing messages being m_1^* and m_2^*. Since replacement acts only on the message content, the transmission rates do not change. Moreover, since replacement leaves unaltered the number of distinct messages within each subnet, the MIR of the subnets,

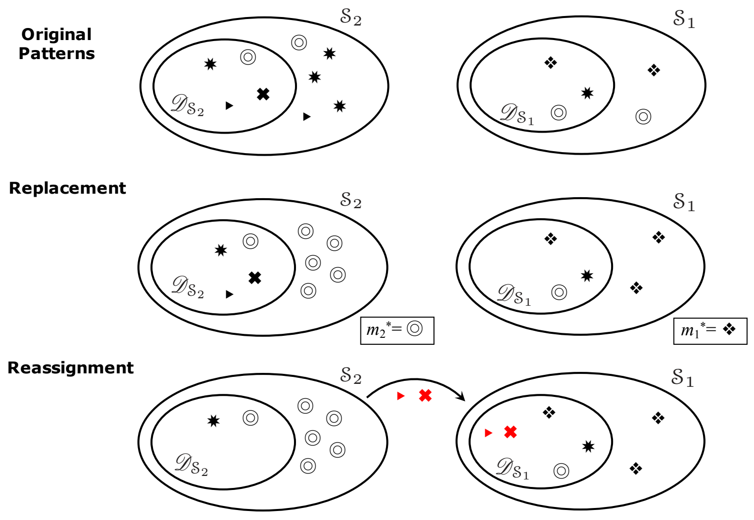


Figure 4.5: The RR procedure, pictorial exemplification.

and the MIR of $\mathcal{S}_1 \cup \mathcal{S}_2$, are unaltered.[4]
2. Reassignment of messages. Some messages will be reassigned from one subnet to the other subnet (only in one direction, namely, either from $\mathcal{S}_2$ to $\mathcal{S}_1$ or from $\mathcal{S}_1$ to $\mathcal{S}_2$). For the sake of clarity, assume that $\mathcal{S}_2$ is "passing" some of its messages to $\mathcal{S}_1$, with the prescription that the replacing message m_2^* is never passed. Since, after replacement, all messages different from m_2^* appear only once in the pattern of $\mathcal{S}_2$, we see that *all* messages passed to $\mathcal{S}_1$ are necessarily distinct. The rate of messages (number of messages normalized to the current time t) that are reassigned from $\mathcal{S}_2$ to $\mathcal{S}_1$ is denoted by Δ. Accordingly, a negative Δ will correspond to the converse situation where $\mathcal{S}_1$ passes some of its messages to $\mathcal{S}_2$. As a result, the transmission rates of the pattern configuration after reassignment are:

$$(\hat{\lambda}'_{\mathcal{S}_1}, \hat{\lambda}'_{\mathcal{S}_2}) = (\hat{\lambda}_{\mathcal{S}_1} + \Delta, \hat{\lambda}_{\mathcal{S}_2} - \Delta). \tag{4.19}$$

Moreover, since the correlation between the two patterns is weak (recall that one of the subnets is composed only by normal users),

[4]The MIR is determined only by the content of the empirical dictionaries.

I assume that it is always possible to reassign messages that do not belong to the intersection of the two empirical dictionaries. Such assumption, along with the fact that all passed messages are distinct, implies that, in terms of individual MIRs, what is lost by a subnet is exactly gained by the other subnet. Formally:

$$(\hat{\rho}'_{\mathcal{S}_1}, \hat{\rho}'_{\mathcal{S}_2}) = (\hat{\rho}_{\mathcal{S}_1} + \Delta, \hat{\rho}_{\mathcal{S}_2} - \Delta). \tag{4.20}$$

Note that not all values of Δ are admissible. For instance, if messages from $\mathcal{S}_2$ are reassigned to $\mathcal{S}_1$, the rate of reassigned messages cannot exceed the rate of distinct messages owned by $\mathcal{S}_2$, namely, $\Delta \leq \hat{\rho}_{\mathcal{S}_2}$. Likewise, in the converse case, $-\Delta \leq \hat{\rho}_{\mathcal{S}_1}$, finally yielding:[5]

$$-\hat{\rho}_{\mathcal{S}_1} \leq \Delta \leq \hat{\rho}_{\mathcal{S}_2}. \tag{4.21}$$

Moreover, since the reassignment changes only the "owner" of a given message, the MIR of the *aggregate* network $\mathcal{S}_1 \cup \mathcal{S}_2$ is left unaltered, namely, $\hat{\rho}'_{\mathcal{S}_1 \cup \mathcal{S}_2} = \hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2}$.

3. Choice of Δ for the equilibrium condition. At the end of the reassignment procedure, the new EDRs corresponding to $\mathcal{S}_1$ and $\mathcal{S}_2$ become, respectively, $\hat{\alpha}'_{\mathcal{S}_1} = \hat{\lambda}'_{\mathcal{S}_1}\hat{\rho}'_{\mathcal{S}_1}(\hat{\lambda}'_{\mathcal{S}_1} - \hat{\rho}'_{\mathcal{S}_1})$, and $\hat{\alpha}'_{\mathcal{S}_2} = \hat{\lambda}'_{\mathcal{S}_2}\hat{\rho}'_{\mathcal{S}_2}/(\hat{\lambda}'_{\mathcal{S}_2} - \hat{\rho}'_{\mathcal{S}_2})$, where (4.10) has been exploited. In order to get a common reference EDR $\hat{\alpha}'$, it is enforced the condition in (4.18), which, using (4.19) and (4.20) into the latter two equations, amounts to seek a value $\Delta^\star$ such that:

$$\hat{\alpha}' = \frac{(\hat{\lambda}_{\mathcal{S}_1} + \Delta^\star)(\hat{\rho}_{\mathcal{S}_1} + \Delta^\star)}{\hat{\lambda}_{\mathcal{S}_1} - \hat{\rho}_{\mathcal{S}_1}} = \frac{(\hat{\lambda}_{\mathcal{S}_2} - \Delta^\star)(\hat{\rho}_{\mathcal{S}_2} - \Delta^\star)}{\hat{\lambda}_{\mathcal{S}_2} - \hat{\rho}_{\mathcal{S}_2}}, \tag{4.22}$$

with the additional prescription that condition (4.21) is met. Therefore, the explicit formula for $\Delta^\star$ is found by solving a quadratic equation, and by simple algebra it can be verified that

[5]Actually, since we exclude the replacing messages m_1^* or m_2^* from the reassignment procedure, a subnet cannot pass all its distinct messages. However, for large t the contribution of a *single* message becomes irrelevant.

the solution fulfilling (4.21) is:

$$\begin{aligned}\Delta^\star &= \frac{\hat{\lambda}_{\mathcal{S}_1}\hat{\lambda}_{\mathcal{S}_2} - \hat{\rho}_{\mathcal{S}_1}\hat{\rho}_{\mathcal{S}_2}}{(\hat{\lambda}_{\mathcal{S}_1} - \hat{\rho}_{\mathcal{S}_1}) - (\hat{\lambda}_{\mathcal{S}_2} - \hat{\rho}_{\mathcal{S}_2})} \\ &- \frac{\sqrt{(\hat{\lambda}_{\mathcal{S}_1} - \hat{\rho}_{\mathcal{S}_1})(\hat{\lambda}_{\mathcal{S}_2} - \hat{\rho}_{\mathcal{S}_2})(\hat{\lambda}_{\mathcal{S}_1} + \hat{\rho}_{\mathcal{S}_2})(\hat{\lambda}_{\mathcal{S}_2} + \hat{\rho}_{\mathcal{S}_1})}}{(\hat{\lambda}_{\mathcal{S}_1} - \hat{\rho}_{\mathcal{S}_1}) - (\hat{\lambda}_{\mathcal{S}_2} - \hat{\rho}_{\mathcal{S}_2})}. \end{aligned} \tag{4.23}$$

From (4.22), it is easily verified that a positive $\Delta^\star$ corresponds to $\hat{\alpha}_{\mathcal{S}_1} < \hat{\alpha}' < \hat{\alpha}_{\mathcal{S}_2}$ (while the latter two inequalities are reversed when $\Delta^\star < 0$), implying that the subnet with the highest EDR "passes" a fraction of its messages to the other subnet. In summary, one can conclude that:

$$\min(\hat{\alpha}_{\mathcal{S}_1}, \hat{\alpha}_{\mathcal{S}_2}) \le \hat{\alpha}' \le \max(\hat{\alpha}_{\mathcal{S}_1}, \hat{\alpha}_{\mathcal{S}_2}) \tag{4.24}$$

According to the above explanation, when at least one of the subnets is composed only by normal users, it is possible to write:

$$\begin{aligned}\hat{\rho}_{\text{sum}}(\mathcal{S}_1, \mathcal{S}_2) &\triangleq \hat{\rho}_{\mathcal{S}_1} + \hat{\rho}_{\mathcal{S}_2} \overset{\text{(a)}}{=} \hat{\rho}'_{\mathcal{S}_1} + \hat{\rho}'_{\mathcal{S}_2} \\ &\overset{\text{(b)}}{=} \mathscr{R}(\hat{\alpha}', \hat{\lambda}'_{\mathcal{S}_1}) + \mathscr{R}(\hat{\alpha}', \hat{\lambda}'_{\mathcal{S}_2}) \\ &\overset{\text{(c)}}{>} \mathscr{R}(\hat{\alpha}', \hat{\lambda}'_{\mathcal{S}_1} + \hat{\lambda}'_{\mathcal{S}_2}) \\ &\overset{\text{(d)}}{=} \mathscr{R}(\hat{\alpha}', \hat{\lambda}_{\mathcal{S}_1} + \hat{\lambda}_{\mathcal{S}_2}) \triangleq \hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2), \end{aligned} \tag{4.25}$$

where (a) follows from (4.20); (b) follows from (4.11); (c) follows from (4.7); and (d) follows from (4.19). On the other hand, when $\mathcal{S}_1$ and $\mathcal{S}_2$ form a botnet, Theorem 1 implies that, for t large enough, $\hat{\alpha}_{\mathcal{S}_1} \approx \hat{\alpha}_{\mathcal{S}_2} \approx \alpha$, which in turn implies that $\hat{\alpha}' \approx \alpha$ in view of (4.24). Therefore, in this case the inequality $\hat{\rho}_{\text{sum}}(\mathcal{S}_1, \mathcal{S}_2) > \hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2)$ is justified by the approximations: $\hat{\rho}_{\text{sum}}(\mathcal{S}_1, \mathcal{S}_2) \approx \mathscr{R}(\alpha, \lambda_{\mathcal{S}_1}) + \mathscr{R}(\alpha, \lambda_{\mathcal{S}_2})$ and $\hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2) \approx \mathscr{R}(\alpha, \lambda_{\mathcal{S}_1} + \lambda_{\mathcal{S}_2})$.

In conclusion, it has been show that, for *arbitrary transmission*

$$\boxed{\rho_{\text{bot}} < \underbrace{\rho_{\text{bot}} + \epsilon(\rho_{\text{sum}} - \rho_{\text{bot}})}_{\text{Threshold } \gamma} < \rho_{\text{sum}}}$$

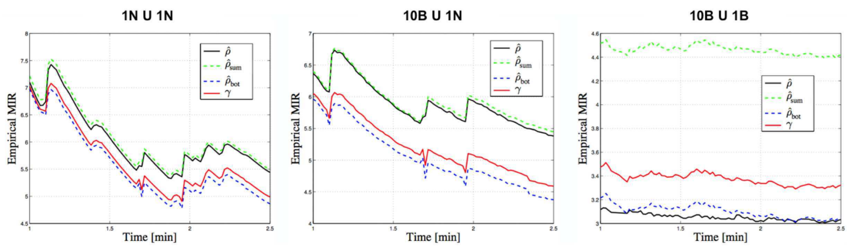


Figure 4.6: Time evolution of the empirical message innovation rate $\hat{\rho}$ (solid, black), compared to the identification threshold γ (solid, red). For comparison purposes, the upper bound corresponding to the case of disjoint dictionaries, $\hat{\rho}_{\text{sum}}$ (dashed, green), and the lower bound corresponding to the botnet case, $\hat{\rho}_{\text{bot}}$ (dashed, magenta) are displayed. Moving from left to right, the different panels refer to *i*) the union of two normal users; *ii*) the union of a botnet of size 10 and a normal user; and *iii*) the union of a botnet of size 10 and a bot.

schedulings and message-picking policies, the empirical MIR of a botnet with reference EDR value (4.22) does *always* provide a lower bound to the sum of individual MIRs.[6]

4.5.2 Threshold setting

Let us introduce an intermediate threshold lying between the lower bound and the upper bound in (4.25), namely, for $\epsilon \in (0, 1)$,

$$\gamma(\mathcal{S}_1, \mathcal{S}_2) = \hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2) + \epsilon\,[\hat{\rho}_{\text{sum}}(\mathcal{S}_1, \mathcal{S}_2) - \hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2)] \qquad (4.26)$$

When $\mathcal{S}_1$ and $\mathcal{S}_2$ form a botnet, from Theorem 1 it is immediately seen (recall that $\hat{\alpha}'$ will converge to the true α) that $\hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2} <$

[6]The aforementioned result does not relate in any way to the deterministic or Poisson scheduling and to the random message picking that characterize the class of DDoS attacks considered in the present work.

$\gamma(\mathcal{S}_1, \mathcal{S}_2)$ as $t \to \infty$.

When at least one of the subnets is made of normal users, the degree of dependence among their patterns is low. Since *i*) it has been shown that there exist two patterns, *with common EDR*, $\hat{\alpha}'$, and with the same *joint* properties (overall transmission rate and MIR) of the original patterns; and *ii*) the RR procedure only replaces and/or reassigns messages, it is expected that the joint MIR of a botnet with EDR $\hat{\alpha}'$ is lower than $\hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2}$. Otherwise stated, it is reasonable to assume that $\hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2}$, even if not coinciding with the upper bound $\hat{\rho}_{\text{sum}}(\mathcal{S}_1, \mathcal{S}_2)$ in (4.25), is still sufficiently far from the lower bound $\hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2)$. These considerations, for small ϵ, implicitly define the following identification condition.

Botnet Identification Condition (BIC)

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two subnets with $\mathcal{S}_1 \bigcap \mathcal{S}_2 = \emptyset$. *If at least one of the subnets is composed only by normal users*:

$$\hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2} \geq \gamma(\mathcal{S}_1, \mathcal{S}_2) \tag{4.27}$$

It is worth remarking that the case of $\mathcal{S}_1$ arbitrary vs. $\mathcal{S}_2$ arbitrary is not dealt with. This is not unintentional, since, as it will be clear from Theorem 2, the two situations discussed are sufficient to devise a *consistent* botnet identification algorithm.

In summary, the following recipe is derived:

$$\mathcal{S}_1 \text{ AND } \mathcal{S}_2 \text{ contain only bots} \quad \Rightarrow \quad \hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2} < \gamma(\mathcal{S}_1, \mathcal{S}_1), \tag{4.28}$$

$$\mathcal{S}_1 \text{ OR } \mathcal{S}_2 \text{ contain only normal users} \quad \Rightarrow \quad \hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2} \geq \gamma(\mathcal{S}_1, \mathcal{S}_1). \tag{4.29}$$

Figure 4.6 shows the significance of the BIC. The normal users' activity refers to a monitoring campaign conducted in a testbed environment. The bots' activity has been generated according to the model described in Sec. 4.4. The details of such a campaign will be given in the forthcoming section. In all the three panels are displayed: the empirical MIR, the threshold γ in (4.26), along with its upper ($\hat{\rho}_{\text{sum}}$) and lower ($\hat{\rho}_{\text{bot}}$) bounds. An observation window of 2.5 min is considered. All the relevant quantities are updated each 1 s, and both quantities are displayed as functions of time, in the interval between 1 and 2.5 min.

In the leftmost panel, the case of a pair of normal users is addressed. It is possible to see that the MIR stays (slightly) below the upper bound, meaning that a certain degree of correlation exists. However, *the MIR stands clear above the threshold, as prescribed by (4.29), and confirming the validity of the BIC.*

In the middle panel, the two subnets under test, $\mathcal{S}_1$ and $\mathcal{S}_2$, are a botnet of size 10, and a normal user, respectively. Conclusions similar to those pertaining to a normal-normal pairing can be drawn, substantiating again the BIC. It is possible further see that, at the beginning of the observation window, the activities of the two subnets are almost independent, i.e., the MIR essentially matches the upper bound. As time elapses, a certain degree of correlation appears, *but the MIR still stays above the threshold.*

Finally, in the rightmost panel, the case of a botnet/bot interaction is addressed. One can see that the empirical MIR: *i*) approaches, as time elapses, the quantity $\hat{\rho}_{\text{bot}}$, in perfect agreement with Theorem 1, and *ii*) stands clear below the threshold, in perfect agreement with (4.28).

In summary, the picture obtained from the above analysis reveals that the theoretical findings of Theorem 1, as well as the conjectured behavior of the normal users implied by the BIC, are confirmed *over the experimental network traces.*[7]

Before dwelling on the detailed description of the botnet identification algorithm, it is worth commenting on a possible limitation of the proposed approach. There might be particular situations where the BIC is violated because some normal users, even if acting in uncoordinated manner, exhibit a certain degree of correlation. Since the identification algorithm is presumed to discover dependencies among traffic patterns, it could erroneously declare the (honest) machines as bots. One example of the aforementioned situations relates to the influence of traffic patterns of distinct

[7]The experiments have been repeated for many pairs of normal users. For illustrative purposes, in Fig. 4.6 is reported one sample of such experiments, which is representative of the observed behavior. The quantitative analysis addressing the average behavior across users is deferred to Sec. 4.7.

machines that perform the same automated update mechanisms of programs and/or operating systems, i.e., during daily/weekly update cycle.

4.6 The BotBuster algorithm

It is the time to focus on the derivation of the inference algorithm aimed at disclosing a botnet possibly hidden in the network. The *BotBuster* algorithm is described by the pseudo-code reported in 4.6, and basically exploits the fact that, given two disjoint subnets, the BIC allows to discriminate the situation where both subnets are part of a botnet, from the situation where at least one of them is made of normal users. It will show that the proposed algorithm possesses the fundamental requirement of *consistency*, namely, *the guarantee that the botnet is correctly identified as t grows.*

Let us examine how the algorithm works. First, note that a botnet made of one user, besides making little sense in practice, is non-identifiable by any means, since I assumed that the characteristics of the messages at a single-user level do not reveal any special information. Now, at the beginning of the algorithm, user 1 is initially declared as a bot, namely, $\hat{\mathcal{B}} = \{1\}$. Then, it is checked whether users 1 and 2 form a botnet. If so, $\hat{\mathcal{B}} = \{1, 2\}$ is taken as the current botnet estimate. If not, $\hat{\mathcal{B}} = \{1\}$ is retained. Then, it is checked whether the currently estimated botnet $\hat{\mathcal{B}}$ forms a bot with user 3, and so on. At the end of the inner loop, the algorithm ends up with an estimate $\hat{\mathcal{B}}$. If the cardinality of the estimated set is greater than one, it is taken as a current estimate.

The procedure is then restarted by choosing user 2 as initial pivot, and sequentially checking the remaining users as explained before. At the end of the inner loop, the algorithm ends up with another estimate $\hat{\mathcal{B}}$. If the cardinality of the estimated set is greater than one *and* greater than the cardinality of the previously estimated set[8], then it is taken as a current estimate. Otherwise,

[8]When t is large and the BIC is *perfectly* verified, the inner loop ends with either an empty set or the true botnet. Thus, selecting the estimate with the

```
Algorithm: $\hat{\mathcal{B}}_{\text{new}}$=BotBuster

$\mathcal{N} = \{1, 2, \ldots, N\}$; $\hat{\mathcal{B}}_{\text{new}} = \emptyset$;
for $b_0 \in \mathcal{N}$ do
    $\hat{\mathcal{B}} = \{b_0\}$;
    for $j \in \mathcal{N} \setminus \{b_0\}$ do
        if $\hat{\rho}(\hat{\mathcal{B}} \cup \{j\}) < \gamma(\hat{\mathcal{B}}, \{j\})$ then
            $\hat{\mathcal{B}} = \hat{\mathcal{B}} \bigcup \{j\}$;
        end
    end
    if $|\hat{\mathcal{B}}| > \max(1, |\hat{\mathcal{B}}_{\text{new}}|)$ then
        $\hat{\mathcal{B}}_{\text{new}} = \hat{\mathcal{B}}$;
    end
end
```

the previous estimate is retained. The procedure ends when all users have been scanned as pivots.

It is possible to see that, under the BIC, all checks performed by the algorithm will give eventually the right answer, with probability tending to 1 as $t \to \infty$. BotBuster is accordingly expected to provide a *consistent* botnet estimator, as will be stated and proved in the forthcoming Theorem 2. The algorithm complexity is $\mathcal{O}(N^2)$ (only pairwise checks are performed), which is definitely tolerable, since we are seeking, within a network of size N, a subset of unknown size that matches some prescribed conditions. Finally, the looping structure of the algorithm makes it naturally open to parallelization, which is especially important for large networks.

In order to quantify the algorithm performance, we need to choose some meaningful indicators. With reference to 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 BotBuster, it is useful to introduce

highest cardinality might appear redundant. Such operation is instead useful when operating under non-ideal conditions, as I shall explain soon.

the following performance indices:

$$\eta_{\text{bot}}(t) = \frac{\mathbb{E}[|\hat{\mathcal{B}}(t) \cap \mathcal{B}|]}{|\mathcal{B}|}, \quad \eta_{\text{nor}}(t) = \frac{\mathbb{E}[|\hat{\mathcal{B}}(t) \cap (\mathcal{N} \setminus \mathcal{B})|]}{|\mathcal{N} \setminus \mathcal{B}|}, \tag{4.30}$$

namely, the expected fraction of *correctly banned users* (i.e., discovered bots), and the expected fraction of incorrectly-banned users (i.e., normal users erroneously declared as bots). Clearly, $\eta_{\text{bot}}(t)$ (resp., $\eta_{\text{nor}}(t)$) is not defined when $\mathcal{B} = \emptyset$ (resp., when $\mathcal{B} = \mathcal{N}$). We would like to see $\eta_{\text{bot}}(t) \to 1$, and $\eta_{\text{nor}}(t) \to 0$ as t goes to infinity. Under the ideal assumption that the BIC is always verified, such requirement is in fact fulfilled, as stated in the following theorem.

Theorem 2 (Consistency of BotBuster)**.** *Consider a network* $\mathcal{N} = \{1, 2, \ldots, N\}$*, containing a botnet* $\mathcal{B}$*, with* $|\mathcal{B}| \neq 1$*, launching a randomized DDoS attack. The bots' transmission policies are either synchronous with constant transmission rate, or independent Poisson processes, while the normal users' transmission policies are arbitrary. Then, for any finite emulation dictionary rate* α*, the algorithm BotBuster is consistent, namely,*

$$\lim_{t\to\infty} \eta_{\text{bot}}(t) = 1, \qquad \lim_{t\to\infty} \eta_{\text{nor}}(t) = 0 \tag{4.31}$$

The claim for the case $\mathcal{B} = \emptyset$ *(resp.,* $\mathcal{B} = \mathcal{N}$*) is intended to hold with reference solely to* $\eta_{\text{nor}}(t)$ *(resp., to* $\eta_{\text{bot}}(t)$*).*

Proof. See Appendix A.3. □

Theorem 2 reveals that the botnet estimated by BotBuster converges to the *true* one as time elapses. The fundamental requirement enabling such result is the BIC validity. On the other hand, in real-world applications, the assumption that the BIC is verified *for all* normal/normal and botnet/normal interactions, as well as *for all* time epochs, is surely an *over-idealized* one. It cannot be excluded that, occasionally, two independent users feature an unusual degree of superposition between their empirical dictionaries, giving rise to spurious clusters of normal users that might

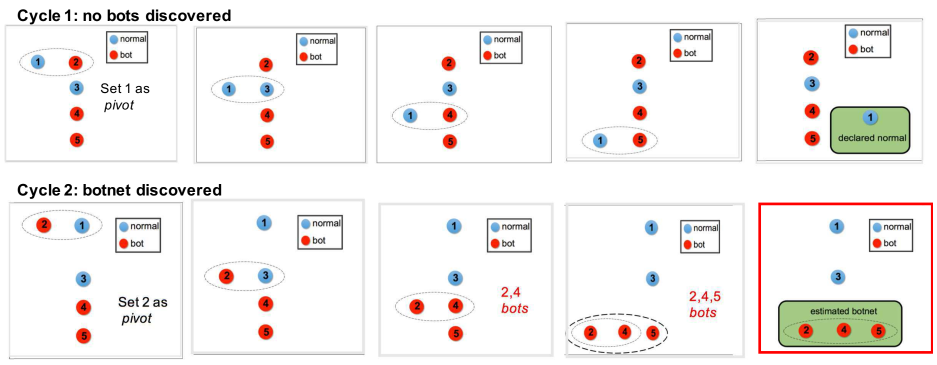


Figure 4.7: Two-cycle example of BotBuster algorithm: *i)* Start from a pivot node; *ii)* Perform comparison sequentially; *iii)* Iterate over pivot node. Because of not all checks are needed, the algorithm does not exhibit a combinatorial behavior and the complexity amounts to $O(N^2)$.

be erroneously included in the estimated botnet. What is expected to be true even in real-world applications, is that such cases are rare and that the clusters' cardinality is small. Now, since the algorithm selects the estimate $\hat{\mathcal{B}}$ with the *highest* cardinality, and since *distributed* DoS attacks with small botnet sizes make little sense, estimated botnets of unreasonably small cardinality should be easily ruled out by BotBuster. As a result, the final estimate is likely to contain the true botnet, plus (possibly) a small fraction of normal users. Thus, *even under non-ideal operation conditions*, it is expected that $\eta_{\text{bot}}(t) \to 1$ as $t \to \infty$, whereas $\eta_{\text{nor}}(t)$ possibly takes on some small value. Figure 4.7 reports a two-cycle example of BotBuster in order to show the basic functionality.

4.7 Numerical results and performance evaluation

As regards the measuring stage that precedes the botnet identification algorithm, the following pipeline has been adopted. Packets are preliminarily filtered by using a popular software package for packet capturing and network protocol analysis. At the output of such preliminary filtering stage: *i*) only the traffic directed to the destination that is being monitored is retained; *ii*) among the surviving packets, only the application-layer traffic is retained; *iii*) the resulting packets are divided on the basis of their source IP address, and are finally fed to the botnet identification algorithm. A popular e-commerce website has been selected as target destination of the attack. Clearly, the normal users have no attacking intent, they perform ordinary surfing activity. About 20 min of (application-layer) traffic have been collected, from 10 independent users, which were students and researchers working in our laboratory, and carrying on their surfing activity almost independently. In order to help understanding the nature and significance of the dataset, it is useful to report that the total number of TCP flows is about 26800, the median of flows across users is 2846, the minimum number of flows is 1042, the maximum number of flows is 3925, and the average packet size is 776 bytes. Supported by these numbers, and by a trace-by-trace inspection, it is possible to conclude that the activity of the users during the monitored period is reasonably sustained, and compatible with typical traffic, meaning that the patterns are neither trivial (users effectively send requests) nor anomalous (users do not overload the destination with huge rates).

The collected streams have been partitioned into chunks of 2 min. In the forthcoming analysis, two perspectives are taken. In one scenario, the number of normal users is 10, each user has multiple 2-min chunks, and, per each trial, I choose randomly one trace per user. In the other scenario, 2-min chunks belonging to the *same* user have been treated as if they were coming from *distinct* users. In this way, the number of normal users is multiplied (fictitiously). This is clearly an approximation, since, e.g., fictitious users stemming from the same user might feature an additional-and-spurious degree of dependence. On the other hand, this (possible) increase of dependence goes in the direction of (possibly) increasing the fraction of normal users mistakenly marked as bots. Therefore, the simulations performed in the "multiplied"

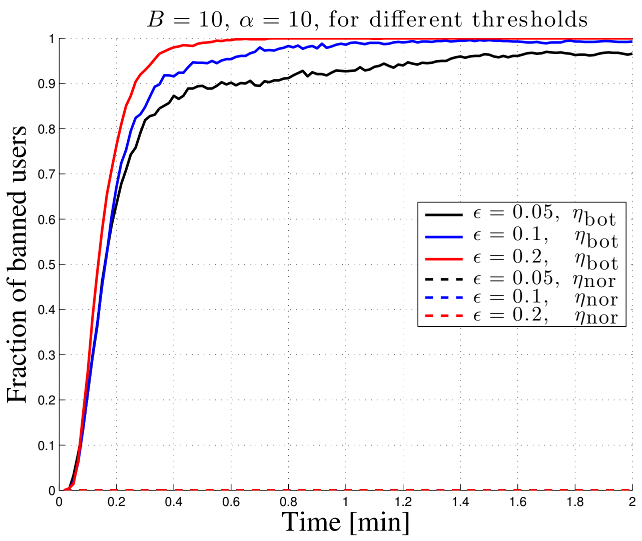


Figure 4.8: Fraction of banned users as a function of time, for different values of the threshold parameter ϵ. The monitored network is composed of 10 normal users, and contains $B = 10$ bots. Solid curves refer to correctly banned bots, whereas dashed curves refer to erroneously banned normal users. The depicted curves are computed over 100 Monte Carlo trials. Per each trial, 2-min chunks of each user are randomly selected among the available chunks.

scenario are expected to provide a conservative performance assessment.

The DDoS attack has been generated so as to fall into the class described in Sec. 4.4. Given the dictionary of messages obtained from the *whole* activity recorded in the laboratory, it is assumed that, at epoch t, only the first $\lfloor e_0 + \alpha t \rfloor$ messages of such a dictionary are available to the botnet, giving rise to the emulation dictionary $\mathscr{E}(t)$, for fixed parameters e_0 (size of the dictionary at $t = 0$) and α. Independently at each bot, a Poisson time-scheduling is randomly generated, and, per each transmission epoch t, each bot picks messages at random from the currently available $\mathscr{E}(t)$. In the forthcoming analysis, e_0 will be set to value 100, unless stated otherwise.

4.7.1 Experimental threshold setting

It is useful to recall that the algorithm is non-parametric, namely, it does not assume knowledge neither of the transmission rates, nor of the parameters of the botnet emulation dictionary (e_0 and α). In contrast, the size of the network is obviously known. The only input parameter is the factor ϵ appearing into (4.26). In Fig. 4.8 I consider a network comprising 10 normal users plus 10 bots. The botnet EDR is $\alpha = 10$. I remark that such a value is compatible with some of the empirical values $\hat{\alpha}$ estimated over the normal users' traces. The BotBuster algorithm has been implemented for three values of the threshold parameter $\epsilon \in (0, 1)$, namely, $0.05, 0.1$, and 0.2, and the estimates obtained for the fraction of banned users have been averaged over 100 Monte Carlo trials. The observation window lasts 2 min, and the simulation points refer to the output of the algorithm taken each 1 s. It is possible to see that the dashed curves are in practice invisible, revealing that the estimated η_{nor} is almost zero for all the considered values of ϵ. This behavior should be contrasted to what will be observed later on in Fig. 4.12, where, in the absence of a botnet, the BIC was occasionally violated. However, as discussed at the end of Sec. 4.6, the spurious-and-small estimated clusters containing normal users can be efficiently ruled out by the fact that the algorithm selects, as a final estimate, only the cluster with maximum size, which is expected to contain only bots.

With regard to the fraction of correctly identified bots, we see that η_{bot} increases as ϵ increases from 0.05 to 0.2. In fact, increasing ϵ makes it easier staying *below* the threshold, which facilitates the inclusion of a node in the estimated botnet.

The analysis summarized in Fig. 3.3 reveals that the choice of the threshold is not critical, and the algorithm offers excellent performance for a relatively large range of ϵ. Indeed, recall that $\epsilon \in (0, 1)$, and that ϵ must be "small", so that $\epsilon = 0.05$ up to 0.2 can be definitely considered a "large", flexible range.

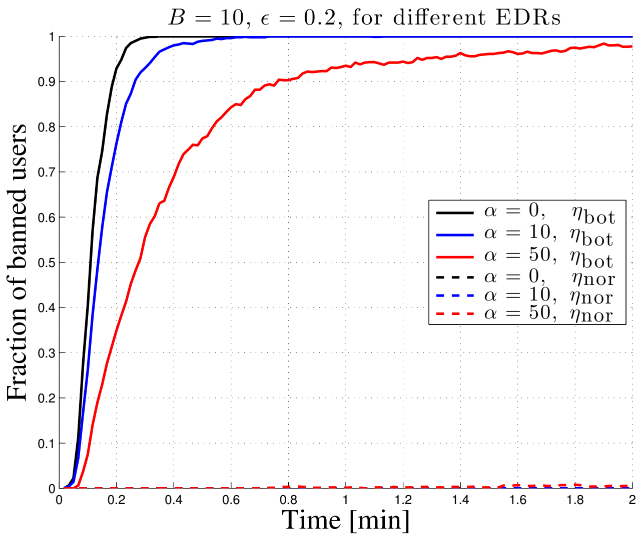


Figure 4.9: Fraction of banned users as a function of time, for different values of the EDR α. The monitored network is composed of 10 normal users, and contains $B = 10$ bots. Solid curves refer to correctly banned bots, whereas dashed curves refer to erroneously banned normal users. The depicted curves are computed over 100 Monte Carlo trials. Per each trial, 2-min chunks of each user are randomly selected among the available chunks.

4.7.2 Role of Emulation Dictionary Rate

In Fig. 4.9, the different curves refer to three EDR values (which, I recall, is *not known* to the algorithm). The threshold parameter ϵ was set to 0.2. Let us start by examining the behavior of η_{nor}. We see that, irrespectively of the EDR value, η_{nor} stays approximately constant at 0, which matches our previous evidences and observations.

Let us switch to the analysis of η_{bot}. The lowermost curve corresponds to the highest EDR value considered in the figure, namely, to $\alpha = 50$. Compared to what it has been observed in the network traces collected in our testbed environment, such an EDR is a kind of relatively high value. It is possible to note that the average percentage of correctly identified bots is relatively large

($> 80\%$), even at the beginning of the monitoring activity. Then, the estimated η_{bot} increases, approaching unity as time elapses, *in perfect accordance with the theoretical results of Theorem* 2.

Next I examine the influence of the EDR on the algorithm performance. We see that the curves corresponding to η_{bot} move upward as α decreases. This sounds perfectly reasonable, since α quantifies the learning ability (i.e., the power) of the botnet. On the other hand, for each value of α, the performance must eventually reach the limiting value of unity after a sufficiently long time. In particular, the uppermost curve corresponds to the degenerate case $\alpha = 0$, namely, to the classical and well-documented case where the botnet uses repeatedly the same patterns. As such, the case $\alpha = 0$ could be addressed by other (simpler) tools, since a normal user will seldom feature such a small innovation rate. In summary, the conducted analysis emphasizes that the performance decreases with the botnet learning ability α.

4.7.3 Network size scaling

In order to ascertain the feasibility of the proposed methodology, it is crucial to capture how the performance scales as the network size is increased.[9] In Fig. 4.10, the fraction of correctly banned users for networks of increasing size, featuring a balanced proportion of bots and normal users has been displayed. The rightmost group of curves corresponds to a size of the initial dictionary chosen as done in the previous numerical experiments ($e_0 = 100$), while the leftmost group corresponds to the limiting case of an almost empty initial dictionary ($e_0 = 1$). As to the rightmost family of curves, the performance increases slowly when the size is varied from 20 to 100, while it stays almost constant when the size is varied from 100 to 200. The opposite behavior is observed for the leftmost group. Joint inspection of the two cases implies the following observations: *i*) no monotonic behavior emerges with respect to N; *ii*) the variation in performance when N is varied

[9]I shall consider the aforementioned scenario where normal users are fictitiously multiplied by treating chunks of the same user as distinct users.

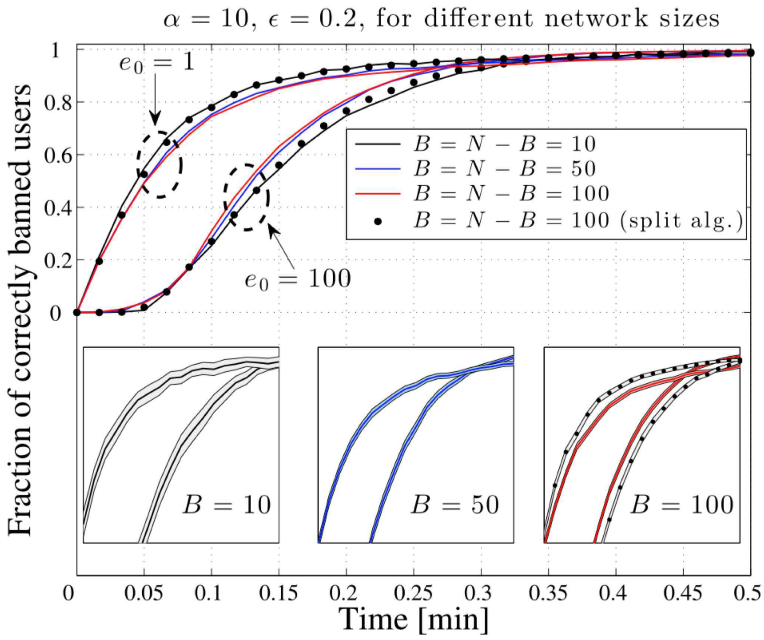


Figure 4.10: Fraction of correctly banned users as a function of time, for different network sizes N ("multiplied" scenario — see main text). The two groups of curves refer to different sizes of the initial dictionary. Markers refer to a parallel implementation of the BotBuster algorithm over 10 disjoint and equal-size subsets of the entire network, for the case $B = N - B = 100$. The depicted curves are computed over 100 Monte Carlo trials. In the insets, a close-up of the curves in the range $\eta_{\rm bot} \geq 0.5$ is displayed, reporting also the 95% confidence intervals (shaded areas).

up to an order of magnitude is modest. The latter clues are crucial to support feasibility of the proposed algorithm.

In addition, in Fig. 4.10 I have reported the performance corresponding to a "partitioned" application of BotBuster (markers), where the largest network ($N = 200$) is examined by splitting the nodes into 10 sub-groups, and BotBuster is then launched in parallel over each sub-group. As it should be expected, such a parallel application delivers approximately the performance corresponding to an individual sub-group.[10] Otherwise stated, a parallel implementation over subnets, say, of 1/10 of the total network size, keeps at least the promises of the performance corresponding to a network 10 times smaller, with additional computational and/or time savings. This is a further important element that supports feasibility of the proposed method. Moreover, it should be noticed that the implementation of BotBuster over the whole network (i.e., without splitting) offers the additional feature of performing joint checks among the sub-groups, which would be skipped by a parallel implementation.

To complete the analysis, I report that η_{nor} (not displayed to enhance readability) is practically zero in all the scenarios of Fig. 4.10, irrespectively of the network size. Finally, it is useful to stress that simulations were carried for networks up to 200 nodes, and the algorithm was able to guarantee the real-time requirement, yet on a standard laptop, without memory/code optimization, with such issues being beyond the scope of the work.

4.7.4 Multiple bots / Spoofed addresses

The setting considered in this work encompasses naturally the relevant scenario of spoofed source IP addresses, which is becoming rather common in DDoS attacks. In such scenario, each bot can change its source IP address by (randomly) choosing from a collection of spoofed addresses. In the randomized DDoS attack considered in this work, the bot traffic streams are constructed by picking subsequent messages independently from an emulation dictionary that is shared among all the bots. Accordingly, a botnet of B nodes employing a set of A randomly spoofed addresses (with $A > B$), is equivalent to a botnet of A nodes performing the attack. Since the goal of the network analyst is banning the machines that launch the attack (not associating a physical machine to its IP address), it is possible to conclude that the performed analysis applies directly to the case of spoofed IP addresses, pro-

[10]A rough performance prediction can be easily made assuming an approximate equipartition of bots among the sub-groups.

vided that the number of bots is replaced by the number of IP addresses globally employed by the botnet. For the sake of brevity, such "effective" number will be still denoted by B.

There are at least two meaningful regimes to examine the case of increasing number of bots and/or spoofed addresses: *i*) the regime where B increases, while the individual bots' transmission rate, λ_{bot}, is constant, implying a growth of the total DDoS attacking rate $B\lambda_{\text{bot}}$; *ii*) the regime where B increases while keeping the attacking rate constant. As regards the former regime, unlike the analysis of the previous section, varying B corresponds to varying the relative proportion of bots and normal users. This notwithstanding, the evidences arising from the simulation pertaining to such scenario are very similar to those observed in Fig. 4.10, and are accordingly not reported. In summary, in this regime the dependence of η_{bot} upon B is not obvious (no monotonic behavior emerges with respect to B, which is partly explained by noting that increasing B should augment the botnet "visibility", but also the number of possible algorithm mistakes), and the performance is little sensitive to variations of B.

Let us now move on to examine the second regime of operation. It is expected that, for a given total attacking rate, the botnet has more convenience in distributing its requests over more bots and/or spoofed addresses, which corresponds to increasing B while proportionally reducing λ_{bot}. Such scenario is illustrated in Fig. 4.11. Since in this case increasing B implies a reduction of the transmission rate, it is expected that the time to reach convergence increases, and that the mutual dependencies are disseminated over a larger number of bots, resulting into a reduced botnet identifiability. Such behavior is reflected by the shifting of the curves in Fig. 4.11.

From a practical perspective, there are two important evidences arising from Fig. 4.11. First, increasing B by even one order of magnitude does not jeopardize botnet identifiability ($\eta_{\text{bot}} \approx 0.99$ before half a minute). Such a mild dependence is supported by the following observation: convergence of the algorithm should roughly depend upon the average number of transmissions $\lambda_{\text{bot}}\, t$.

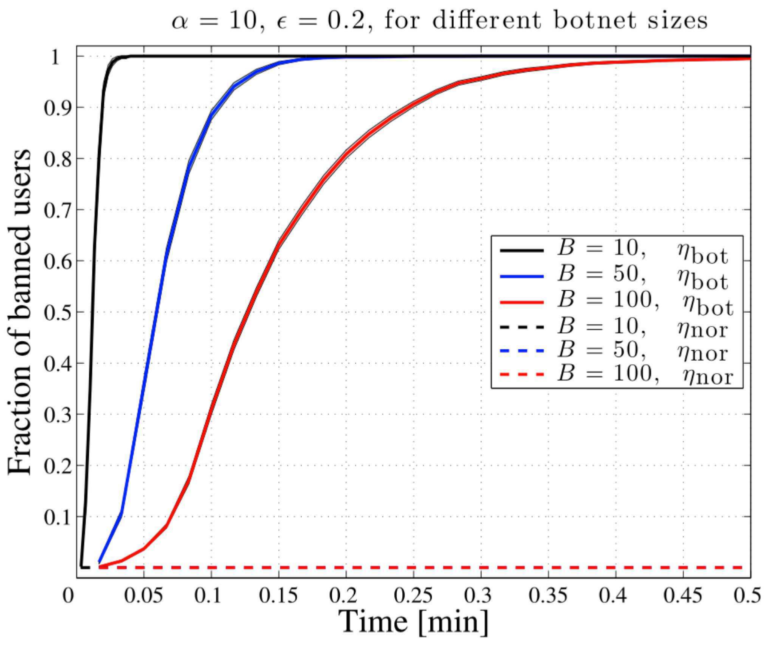


Figure 4.11: Fraction of banned users as a function of time, for different botnet sizes B, in the constant attack-rate regime. The monitored network contains 100 normal users ("multiplied" scenario — see main text). Solid curves refer to correctly banned bots, whereas dashed curves refer to erroneously banned normal users. The depicted curves are computed over 100 Monte Carlo trials. Shaded areas correspond to 95% confidence intervals.

Accordingly, the curves are expected to undergo a delay shift that scales approximately as $1/\lambda_{\text{bot}}$, i.e., almost linearly with B because the product $B\lambda_{\text{bot}}$ is constant.

More remarkably, it is possible to see that a relatively strong botnet power is required in order to impair substantially the algorithm performance. Let us now see why. The rate of requests of the "normal" part of the network scales as $(N - B)\lambda_{\text{nor}}$. Assume that the request rate necessary to saturate the attacked website resources must be κ times larger than the overall rate of normal users. This implies the condition $\lambda_{\text{bot}}/\lambda_{\text{nor}} = \kappa(N - B)/B$. Now, the rightmost curve (the more advantageous for the botnet) cor-

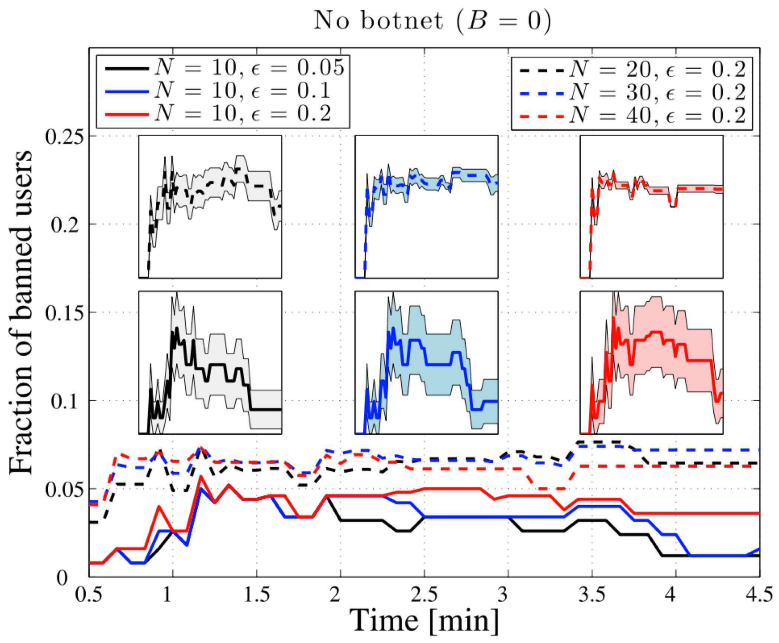


Figure 4.12: Fraction of banned users as a function of time, for different values of the threshold parameter ϵ, and for different network sizes. The monitored network contains no bots. The depicted curves are computed over 100 Monte Carlo trials. Per each trial, 5-min chunks of each user are randomly selected among the available chunks. In the insets, the pertinent curves are displayed along with the 95% confidence intervals (shaded areas).

responds to the case that $\kappa = 1$ and $\lambda_{\text{bot}} = \lambda_{\text{nor}}$, namely, to the optimistic (from the botnet perspective) assumptions that a relatively low attack rate is sufficient to impair the target site ($\kappa = 1$), and that the available number of bots and/or spoofed addresses equals the number of normal users. In contrast, the leftmost curve corresponds to the (perhaps more realistic) case that the botnet is one order of magnitude smaller than the "normal" part of the network. In order to increase substantially the time needed to reach a satisfying accuracy, the botnet should use a number of bots that exceeds the number of normal users by at least one order of magnitude.

4.7.5 No botnet setting

One might question that, in an IDS pipeline, a botnet identification algorithm is usually triggered by (or coupled with) a detection stage hunting for anomalous rises in the request rate. Thus, in the absence of such anomaly, identifying a dependence among small group of users does not imply (nor legitimate) a banning action. Nevertheless, it is useful to verify that the algorithm works properly even with $B = 0$. Such analysis is also an indirect validation of the BIC, since it focuses specifically on the dependencies among normal users. In Fig. 4.12, I display the fraction of erroneously banned users, $\eta_{\rm nor}$, for several scenarios, namely: a network with 10 normal users, for three values of the threshold parameter ϵ; networks of increasing size, examined for the same value of ϵ. Unlike the previous analysis, here I focus on a longer observation window, in order to track possible (undesirable) increasing trends of $\eta_{\rm nor}$. Let us start by examining the dependence upon ϵ. Were the BIC exactly verified for any subset of normal users, and for any time epoch, the fraction of banned users should be always zero. As already discussed, in practice the BIC is expected to be approximately verified. This notwithstanding, in Fig. 4.12 it is possible to see that the percentage of erroneously banned users is very small for all the thresholds in the considered range, never exceeding 6%. Notably, such behavior suggests that a BIC violation is unlikely to occur, and that, in any case, it involves small groups of users. Thus, as happens for the case $B > 0$, the dependence of performance upon the threshold is not critical. Let's now examine the variation in performance as the number of users increases. It is possible to see that the performance approximately increases (see curves corresponding to $N = 10, 20, 30$), but then approximately decreases ($N = 40$). This behavior suggests that, as N grows, the cardinality of spurious clusters is not significantly influenced by the latter growth, which in turn implies a reduction in the relative *fraction* of erroneously banned users. In summary, it is possible to conclude that, as already shown for the case $B > 0$, the scaling of performance with the network size does not emerge as an issue.

Chapter 5

Randomized DDoS attacks in a multi-clustered environment

5.1 The multi-clustered scenario

The key idea in this section is to extend the results obtained in Sec. 4 to a more challenging environment. In this new setting, in fact, the botnet is supposed to be spread across the network in many non-overlapping clusters, each of which has access to an emulation dictionary $\mathscr{E}_c(t)$, for $c = 1, 2, \ldots, C$ clusters as depicted in Fig. 5.1. The new challenge is to reveal the presence of multiple botnet clusters hidden in the network via a new algorithm derived from BotBuster algorithm described in the previous section.

5.2 Network indicators for multi-clustered DDoS

It is worth to re-arrange some network indicators introduced in the previous section, in order to face the novel multi-clustered DDoS attack. The *empirical transmission rate* and the *empirical MIR* are again defined in accordance to (4.2) and (4.3) respectively.

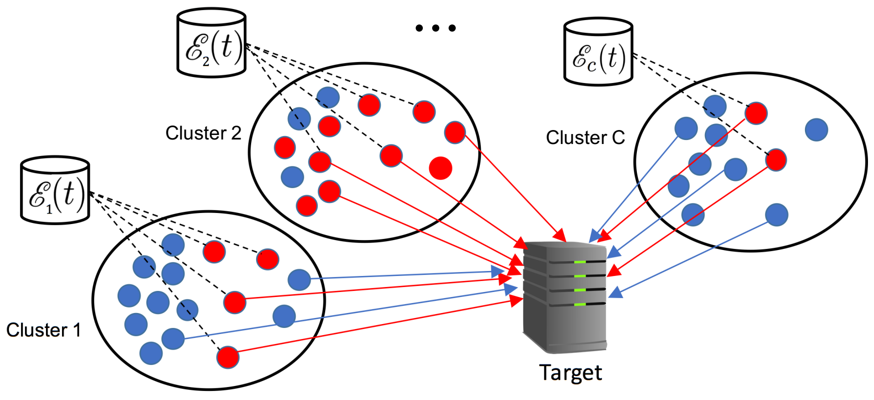


Figure 5.1: The botnet if made of C non-overlapping clusters. Every cluster has the possibility to access its own emulation dictionary $\mathscr{E}_c(t)$, for $c = 1, 2, \ldots, C$.

As previously specified, the model for *multi-clustered* DDoS constitutes a generalization of the DDoS class originally proposed in [39]. The key assumption is that the botnet is made of C non-overlapping clusters, each of which has access to an emulation dictionary (at time t) denoted by $\mathscr{E}_c(t)$, for $c = 1, 2, \ldots, C$. A bot of the c-th cluster performs normal traffic emulation by picking admissible messages from $\mathscr{E}_c(t)$. In order to guarantee a non-suspicious innovation rate, the dictionary is learned in a *continuous* fashion, namely, its cardinality increases with t. To quantify the richness of the emulation dictionary, it is possible to introduce the Emulation Dictionary Rate (EDR) if the c-th cluster:

$$\alpha_c \triangleq \lim_{t\to\infty} \frac{|\mathscr{E}_c(t)|}{t} \tag{5.1}$$

When a bot of the c-th cluster transmits, it picks (uniformly at random) a message from the available emulation dictionary $\mathscr{E}_c(t)$. As a result of the transmission activity, to any subnet $\mathcal{B}$ of the botnet it is possible to associate a certain *empirical* dictionary, $\mathscr{D}_{\mathcal{B}}(t)$. At time $t + s$, such an empirical dictionary is possibly increased by embodying the *distinct* messages (which were not initially contained in $\mathscr{D}_{\mathcal{B}}(t)$) picked during interval s by the bots in $\mathcal{B}$. As in the case of classical botnet environment where a formalization of MIR is offered, it is useful to formalize the MIR concept in a clusterized setting, by rephrasing the Theorem 1 of Sec. 4:

Theorem 3 **(MIR of a multi-clustered botnet).** *Let $\mathcal{B}_{\text{tot}}$ be a multi-clustered botnet, and let the transmission policies be either synchronous with constant transmission rate, or independent Poisson processes, with rates λ_u, for $u \in \mathcal{B}_{\text{tot}}$. Let $\mathcal{B} = \bigcup_{c=1}^{C} \mathcal{B}_c$, where $\mathcal{B}_c$ is a subnet of the c-th botnet cluster, and let α_c the EDR of the c-th cluster. If $\mathcal{B}_c \neq \emptyset$, the (limiting) MIR of $\mathcal{B}_c$ is:*

$$\rho_{\mathcal{B}_c} = \frac{\alpha_c \, \lambda_{\mathcal{B}_c}}{\alpha_c + \lambda_{\mathcal{B}_c}} \tag{5.2}$$

where $\lambda_{\mathcal{B}_c} = \sum_{u \in \mathcal{B}_c} \lambda_u$ is the aggregate transmission rate of $\mathcal{B}_c$. Moreover, the overall MIR of $\mathcal{B}$ fulfills the inequality:

$$\rho_{\mathcal{B}} \leq \sum_{c:\mathcal{B}_c \neq \emptyset} \frac{\alpha_c \, \lambda_{\mathcal{B}_c}}{\alpha_c + \lambda_{\mathcal{B}_c}} \tag{5.3}$$

which is satisfied with equality when the emulation dictionaries of the different clusters are mutually disjoint. ■

As regards the individual-cluster MIR in (5.2), the result comes directly from Theorem 1 in as described in Sec. 4. As regards the overall MIR in (5.3), the result comes from the fact that the MIR is sub-additive, while the equality follows because disjointness of the emulation dictionaries implies disjointness of the corresponding empirical dictionaries and, hence, additivity of the corresponding MIRs.

5.3 Malicious clusters identification algorithm

The possibility of a successful botnet identification relies on the fact that bots and normal users are expected to behave quite differently as regards their degree of innovation. In fact, the members of a botnet cluster produce their transmission activity by picking messages from one and the same emulation dictionary. The implied commonalities between two members of the same botnet cluster are expected to emerge in terms of a MIR that is lower than the MIR that would be obtained, e.g., if the two users were normal. This is because the mutual independence of the activities of two normal users, or of a normal user and a bot, implies typically a low degree of correlation (some partial overlap could arise due to, e.g., common interests, popular web-pages, peculiar website structure), which is reflected in a small intersection between the corresponding (individual) empirical dictionaries. Such heuristic argument has been made precise in Sec. 4. Specifically, given two disjoint subnets, $\mathcal{S}_1$ and $\mathcal{S}_2$, two MIRs are introduced, namely, the sum of MIRs: $\hat{\rho}_{\text{sum}}(\mathcal{S}_1, \mathcal{S}_2) \triangleq \hat{\rho}_{\mathcal{S}_1} + \hat{\rho}_{\mathcal{S}_2}$, and the MIR of a reference botnet: $\hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2) \triangleq \frac{\hat{\alpha}'(\mathcal{S}_1,\mathcal{S}_2)(\hat{\lambda}_{\mathcal{S}_1}+\hat{\lambda}_{\mathcal{S}_2})}{\hat{\alpha}'(\mathcal{S}_1,\mathcal{S}_2)+\hat{\lambda}_{\mathcal{S}_1}+\hat{\lambda}_{\mathcal{S}_2}}$, with explicit dependence upon t being suppressed for ease of notation. The value $\hat{\alpha}'(\mathcal{S}_1, \mathcal{S}_2)$ in the latter formula is a reference EDR estimated from the data. Then, for $\epsilon \in (0, 1)$, an intermediate threshold is defined as: $\gamma(\mathcal{S}_1, \mathcal{S}_2) = \hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2) + \epsilon\,[\hat{\rho}_{\text{sum}}(\mathcal{S}_1, \mathcal{S}_2) - \hat{\rho}_{\text{bot}}(\mathcal{S}_1, \mathcal{S}_2)]$. The heuristic reasoning about identifiability translates into the following conditions. When the two subnets belong to the *same* botnet cluster (below referred to as "joint case"), the empirical MIR, $\hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2}$, converges toward $\hat{\rho}_{\text{bot}}$ as time elapses, as predicted by Theorem 1. Next, consider the case that one subnet contains normal users and/or bots belonging to clusters not contained in the other subnet. In such case (below referred to as "nearly-disjoint case") it is realistic to assume that the degree of dependence between the two subnets is lower than the degree of dependence observed when both subnets belong to the *same* botnet cluster. The above arguments lead to:

$$\text{Joint case} \;\Rightarrow\; \hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2} < \gamma(\mathcal{S}_1, \mathcal{S}_1), \tag{5.4}$$

$$\text{Nearly-Disjoint case} \;\Rightarrow\; \hat{\rho}_{\mathcal{S}_1 \cup \mathcal{S}_2} \geq \gamma(\mathcal{S}_1, \mathcal{S}_1). \tag{5.5}$$

```
Algorithm: $\hat{\mathcal{B}}$=BotClusterBuster(traffic patterns, $\epsilon$, $\kappa$, $\xi$)
$\mathcal{N} = \{1, 2, \ldots, N\}$; $\hat{\mathcal{B}} = \emptyset$
for $i \in \mathcal{N}$ do
    $\hat{\mathcal{B}}_i = \{i\}$
    for $j \in \mathcal{N} \setminus \{i\}$ do
        if $\hat{\rho}(\hat{\mathcal{B}}_i \cup \{j\}) < \gamma(\hat{\mathcal{B}}_i, \{j\})$ then $\hat{\mathcal{B}}_i = \hat{\mathcal{B}}_i \bigcup \{j\}$
    end
    if $|\hat{\mathcal{B}}_i| = 1$ then $\hat{\mathcal{B}}_i = \emptyset$
    if $\hat{\lambda}_{\hat{\mathcal{B}}_i} \leq \frac{\kappa}{1+\kappa} \xi \, \hat{\lambda}_{\mathcal{N}}$ then $\hat{\mathcal{B}}_i = \emptyset$      (cluster expurgation)
end
$\hat{\mathcal{B}} = \bigcup_{i=1}^{N} \hat{\mathcal{B}}_i$
```

Actually, when (5.5) is exactly verified (the verification of (5.4) being guaranteed, for t large enough, by Theorem 3), we shall say that the Botnet Identification Condition (BIC) is fulfilled.

However, there is an issue that forbids successful applicability of the BotBuster algorithm (see Sec. 4) to the multi-clustered case addressed in this work. Such issue relates to the fact that (as experimental verification reveals) the BIC is *not* always verified in practice. As a result, during its flow, the algorithm occasionally produces, along with the (nearly-)right botnet, spurious groups of users that are not the right botnet. In the single-cluster case, such pathology is remediated by choosing, at the end of the procedure that scans all the nodes as pivots, the estimated botnet with the highest cardinality. Such choice is based on the observation that the cardinality of groups erroneously marked as botnet is typically much smaller than the cardinality of a real botnet. In the multi-clustered case, opting for the same maximum-cardinality rule is clearly detrimental, since it would select only the largest botnet cluster, which might be a largely insufficient measure of protection to face the DDoS attack. Therefore, different strategies to face a multi-clustered DDoS attack are necessary, and, some details about them are offered in the following.

5.3.1 Algorithm examination

Let's start to examine the main routing of BotClusterBuster, whose pseudo-code is reported in the top of previous page. For the sake of simplicity, dependence upon time is suppressed. Initially, the algorithm selects the first user as pivot (this operation will be repeated for all N nodes). User 1 is initially declared as a bot ($\hat{\mathcal{B}}_1 = \{1\}$). Then, by means of (5.4) and (5.5), it is declared whether users 1 and 2 form a botnet. If so, $\hat{\mathcal{B}}_1 = \{1, 2\}$, otherwise $\hat{\mathcal{B}}_1 = \{1\}$. Then, it is declared whether the currently estimated botnet $\hat{\mathcal{B}}_1$ forms a botnet with user 3, and so on. At the end of this loop, a candidate botnet cluster $\hat{\mathcal{B}}_1$ is obtained (if the candidate cluster has cardinality equal to one, it is automatically discarded). After iterating such inner loop over the entire set of pivots, the algorithm ends up with a sequence of candidate clusters, namely, $\hat{\mathcal{B}}_1, \hat{\mathcal{B}}_2, \ldots, \hat{\mathcal{B}}_N$. It is worth noting that, unlike the BotBuster algorithm, *all* the candidate botnet clusters produced in the intermediate algorithm steps should be retained, in order to take into account the possible presence of *multiple* botnet clusters. The situation is pictorially illustrated in Fig. 5.2, where are displayed the candidate botnet clusters estimated by the algorithm at a certain time, with reference to a network composed by 100 normal users and 100 bots, with 4 true botnet clusters, with sizes $10, 20, 30, 40$. The i-th "row" of the image represents the output of the algorithm when user i is chosen as a pivot. A white pixel means "estimated bot presence", a black pixel means "estimated bot absence". Accordingly, if the (i, j)-th pixel is white, the algorithm is estimating that user j is a bot when user i is chosen as pivot. From Fig. 5.2, it is possible to appreciate the emergence of 4 clusters, corresponding to the true botnet clusters (bots are ordered so as to appear well-clusterized in the image, a choice made only for clarity, since the algorithm is clearly invariant to permutations). On the other hand, we also see that a couple of small spurious clusters is wrongly identified by the algorithm. Now, were the BIC verified,

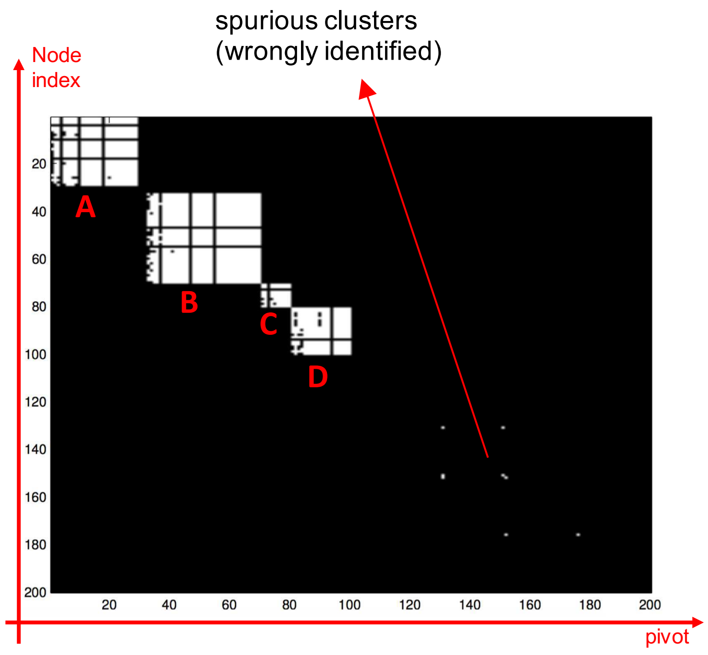


Figure 5.2: Algorithm applied to a network made of 100 bots and 100 normal users. Four significant clusters emerge (A,B,C,D), plus spurious micro-clusters.

all checks performed by the algorithm would give the right answer (with probability tending to 1 as $t \to \infty$), and the sequence of candidate clusters would contain exclusively (repetitions of) the true botnet clusters. Therefore, a sufficient criterion to produce the global botnet estimates would consist of applying the union operator: $\hat{\mathcal{B}} = \bigcup_{i=1}^{N} \hat{\mathcal{B}}_i$. The corresponding algorithm will be referred to as UnionBotBuster. The pseudo-code for UnionBotBuster can be retrieved from the pseudo-code of BotClusterBuster, by simply skipping the instruction referred to *cluster expurgation*. However, since in practice the BIC is only approximately verified, the union rule would favor inclusion of spurious clusters. Therefore, some refined criterion to select the best clusters is desirable.

Actually, the DDoS power can be measured in terms of the global botnet transmission rate, $\lambda_{\mathcal{B}}$. For a DDoS attack to be effective, one expects that, for $\kappa \geq 1$: $\lambda_{\mathcal{B}} = \kappa \lambda_{\mathcal{N} \setminus \mathcal{B}}$, with $\lambda_{\mathcal{N} \setminus \mathcal{B}}$ being the global transmission rate of normal users. The lower bound $\kappa = 1$ corresponds to the optimistic (at the botnet's side) assumption that a low DDoS rate is sufficient to impair the target site. In terms of the overall network transmission rate, one gets: $\lambda_{\mathcal{B}} = \frac{\kappa}{1+\kappa} \lambda_{\mathcal{N}}$. It is reasonable to assume that a true botnet cluster concurs to the attack with a significant fraction of the attacking rate, which leads to the following botnet membership condition: for $i = 1, 2, \dots, N$, and for $\xi \in (0,1)$, the candidate cluster i is retained if:

$$\hat{\lambda}_{\hat{\mathcal{B}}_i} > \frac{\kappa}{1+\kappa} \xi \, \hat{\lambda}_{\mathcal{N}} \triangleq \tau, \tag{5.6}$$

and is discarded otherwise. The final estimate is then produced by applying the union operator to the survived clusters, namely, $\hat{\mathcal{B}} = \bigcup_{i : \hat{\lambda}_{\hat{\mathcal{B}}_i} > \tau} \hat{\mathcal{B}}_i$. In summary, it is possible to derive three rules to identify the clusters:

- *Max Rule*: this rule selects only the maximum-size emerged cluster. In the example shown in Fig. 5.2, the algorithm would select cluster A.
- *Union Rule*: this rule select *all* clusters. In the example shown in Fig. 5.2, the algorithm would select clusters A, B, C, D, plus spurious micro-clusters.
- *Cluster Selection*: this rule retains the true clusters and *expurgates* the spurious ones, by exploiting the structural properties of the candidate clusters. In the example shown in Fig. 5.2, the algorithm would select only clusters A, B, C, D,.

Let us now introduce two performance indices: for a network $\mathcal{N} = \{1, 2, \dots, N\}$, containing a botnet $\mathcal{B}$ ($\hat{\mathcal{B}}(t)$ is the final botnet estimated at time t):

$$\eta_{\text{bot}}(t) = \frac{\mathbb{E}[|\hat{\mathcal{B}}(t) \cap \mathcal{B}|]}{|\mathcal{B}|}, \quad \eta_{\text{nor}}(t) = \frac{\mathbb{E}[|\hat{\mathcal{B}}(t) \cap (\mathcal{N} \setminus \mathcal{B})|]}{|\mathcal{N} \setminus \mathcal{B}|}, \tag{5.7}$$

which are the expected fraction of correctly identified bots and the expected fraction of normal users declared as bots, respectively. It is possible to say that an identification algorithm is consistent if:

$$\lim_{t\to\infty} \eta_{\mathrm{bot}}(t) = 1, \qquad \lim_{t\to\infty} \eta_{\mathrm{nor}}(t) = 0 \tag{5.8}$$

Theorem 4 **(Consistency of the algorithms BotClusterBuster and UnionBotBuster).** *Let $\mathcal{N} = \{1, 2, \ldots, N\}$ be a network containing a multi-clustered botnet $\mathcal{B}$, with asymptotically disjoint emulation dictionaries. The bots' transmission policies are either synchronous with constant transmission rate, or independent Poisson processes. The normal users' transmission policies are arbitrary. If the BIC in (5.5) holds, then the algorithm UnionBotBuster is consistent. Moreover, let $\lambda_{\min}$ be the smallest (limiting) transmission rate of a botnet cluster. If:*

$$\lambda_{\min} > \frac{\kappa}{1+\kappa}\,\xi\lambda_{\mathcal{N}}, \tag{5.9}$$

then the algorithm BotClusterBuster is consistent. ■

Consistency of UnionBotBuster holds because: *i*) by Theorem 3, as $t \to \infty$, a candidate cluster obtained starting with a *bot* pivot converges to the *true* botnet cluster containing the pivot; *ii*) since the BIC is assumed to be perfectly verified, a candidate cluster obtained starting with a normal-user pivot converges to the empty set. As regards BotClusterBuster, convergence of $\eta_{\mathrm{nor}}(t)$ to zero is still implied by *ii*). On the other hand, successful inclusion of all candidate clusters coming from a bot pivot requires that the botnet-membership condition in (5.6) is verified for all clusters, at least for t large enough. This amounts to assume that, for the least favorable case (i.e., the cluster with the lowest transmission activity) the threshold is crossed. Since such condition is required asymptotically, the empirical values of the transmission rates in (5.6), are replaced, in (5.9), by their limiting counterparts. It is worth to remark that the theorem focuses on the case of (at least asymptotically) disjoint emulation dictionaries.

5.4 Experimental results

Basically, the experimental campaign has been carried out by considering the same laboratory setting exploited in Sec. 4. Accordingly, a popular e-commerce portal has been selected as target destination of the attack. A standard software for packet capturing and for performing a preliminary filtering has been deployed, so: *i*) only the application-layer traffic directed to the target destination is retained; *ii*) the survived packets are divided on the basis of source IP address, before being fed to the identification algorithm. As in the case of DDoS attack faced in Sec. 4, more than 20 minutes of traffic, from 10 users (students/researchers in laboratory based at University of Salerno) have been gathered. The obtained traffic streams have been then partitioned into 1-minute chunks. Then, each distinct chunk is treated as representative of a distinct user, for a total number of equivalent normal users ≈ 200. As regards the multi-clustered DDoS attack, an emulation dictionary is needed. At this aim, all the *distinct* messages present in the *overall* dataset have been first taken. Then, such ensemble has been split into C disjoint sets, with C being the number of clusters. Finally, at epoch t, the c-th emulation dictionary $\mathscr{E}_c(t)$ is constituted by the first $\lfloor e_0 + \alpha_c t \rfloor$ messages of such ensemble (e_0 is the initial dictionary size, set to 100 messages in the simulations). In a first set of experiments, the individual bot transmission rate has been chosen as twice the average rate of normal users, and the EDR has been chosen as compatible with the innovation rates estimated over the normal users' traces. Such choices are made to let the bots well concealed in the midst of legitimate users. An equal number (100) of normal users and bots, which is again a favorable choice for the attacker has been chosen. With reference to such setting, a single-cluster DDoS attack has been first launched. The botnet identification algorithms for this case are: the BotBuster algorithm introduced in Sec. 4 (which selects the maximum-cardinality cluster only), and the UnionBotBuster algorithm. Then, a *multi-clustered* DDoS, where the 100 bots have been split over $C = 4$ clusters, with sizes $10, 20, 30, 40$ (cluster

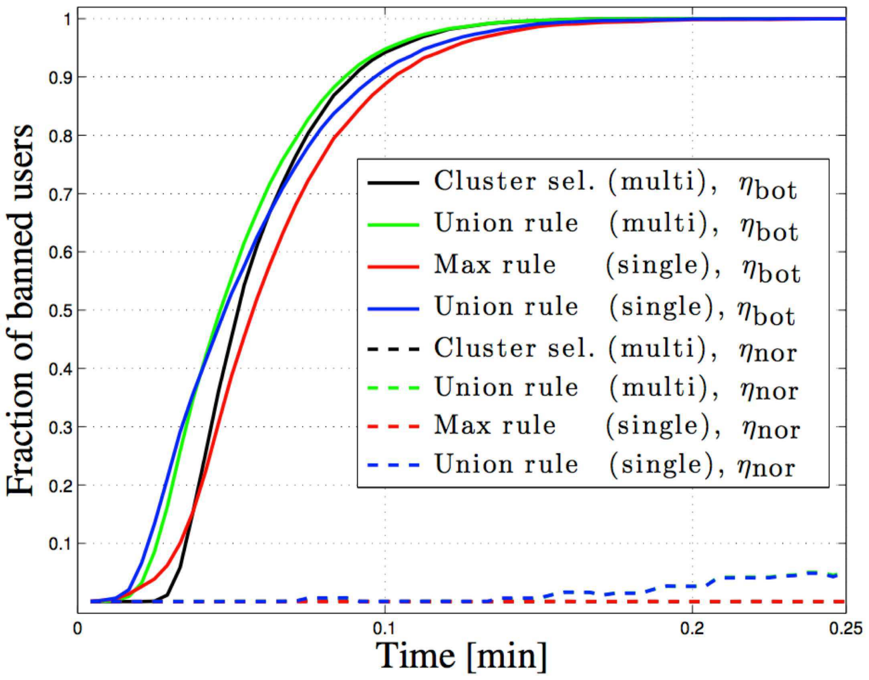


Figure 5.3: Expected fraction of banned users (estimated over 100 Monte Carlo runs) as a function of time, for different types of attack and identification algorithm, as detailed in the legend. The network is made of 100 normal users and 100 bots.

ordering is immaterial, since, in our simulations, the bots are randomly spread over the network) has been launched. The EDR of each cluster has been correspondingly reduced by a factor 4. For the multi-clustered scenario, the identification algorithms are BotClusterBuster and UnionBotBuster. As regards the threshold parameters of BotClusterBuster, the following values have been chosen: $\kappa = 1, \quad \xi = 1/10$. Both choices are not "too selective", in the sense that they tend to favor cluster inclusion, and, hence, they tend to increase η_{nor}. In fact, $\kappa = 1$ corresponds to a low reference attacking rate. Likewise, $\xi = 1/10$ corresponds to assume that the smallest cluster must contribute to the total attacking rate for at least one tenth of the total, which seems a rather conservative choice.

In Fig. 5.3, η_{bot} and η_{nor} as functions of time are displayed. Points of each curve correspond to the algorithms' output sampled each 0.25 seconds, with the average performance indices estimated over 100 Monte Carlo runs. As a general comment, it is possible to note that all algorithms do their job properly, since: *i*) the fraction of correctly banned users increases up to unity as time elapses, matching the theoretical results about algorithms' consistency; *ii*) the fraction of erroneously banned users is kept small. It is not zero, as a consequence of the forewarned imperfect verification of the BIC. Let us first compare the single-cluster case to the multi-clustered case. In particular, I focus on the comparison between the maximal-cluster rule (single) and BotClusterBuster (multi). One can see that η_{nor} is in practice zero for both cases (dashed red curve almost perfectly superimposed to dashed black curve). Switching to η_{bot}, it is possible to see that the performance in the multi-clustered case outperforms the performance corresponding to the single-cluster case, but for uninteresting small values of η_{bot}. Such behavior can be explained as follows. The power of a botnet cluster is ruled by the EDR, which, in the multi-clustered case, is smaller than the EDR of the single-cluster case. Now, provided that the cluster selection operates properly, the main factor ruling the algorithm performance is the EDR. Accordingly, the botnet identification is quicker in the multi-clustered case (lower EDR $\Leftrightarrow$ lower attack strength) than in the single-cluster case (higher EDR $\Leftrightarrow$ higher attack strength). Let's now move on examining the connections between the cluster-expurgation rule and the union rule. The fraction of banned users for the latter is always higher than that corresponding to the former. This behavior is expected, since the union rule does *not* attempt to select the best cluster(s), but merges all candidate bots. Clearly, the increase in the percentage of correctly banned users is paid in the coin of an undesired increase of the wrongly banned ones. In a second scenario (see Fig. 5.4), two different implementations of the multi-clustered DDoS are considered. In a first setting (case 1 in Fig. 5.4), the identification problem is made more challenging by *i*) reducing the individual bots transmission rate (by a

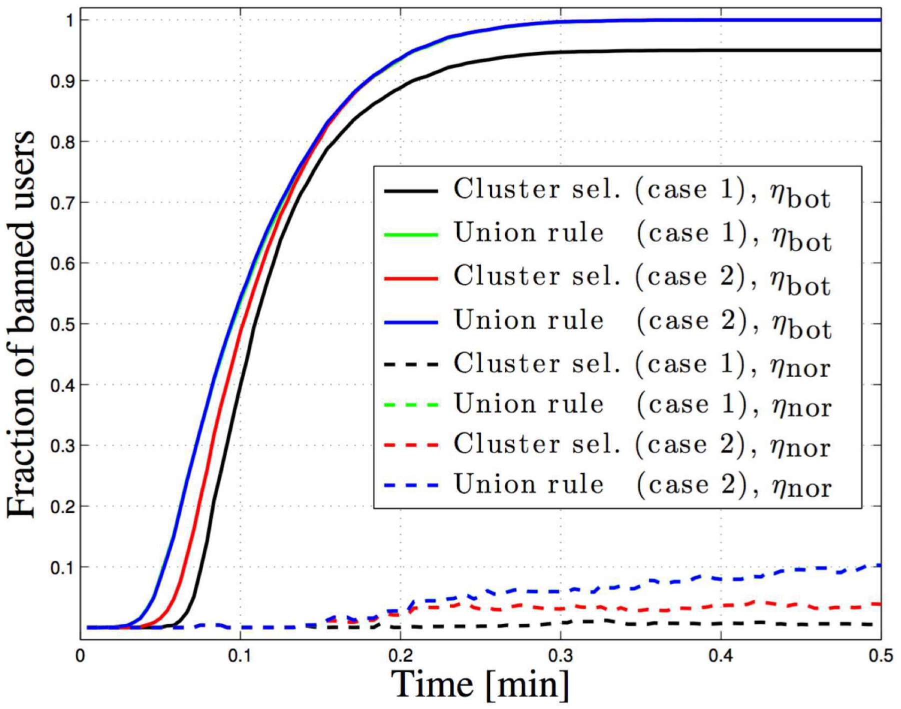


Figure 5.4: Expected fraction of banned users (estimated over 100 Monte Carlo runs) as a function of time, for different parameters of the multi-clustered DDoS and of the algorithms. The network is made of 100 normal users and 100 bots.

factor 2); *ii*) reducing the size of the smallest botnet cluster, with the cluster sizes being $5, 20, 30, 45$. In the second setting (case 2 in Fig. 5.4), the reduced transmission rate is still considered, but the cluster sizes are restored to the values $10, 20, 30, 40$, while the value of the cluster-selection parameter ($\xi = 1/20$) has been reduced. With this choice, we are in a sense over-estimating the maximum number of clusters, and, hence, we are favoring the inclusion of *spurious* clusters in the final estimate. The results of the experiments are displayed in Fig. 5.4. Let's start by focusing on the BotClusterBuster algorithm. As a general trend (applying both to case 1 and case 2) one can see that the new challenges (reduced transmission rate, reduced cluster size, reduced threshold parameter) correspond to an increase of η_{nor}. This is expected, since all the modifications w.r.t. the setting of Fig. 5.3 goes in the direction of favoring the inclusion of spurious clusters. In contrast, as regards η_{bot}, the situation changes in the two cases. In case 1, the low value of the smallest cluster size (equal to 5), leads to a violation of hypothesis (5.9), which forces BotClusterBuster to exclude the smallest cluster, and in turn explains why η_{bot} saturates to 0.95 as time elapses. Instead, in case 2, the variations of the sensible parameters are not sufficient to impair consistency. As regards the UnionBotBuster algorithm, considerations similar to the cases in Fig. 5.3 can be drawn. However, since now the observation window is increased, it is possible to notice an increasing trend in η_{nor}, which is probably due to the fact that, even if the spurious clusters are typically few and small, the union rule tend to enhance their contribution. As a result of the conducted analysis, some useful insights can be gained. While no superiority of one class of algorithms over the other class can be generally claimed, it is possible to state that algorithms performing a cluster selection could be more appropriate in typical DDoS scenarios, for the following reason. In a DDoS attack, the main goals of the network defender are: *i*) avoiding that the destination site crashes; *ii*) guaranteeing proper service to the legitimate users. Thus, it seems preferable to ban few normal users (lower η_{nor}), at the price of losing some bots (lower η_{bot}). Indeed, banning, e.g., 95% of the bots should not impair the destination site, and keeping η_{nor} as low as possible means minimizing the number of normal users with denied service.

Chapter 6

High Availability (HA): an effective prevention strategy

6.1 HA concepts in modern data networks

Today, modern data networks have to cope with unpredictable events such as natural disasters or increasingly sophisticated network attacks. From an infrastructural viewpoint, the most effective way to face these issues is to guarantee high availability requirements by designing redundancy strategies. With regard to network and telecommunication systems, high availability refers to an optimal redundant scheme satisfying the so-called "five nines" condition, where a maximum downtime of 5 minutes and 26 seconds per year is admitted. Accordingly, the present Section is focused on the formalization of the network availability problem (broadly intended, not only as a response to network attacks) along with the consequent design of an optimal prevention strategy. In particular, two statistical frameworks are proposed to model the high availability requirements: the Universal Generating Function (UGF) and the Stochastic Reward Networks (SRN). The former (see Sect. 2.4) is exploited to find the system performance distribution by considering the performance distribution of its elements and, hence, it turns to be useful in evaluating the whole system availability. Such a framework has been generalized by proposing a novel (to the best author's knowledge) Multi-dimensional extension referred to as MUGF (Multi-dimensional UGF). The latter (see Sect. 2.6) represents a state-space model exploited to characterize the probabilistic behavior of network elements also in terms of connections among them. UGF and SRN methods have been evaluated by considering a Software Defined Networking (SDN) setting and a Network Function Virtualization (NFV) environment, respectively.

6.2 HA and Software Defined Networking (SDN)

SDN has been introduced as a novel paradigm in traffic engineering aimed at dramatically simplifying network control and management enabling innovation through network programmability. Although classical traffic engineering techniques such as Asynchronous Transfer Mode (ATM) or Multi Protocol Label Switching (MPLS) were born with the objective of optimizing the performance of communication flows by dynamically analyzing, predicting and regulating the behaviour of transmitted data, the SDN technology requires an additional step for exploiting global network view, status and flow patterns/characteristics available for better traffic control and management [74]. In a traditional network element as the router, the data forwarding plane assumes responsibility for packet buffering, packet scheduling, header modification and forwarding; forwarding functionalities include logic and tables for choosing how to deal with incoming packets based on characteristics such as Media Access Control (MAC) address, Internet Protocol (IP) address, Virtual Local Area Network Identifier (VLAN ID) and so on. The control plane instead has its principal role in applying control and routing protocols to the

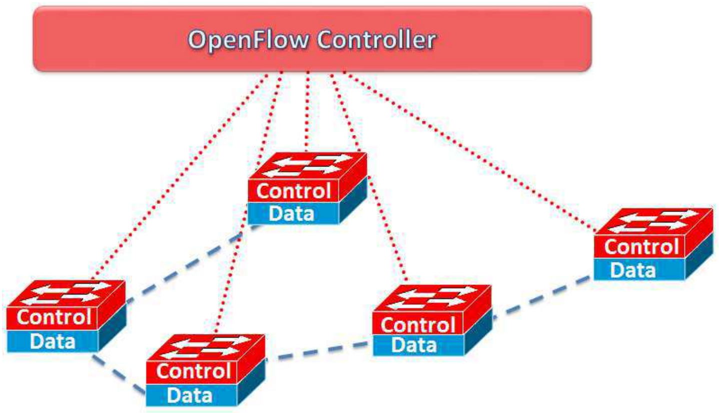


Figure 6.1: An overview of SDN/OpenFlow architecture with data (dashed) and control (dotted) flows.

data traffic. The management plane, at last, allows network engineers to configure and monitor the network devices. The key idea behind SDN paradigm resides in decoupling data and control plane so as the latter could be directly programmable. The most used protocol in an SDN environment is OpenFlow [75] enabling the communication between the control plane (represented by the controller element) and the data plane (represented by a set of SDN network elements called switches). The Fig. 6.1 depicts a schematic representation of an SDN architecture with its characteristic elements.

6.2.1 SDN Controller performance model

Taking into account all the advantages and benefits a virtualized SDN solution can bring, it is useful to focus on a scenario with a single controller element hosting and supervising some virtual instances as shown in Fig. 6.2, where every instance S_i represents the Master controller of the i-th service provider and is responsible of managing a set of OpenFlow-capable switches. I refer to S_i as the i-th Virtual Provider Instance (VPI).

The main benefit of this approach lies in providing a flexible and scalable network infrastructure that can be easily updated

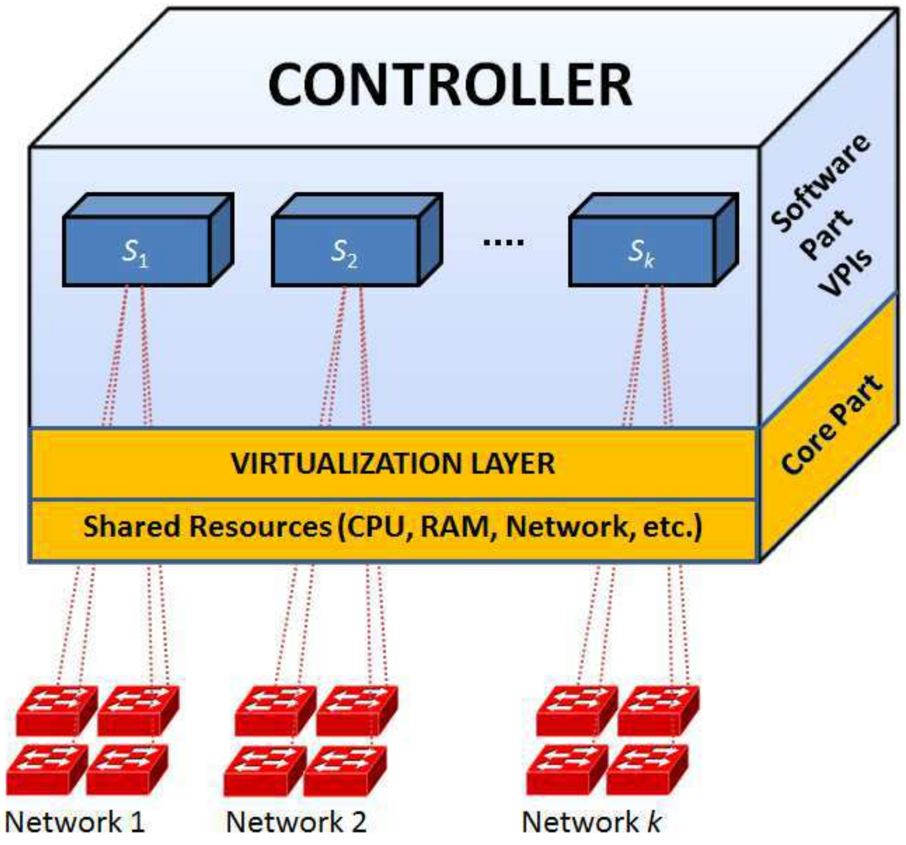


Figure 6.2: An SDN Controller architecture with k Virtual Provider Instances (VPIs).

by negotiating suitable Service Level Agreements (SLAs) with providers resulting in better quality of delivered services. The main drawback of this approach is that the physical controller becomes a critical point of failure and some redundancy mechanisms have to be introduced. Even if a suitable model for an SDN controller is mainly impacted by specific hardware and software implementations, it is reasonable to consider the controller element as composed by [76]:

- a *core part*, including all the hardware equipment (e.g. processors, memories, power supply, blades etc.) and basic software (e.g. operating system, hypervisor etc.);

- a *software part*, representing the virtual instances of controller, able to serve a certain number of OpenFlow connections towards a bundle of switches.

Since each VPI can manage a finite number of concurrent OpenFlow sessions within its data network, I select as a performance metric the number of the actual concurrent OpenFlow sessions that each virtual operator is able to handle, namely its controller serving capacity.

In the proposed model, each VPI in the controller node is supposed to manage n OpenFlow concurrent sessions. Furthermore, I suppose that every SDN controller can run k VPIs. A multi-state performance model for SDN controller can be proposed under the following hypotheses: each VPI and the core part of SDN controller behaviour may be described by a two-state model: an "up" state and a "down" state; failures and repairs are statistically independent; i-th VPI and core failures are statistically independent Homogeneous Poisson processes (HPPs), resulting in independent and exponential inter-failure arrivals and constant hazard rates λ_i and λ_c, respectively; VPI and core repair times are independent exponentially-distributed with rates μ_s and μ_c, respectively.

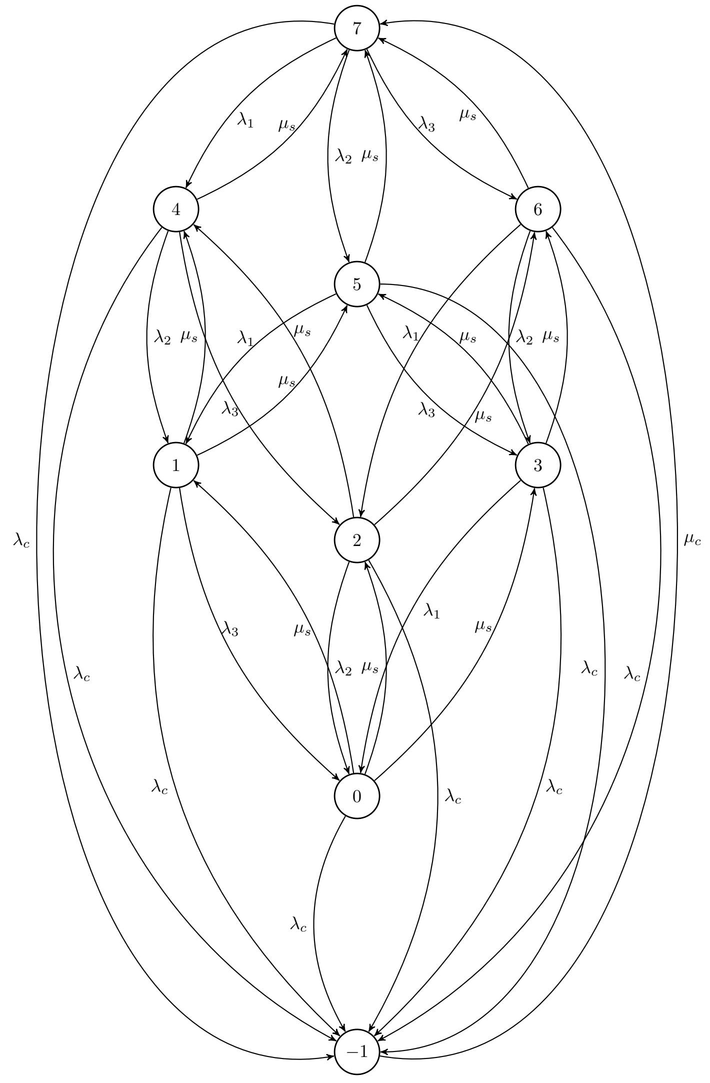


Figure 6.3: A multi-state model of the SDN controller with 3 Virtual Provider Instances (VPI). It relies on a representation via Continuous-Time Markov Chain (CTMC) where exponential inter-failure arrivals and constant hazard rates are indicated by λ_i and λ_c, respectively; VPI and core repair times are independent exponentially-distributed with rates μ_s and μ_c, respectively.

Table 6.1: Mapping between states and VPIs condition/performance triples.

State number	VPIs condition	Performance
7	(S_1, S_2, S_3)	(n, n, n)
6	$(S_1, S_2, \overline{S_3})$	$(n, n, 0)$
5	$(S_1, \overline{S_2}, S_3)$	$(n, 0, n)$
4	$(\overline{S_1}, S_2, S_3)$	$(0, n, n)$
3	$(S_1, \overline{S_2}, \overline{S_3})$	$(n, 0, 0)$
2	$(\overline{S_1}, S_2, \overline{S_3})$	$(0, n, 0)$
1	$(\overline{S_1}, \overline{S_2}, S_3)$	$(0, 0, n)$
0	$(\overline{S_1}, \overline{S_2}, \overline{S_3})$	$(0, 0, 0)$
-1 (core fault)	$(\overline{S_1}, \overline{S_2}, \overline{S_3})$	$(0, 0, 0)$

The resulting multi-state model of the SDN controller is a Continuous-Time Markov Chain (CTMC), where every state conveys the information on the working conditions (and thus the serving capacity) of the k VPIs. For sake of simplicity, a CTMC model is depicted in Figure 6.3, where $k = 3$ VPIs are supposed to work on the same controller and where:

- the total number of states are $2^k + 1$, where the correspondence between the state number and the triples containing all the permutations of the up-down conditions of the VPIs is illustrated in Table 6.1. The up and down condition of the i-th VPI is indicated by S_i and $\overline{S_i}$, respectively, and the serving capacity of the VPI is correspondingly n or 0;
- state -1 refers to the condition that core component is not working (*core fault*) and no VPI can be up, thus corresponds to the triple $(\overline{S_1}, \overline{S_2}, \overline{S_3})$ for $k = 3$;
- from the state -1, one transition towards a completely repaired controller is assumed.

The state probabilities $p_j(t)$, $j = -1, 0, 1, .., 2^k - 1$ can be obtained by solving a system of differential equations. For the CTMC in Figure 6.3, they are:

$$\begin{cases}
\dfrac{dp_7(t)}{dt} = \mu_s[p_4(t)+p_5(t)+p_6(t)]+\mu_c p_{-1}(t)+ \\
\qquad -(\lambda_1+\lambda_2+\lambda_3+\lambda_c)p_7(t) \\
\dfrac{dp_6(t)}{dt} = \mu_s[p_2(t)+p_3(t)]+\lambda_3 p_7(t)+ \\
\qquad -(\lambda_1+\lambda_2+\lambda_c+\mu_s)p_6(t) \\
\dfrac{dp_5(t)}{dt} = \mu_s[p_1(t)+p_3(t)]+\lambda_2 p_7(t)+ \\
\qquad -(\lambda_1+\lambda_3+\lambda_c+\mu_s)p_5(t) \\
\dfrac{dp_4(t)}{dt} = \mu_s[p_1(t)+p_2(t)]+\lambda_1 p_7(t)+ \\
\qquad -(\lambda_2+\lambda_3+\lambda_c+\mu_s)p_4(t) \\
\dfrac{dp_3(t)}{dt} = \mu_s p_0(t)+\lambda_3 p_5(t)+\lambda_2 p_6(t)+ \\
\qquad -(\lambda_1+\lambda_c+2\mu_s)p_3(t) \\
\dfrac{dp_2(t)}{dt} = \mu_s p_0(t)+\lambda_3 p_4(t)+\lambda_1 p_6(t)+ \\
\qquad -(\lambda_2+\lambda_c+2\mu_s)p_2(t) \\
\dfrac{dp_1(t)}{dt} = \mu_s p_0(t)+\lambda_2 p_4(t)+\lambda_1 p_5(t)+ \\
\qquad -(\lambda_3+\lambda_c+2\mu_s)p_3(t) \\
\dfrac{dp_0(t)}{dt} = -(\lambda_c+3\mu_s)p_0(t)+ \\
\qquad +\lambda_3 p_1(t)+\lambda_2 p_2(t)+\lambda_1 p_3(t) \\
\dfrac{dp_{-1}(t)}{dt} = -\mu_c p_{-1}(t)+\lambda_c \displaystyle\sum_{i=0}^{7} p_{(i)}(t)
\end{cases} \tag{6.1}$$

with initial conditions $p_7 = 1$, $p_i = 0$ with $i = -1, 0, ..., 6$, corresponding to an initial fully working system.

The performance level in every state of the CTMC is a k-tuple containing the serving capacity of the VPIs in that state, as reported in Table 6.1 for $k = 3$.

In order to take into account redundancy units of the controller that will be further introduced, it is possible to represent the performance levels of l-th controller by the set

$$\mathbf{g}^{(l)} = \{\mathbf{g}_{-1}^{(l)}, \mathbf{g}_{0}^{(l)}, ..., \mathbf{g}_{2^k-1}^{(l)}\}, \tag{6.2}$$

where $\mathbf{g}_j^{(l)} = \left(g_{1,j}^{(l)}, ..., g_{k,j}^{(l)}\right)$ is the performance k-tuple, with $g_{i,j}^{(l)}$ the serving capacity of the i-th VPI (i.e., offered to the i-th network) provided by the l-th parallel element in the state $j = -1, 0, ..., 2^k - 1$. Thus, the stochastic process $\mathbf{G}^{(l)}(t) \in \mathbf{g}^{(l)}$ represents the performance level of the l-th element at any instant $t \geq 0$, whose probability $p_j^{(l)}(t) = \Pr\{\mathbf{G}^{(l)}(t) = \mathbf{g}_j^{(l)}\}$ is computed by (6.1). Since we are interested on a performability evaluation in long runs, it is useful to compute the steady-state probabilities

$$p_j^{(l)} = \lim_{t \longrightarrow \infty} \Pr\{\mathbf{G}^{(l)}(t) = \mathbf{g}_j^{(l)}\}. \tag{6.3}$$

They can be derived by solving the equations (6.1), where all the derivatives are posed equal to 0 and $p_j^{(l)}(t)$ are replaced by $p_j^{(l)}$, together with the normalizing condition

$$\sum_{j=-1}^{2^k-1} p_j^{(l)} = 1. \tag{6.4}$$

6.2.2 Multi-dimensional UGF

SDN controller is considered successful when it is able to meet a required performance level (also referred to as *demand*) for each network, then a vectorial demand $\mathbf{W}(t) = (W_1(t), ..., W_k(t))$ is introduced.

In most practical cases, some redundancy may be necessary to meet demand. In this work, the focus is on the Master-Slave configuration, where replicas of the whole SDN controller (core and software/VPIs parts) are admitted: in this configuration, all VPIs relating to a single domain are synchronized and share the same information on the network, so it is not relevant which redundant VPI has managed which specific flow entry.

Then, the SDN controller is a logical node with h parallel elements without *flow dispersion* [52], and the stochastic process describing the serving capacity offered to the i-th network is

$$G_i(t) = \max_{l=1,\dots,h} G_i^{(l)}(t), \tag{6.5}$$

being $G_i^{(l)}(t)$ the i-th element of the k-tuple $\mathbf{G}^{(l)}(t)$.

The steady-state values of the random processes $G_i(t)$, for $i = 1, \dots, k$ and $t \longrightarrow \infty$, can be represented by a (discrete) random vector $\mathbf{G} = (G_1, \dots, G_k)$ with a multi-dimensional probability function $p_{\mathbf{G}}(\cdot)$, given by the steady-state distribution of the overall CTMC model provided by the parallel nodes composing the controller.

In line with [52], the controller *instantaneous availability* $A_{SDN}(t)$ is the probability that the SDN controller at $t > 0$ is in one of the states where performance is not less than demand for each network $W_i(t), i = 1, \dots, k$ (*acceptable states*), viz.

$$A_{SDN}(t) = \Pr\{G_i(t) - W_i(t) \geq 0, \forall i = 1, \dots, k\}. \tag{6.6}$$

For large t, the controller initial state has no practical influence on its availability. Therefore, given a constant demand level $W_i(t) = w, i = 1, \dots, k$, the stationary availability of the SDN controller $A_{SDN}(w)$ can be determined by

$$\begin{aligned} A_{SDN}(w) = \textstyle\sum_{j=1}^{m} p_{\mathbf{G}}\left(\mathbf{g}_j^{SDN}\right) \cdot \\ \mathbf{1}\left(g_{i,j}^{SDN} \geq w, \forall i = 1, \dots, k\right), \end{aligned} \tag{6.7}$$

where $\mathbf{g}_j^{SDN}$ identifies the j-th (vectorial) state of the SDN controller with parallel elements, being a CTMC with m states, and $\mathbf{1}(True) = 1$, and $\mathbf{1}(False) = 0$.

A convenient procedure to evaluate the system availability is based on the Universal Generating Function (UGF) method, as introduced in Sec. 2.4. The UGF of a discrete random variable X is a polynomial-shape function $u(z) = \sum_{i=1}^{I} q_i z^{x_i}$, where X has I values x_i and $q_i = \Pr\{X = x_i\}$. The UGF of the performance distribution of a multi-state system turns to be useful to evaluate its availability. Furthermore, the UGF of a series-parallel system can be efficiently computed by composing the UGFs of all subsystems through suitable operators for both series and parallel connections. See [52] for further details.

However, the UGF approach must be extended to the multi-dimensional case in order to handle performance random vectors, such as $\mathbf{G}^{(l)}$ and $\mathbf{G}$.

Consequently, it is possible to define the *Multi-dimensional* UGF (MUGF) $u(\mathbf{z})$ of the k-dimensional random vector $\mathbf{G}$, with values in the set $\{\mathbf{g}_1, ..., \mathbf{g}_m\}$ and multi-dimensional probability function $p_{\mathbf{G}}(\cdot)$, as

$$u(\mathbf{z}) = \sum_{j=1}^{m} p_{\mathbf{G}}(\mathbf{g}_j) \prod_{i=1}^{k} z_i^{g_{i,j}}, \tag{6.8}$$

where $\mathbf{z} = (z_1, ..., z_k)$.

Let h be the number of parallel elements without flow dispersion composing the SDN controller, whose serving capacity is ruled by (6.5). The MUGF of the controller can be calculated by the following π operator:

$$\begin{aligned} u_{SDN}(\mathbf{z}) &= \sum_r p_r \prod_{i=1}^{k} z_i^{g_{i,r}^{SDN}} = \\ &= \pi\left(u_1(\mathbf{z}), u_2(\mathbf{z}), ..., u_h(\mathbf{z})\right) = \\ &= \sum_{j_1=-1}^{2^k-1} \sum_{j_2=-1}^{2^k-1} \cdots \sum_{j_h=-1}^{2^k-1} \prod_{l=1}^{h} p_{j_l}^{(l)} \prod_{i=1}^{k} z_i^{\max_{l=1,...,h} g_{i,j_l}^{(l)}}, \end{aligned} \tag{6.9}$$

where $p_{j_l}^{(l)}$ are the steady-state probabilities in (6.3) corresponding to the performance k-tuple $\mathbf{g}_{j_l}^{(l)}$ in (6.2). From (6.9), we derive p_r and $g_{i,r}^{SDN}$ that are used to compute the stationary availability of the SDN controller by (6.7).

In order to minimize the cost of the SDN controller, it is relevant to find the minimal number of parallel h^* that meets a condition on the stationary availability A_0, viz.

$$h^* = \arg\min_{h \in \mathbb{N}} \left(A_{SDN}(w,h) \geq A_0 \right). \tag{6.10}$$

6.2.3 A numerical example

In this section, a numerical example is provided as an application of the methodology proposed in the previous sections. One can suppose that every Service Provider has negotiated the same Service Level Agreements (SLAs), resulting in the same number of concurrent OpenFlow sessions managed by the SDN controller. This assumption corresponds to a service demand level equal to $w = 9500$ sessions per time unit, chosen to be close to the highest performance level of a single VPI.

Each VPI has a capacity of $n = 10000$ sessions per time unit while failure and repair rates are: $\lambda_1 = 3.858 \times 10^{-7}$ sec^{-1} (corresponding to 1 fault per month for S_1), $\lambda_2 = 7.716 \times 10^{-7}$ sec^{-1} (corresponding to 2 fault per month for S_2), $\lambda_3 = 1.157 \times 10^{-7}$ sec^{-1} (corresponding to 3 fault per month for S_3), and $\mu_s = 1.666 \times 10^{-3}$ sec^{-1} (corresponding to a mean repair time of 10 min for all VPIs). It is worth noting that different failure rates for each VPI have been assumed to take into account the possible differentiation in terms of operating systems on board of each virtual operator, resulting in VPIs with different behaviours. On the contrary, one and the same value μ_s has been considered by presuming common repair operations for each VPI (typically a mere reboot is enough). Node core failure and repair rates are $\lambda_c = 9.513 \times 10^{-8}$ sec^{-1} (corresponding to 3 core faults per year) and $\mu_c = 6.944 \times 10^{-5}$ sec^{-1} (corresponding to a mean repair time of 4 hours for a technical activity on site), respectively. Table 6.2 summarizes the illustrative example data. With the considered values, the "5 nines" availability of the SDN controller requires 3 parallel elements with a stationary availability

$$A_{SDN}(w,h)\Big|_{\substack{w=9500 \\ h=3}} = p_7 = 0.999999986$$

The steady-state probabilities of a single node are derived by

Table 6.2: Numerical data

Parameter	Value
n	10000 sessions/time unit
λ_1	3.858×10^{-7} sec^{-1}
λ_2	7.716×10^{-7} sec^{-1}
λ_3	1.157×10^{-7} sec^{-1}
μ_s	1.666×10^{-3} sec^{-1}
λ_c	9.513×10^{-8} sec^{-1}
μ_c	6.944×10^{-5} sec^{-1}
w	9500 sessions/time unit
A_0	0.99999

solving (6.1) and (6.4), while the output performance levels and steady-state probabilities of SDN controller after redundancy optimization are shown in Table 6.3.

6.3 HA and Network Function Virtualization (NFV)

As enabler of fifth generation (5G) networks, NFV paradigm has been conceived to boost the deployment of new services by adapting the classical virtualization concepts to the network environments. Therefore, classic network elements (e.g. routers, firewalls, load balancers) are deployed in a cloud computing infrastructure by means of virtual machines running on general purpose hardware. The resulting architecture consists in a set of Virtualized Network Functions (VNFs), realizing novel services. A composition of VNFs is often referred to as Service Function Chain (SFC), and is typically governed by the Virtualized Infrastructure Manager (VIM), namely, the key element of the whole NFV architecture. The result is a system, referred in this section to as Network Service (NS), composed by the VIM that plays a role of the manager furnishing vital services for VNFs (deployment, fault control, network and storage resources tuning etc.), and by the SFC that

Table 6.3: Steady-state probabilities and performance levels of SDN controller after redundancy optimization.

Probability	Performance (sessions/time unit) Triples
2.561×10^{-9}	$(0, 0, 0)$
1.801×10^{-12}	$(10000, 0, 0)$
9.001×10^{-13}	$(0, 10000, 0)$
6.191×10^{-9}	$(10000, 10000, 0)$
5.999×10^{-13}	$(0, 0, 10000)$
3.5668×10^{-9}	$(10000, 0, 10000)$
1.527×10^{-9}	$(0, 10000, 10000)$
0.999999986	$(10000, 10000, 10000)$

provides the service itself by means of its VNFs. This kind of deployment raises a critical availability issue, since the fault of an individual block could compromise the whole functionality. In this section, the main purpose is to characterize the NS availability in terms of "five nines" availability requirement. Even if the goal is the same, alternative techniques (in respects of ones used for SDN controller modeling) have been exploited. Such a choice is motivated by the particular structure of a Network Service that makes it suited to be characterized by two different formalisms: *i)* the Reliability Block Diagrams (RBDs), a combinatorial model used to characterize the high level dependencies among the elements, and *ii)* the Stochastic Reward Networks (SRNs), a state-space model exploited to typify the probabilistic behavior of the underlying system structure that has been sketched in Sec. 2.6.

6.3.1 Modeling a network service via SRN

On the other hand, NFV allows the creation of an SFC by a chain of VNFs traversed in a predefined order, that in the NFV context, can be interpreted as a VNF Forwarding Graph (VNF-FG). A VNF can be regarded as independent (often geographically separated from other VNFs) element composed by the following three modules:

- *software*: a module implementing a service functionality;
- *hardware*: a module aggregating all physical components (CPU, RAM, Power Supply etc.);
- *hypervisor* (often referred to as Virtual Machine Monitor - VMM): a software layer acting as an interface between hardware and software modules.

On the other hand, the Virtualized Infrastructure Manager is a part of the MANO (MANagement and Orchestration domain) [77] that performs some critical operations as managing a set of resources (storage, networking etc.) to be deployed on demand, mapping the virtual resources on the physical ones and managing the chain of VNFs. In accordance to a common implementation based on OpenStack, we consider the VIM (in a virtualized implementation) as composed by five modules: hardware, hypervisor (similar to those in SFC), and other three elements:

- *database*: devoted to store the critical data of the whole system (inventory of hardware resources, register of available VNFs and others);
- *HAproxy*: the High Availability proxy acts as a load balancer distributing the load among the redundant VIM nodes;
- *functional blocks*: a collection of sub-elements accomplishing several functionalities, such as nova-keystone (providing authorization and authentication mechanisms), nova-scheduler (dispatcher of computational requests), rabbitmq (allowing the communication among internal components).

As before said, from an infrastructural viewpoint, it is possible to represent the NS in terms of an RBD as illustrated in Fig. 6.4, where it has been emphasized that an SFC is a chain of VNF nodes. Being a series configuration, the NS is available (working)

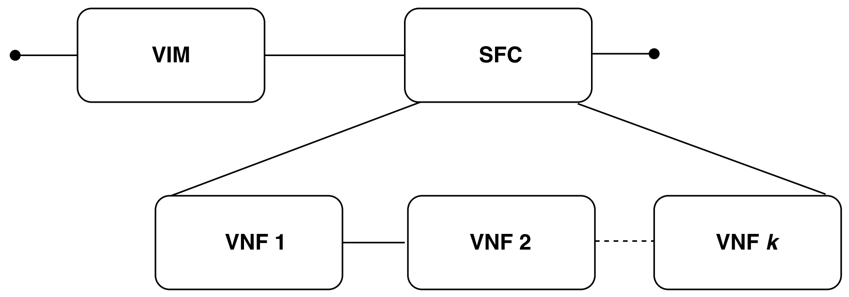


Figure 6.4: RBD representation of the Network Service obtained by aggregating VIM and SFC. The latter is in turn composed by a sequence of VNFs.

when each subsystem (VIM, VNF_1,...,VNF_k) is available (working). Let's start to separately analyze the SRN models for VIM and for SFC (and hence for the VNFs) respectively. After, the two models will be recombined in order to form the original Network Service.

SRN Model of VIM

An SRN model is based on a Stochastic Petri Net and is represented by a bipartite directed graph, where a *place* (drawn as a circle) accounts for a specific condition (e.g. a system element up or down) and can contain one or more *tokens*, namely a parameter value associated to a condition. The distribution of tokens (at a time t) is referred to as *marking*, and can be described by a vector $\mathbf{m} = (m_1, m_2, \ldots, m_k)$ where m_h is the number of tokens in the place h. A *transition* (drawn as a rectangle) accounts for a specific event (e.g. a system element that crashes), and lets a token to be transferred from a place to another one, resulting in a marking alteration. In particular, we discriminate between: *timed transitions* (drawn as unfilled rectangles), associated to the events whose times are assumed to be exponentially distributed random variables with a given rate parameter (*firing rate*), and *immediate transitions* (drawn as thin and filled rectangles), whose

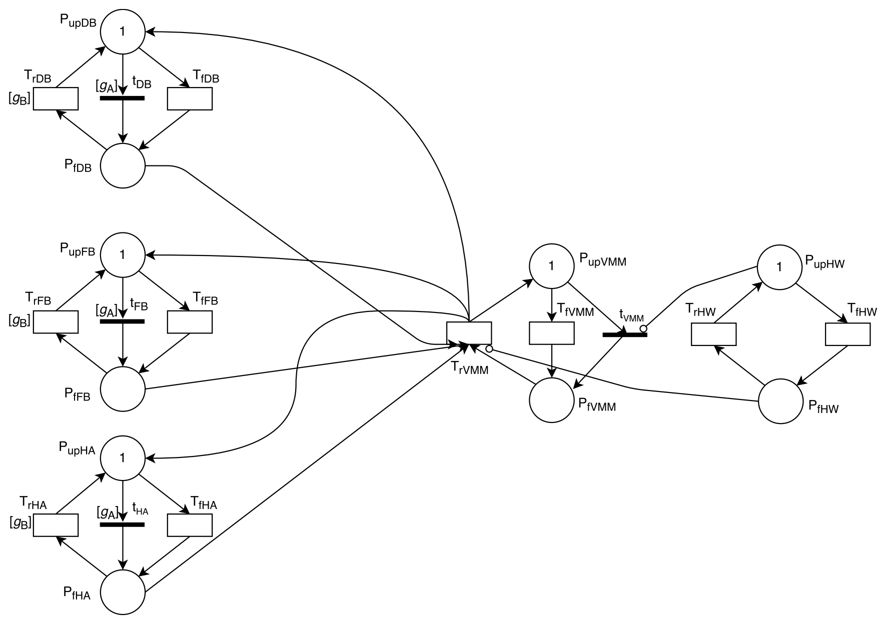


Figure 6.5: SRN model of the VIM node. Such model follows the classical OpenStack deployment.

transition time is equal to zero. When the firing rate of timed transitions depends on tokens distribution in the SRN graph, we use the symbol (#) nearby the correspondent transition and we refer to as *marking dependent transitions.* A transition can be optionally controlled by a *guard function* (indicated by g) that assumes value 1 when transition has to be enabled. Besides, a transition can be inhibited by an *inhibitory arc* (depicted as a line with a blank unfilled circle nearby the correspondent transition).

The SRN model associated to the VIM node is reported in Fig. 6.5. It is useful to remark that the VIM subsystem is composed by N redundant VIM nodes. Places P_{upDB} [resp. P_{fDB}], P_{upFB} [P_{fFB}], P_{upHA} [P_{fHA}], P_{upVMM} [P_{fVMM}] and P_{upHW} [P_{fHW}] indicate the conditions where the database, the functional blocks, the HAproxy, the hypervisor and the hardware of the VIM are up [down], respectively. The numbers in the places (tokens) represent the corresponding initial conditions. The transitions T_{fDB}

Table 6.4: Guard Functions defined on the SRN model of VIM structure

Guard Function	Value
g_A	1 if # $P_{fVMM} = 1$, 0 otherwise
g_B	1 if # $P_{upVMM} = 1$, 0 otherwise

$[T_{rDB}]$, T_{fFB} $[T_{rFB}]$, T_{fHA} $[T_{rHA}]$, T_{fVMM} $[T_{rVMM}]$ and T_{fHW} $[T_{rHW}]$ model the time to failure [repair] of database, functional blocks, HAproxy, hypervisor and hardware, respectively. Let us now briefly discuss the dynamics of such SRN. We start by considering a fully working system (namely all the elements are up and running). As an exemplary case, I focus on the sub-model of the HAproxy (similar considerations hold for database, functional blocks, hypervisor and hardware modules). When HAproxy fails, the transition T_{fHA} is *fired* and the token removed from P_{upHA} is deposited into P_{fHA}. In the considered model I also introduce some immediate transitions to cope with common cause failures: t_{DB}, t_{FB} and t_{HA} (associated to database, functional blocks and HAproxy, respectively) that are fired when the transition T_{fVMM} is fired, meaning that a hypervisor failure implies the three virtual modules failure as well. Similarly, the immediate transition t_{VMM} accounts for a hypervisor failure as a consequence of a hardware failure. The *inhibitory arc* between P_{upHW} and t_{VMM} compels the hypervisor failure in case of hardware failure. The *inhibitory arc* between P_{fHW} and T_{rVMM} forbids the hypervisor repair in case of hardware failure. Besides, in order to formally describe the dependencies among hypervisor and the three virtual modules (the case of hardware failure is included in the case of hypervisor failure) it is possible to introduce two guard functions g_A and g_B described in Table 6.4. The guard function g_A enables t_{DB}, t_{FB} and t_{HA} when hypervisor fails, namely, when a token is moved from P_{upVMM} to P_{fVMM}. The guard function g_B, instead, inhibits the repair of the three virtual modules in case of hypervisor failure.

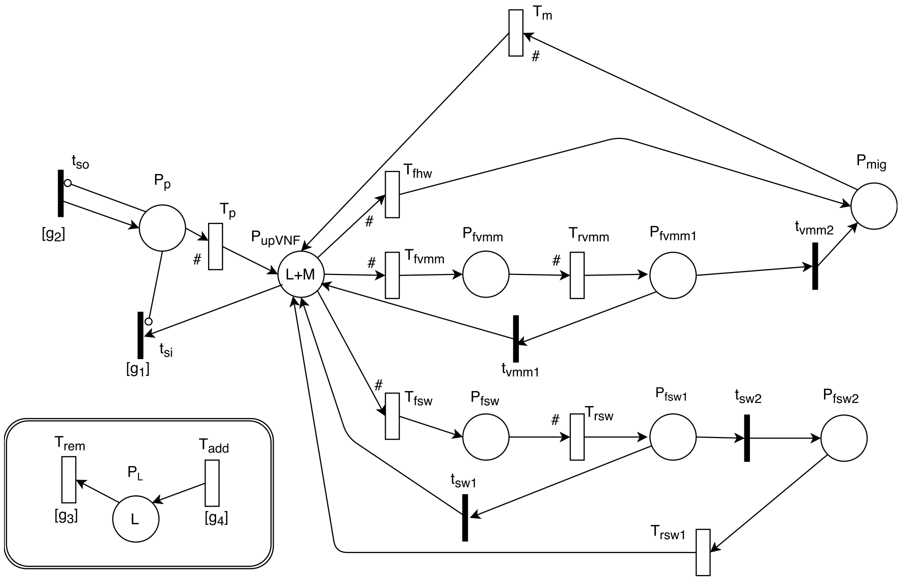


Figure 6.6: SRN model of a single Virtualized Network Function.

SRN Model of VNF

In this model, I adopt the realistic assumption that the number of deployed VNFs can vary dynamically with the time, in accordance to a pay-per-use cloud model. In an exemplary scenario, I consider L replicas sharing a (time-varying) load, and M (extra) replicas needed to satisfy a certain availability requirement. Thus, when the load increases or decreases (as per day/night variations), the number of replicas L changes, resulting in a possible variation of the number of M replicas in order to preserve the desired availability.

Figure 6.6 shows the model of a single VNF. The place P_{upVNF} indicates the VNF working condition, implying that the hardware, software and virtual resources are correctly working. The token value inside P_{upVNF} amounts to $L + M$, namely, the number of initial working VNFs replicas. Let us analyze directly the evolution of such model. First of all I take into account the Scale-Out (S-O) and Scale-In (S-I) operations corresponding to a provisioning phase (deploying replicas) and a de-provisioning phase (undeploying replicas) respectively. Thus, when an S-O operation is requested, t_{so} is fired and a token is deposited in place P_p, modeling the condition of a replica requested but not working yet, until the token enters P_{upVNF} after T_p is fired. It is worth noting that the *inhibitory arc* from P_p to t_{so} models the impairment of multiple provisioning stages. On the contrary, when an S-I operation is requested, t_{si} is fired and the *inhibitory arc* from P_p to t_{so} preventing S-I operations during provisioning stages. In case of an hardware failure, T_{fhw} is fired and a token passes from P_{upVNF} to P_{mig}, being the latter a place ruling a migration process (resources are transferred into another hardware platform with no state loss). Once T_m is fired, the migration is completed and the token can come back into P_{upVNF}. In case of a hypervisor failure, T_{fvmm} is fired and a token enters P_{fvmm}. The repair procedure is governed by T_{rvmm} that, when fired, lets the token move into P_{fvmm1}. In this place, two alternatives are admissible: *i)* the VMM repair procedure is successful (e.g. a simple *reboot* solves the problem) with a certain probability p_{vmm}, resulting in firing transition t_{vmm1} and the returning in the working place P_{upVNF}; *ii)* the vmm repair procedure is unsuccessful with probability $(1 - p_{vmm})$, the transition t_{vmm2} is fired, the token reaches place P_{mig}, and the migration process described previously is activated. In case of software failure, T_{fsw} is fired and the token is deposited in P_{fsw}; the repair process is ruled by T_{rsw} and the token passes into P_{fsw1}. Similar to the previous case, two options are allowed: *i)* the software repair procedure is successful with a certain probability p_{sw}, thus t_{sw1} is fired and initial condition is gained; *ii)* the repair procedure is vain with probability $(1-p_{sw})$, hence t_{sw2} is fired and place P_{fsw2} (indicating a tough fault condition needing a repairman) is entered; once repair process is terminated, T_{rsw1} is fired and place P_{upVNF} is reached.

Finally, place P_L (see the boxed sub-model) takes into account the condition of L replicas that can be added (T_{add}) or removed (T_{rem}). It is useful to note that transitions T_p, T_{fhw}, T_{fvmm}, T_{fsw}, T_{rsw}, T_{rvmm} and T_m, depend only on the number of tokens in their

Table 6.5: Guard Functions defined on the SRN model of single VNF

Guard Function	Value
g_1	1 if $N_{tot} > L +$ M , 0 otherwise
g_2	1 if $N_{tot} < L +$ M , 0 otherwise
g_3	1 if $L < L_{max}$, 0 otherwise
g_4	1 if $L > L_{min}$, 0 otherwise

originating places, and their overall firing rates are proportional to those numbers.

The guard functions present in this model are defined as follows. Said $\#P_k$ the number of tokens in the place k[1], it is possible to define N_{tot} the number of VNFs replicas in the SRN at time t as: $N_{tot} = \#P_p + \#P_{upVNF} + \#P_{fvmm} + \#P_{fsw} + \#P_{fsw2} + \#P_{mig}$, where omit the time dependence is omitted [2]. Guard g_1 inhibits S-I operations when N_{tot} undergoes the sum $L + M$. Similarly, g_2 inhibits S-O operations when N_{tot} exceeds the sum $L + M$. Guard g_3 prevents that L exceeds the maximum number of admissible replicas (L_{max}) and g_4 prevents that L could be lower than the minimum number of admissible replicas (L_{min}), as summarized in Table 6.5.

SRN Model of Network Service

It's now the time to consider the whole NS, given by the combination of N VIM replicas and the $L + M$ replicas of the three VNFs arranged in a chain. As regards the VIM, let r_i be the reward value assigned to marking i and $p_i(t)$ the probability for SRN in Fig. 6.5 to be in marking i at time t; according (2.17) it is possible to express the instantaneous availability $A_{VIM}(t)$ as:

[1] In Petri Net jargon, there is a little abuse of (#) symbol that indicates either number of tokens, and marking-dependent transitions.

[2] Places P_{fvmm1} and P_{sw1} do not contribute to N_{tot} because the time spent in such places is zero.

$$A_{VIM}(t) = \sum_{i \in I} r_i p_i(t), \tag{6.11}$$

where I is the set of *tangible markings* (markings where no immediate transitions are enabled). The instantaneous availability $A_{VIM}(t)$ in fact, is the probability that the VIM is available at time t and, being the markings mutually exclusive, can be expressed as the sum of the probabilities of all markings that at time t result in a working condition for the VIM subsystem.

The reward value r_i assumes value 1 for the markings identifying the VIM working conditions, namely all the operating conditions where at least two Database instances (as typical in many active-active configurations), one Functional Block and one HAproxy are active. For all the remaining states, r_i is set to 0. Thus, the reward value can be written as:

$$r_i = \begin{cases} 1 & \text{if} \quad \left(\sum_{k=1}^{N} \#P_{upDB}(k) \geq 2\right) \wedge \\ & \quad \left(\sum_{k=1}^{N} \#P_{upFB}(k) \geq 1\right) \wedge \\ & \quad \left(\sum_{k=1}^{N} \#P_{upHA}(k) \geq 1\right) \\ \\ 0 & \text{otherwise,} \end{cases}$$

where k goes from 1 to the number of replicas N, and $\#P_{upDB}(k)$, $\#P_{upFB}(k)$ and $\#P_{upHA}(k)$, indicate the number of tokens in the "up" places of virtual modules, for the $k-th$ parallel element. The hardware and hypervisor "up" conditions do not appear, being included in expression when at least 1 virtual module is active. The VIM steady-state availability is given by (6.11) as $t \rightarrow \infty$,

$$A_{VIM} = \lim_{t \rightarrow +\infty} A_{VIM}(t) = \sum_{i \in I} r_i p_i, \tag{6.12}$$

where p_i is the steady-state probability of state i, i.e. $p_i =$

$\lim_{t \to +\infty} p_i(t)$. A similar reasoning holds for the case of VNF. Let s_j be the reward value assigned to marking j and $q_j(t)$ the probability for SRN in Fig. 6.6 to be in marking j at time t. The expected reward value at time t for the VNF model corresponds to:

$$A_{VNF}(t) = \sum_{j \in J} s_j q_j(t), \tag{6.13}$$

where J identifies the set of *tangible markings* of the VNF model. In this case, the reward value s_j associated to the tangible marking j follows:

$$s_j = \begin{cases} 1 & \text{if } (\#P_{upVNF} \geq \#P_L) \\ 0 & \text{otherwise.} \end{cases}$$

The VNF working condition ($s_j = 1$) occurs when the number of total replicas (represented by the tokens in P_{upVNF}) is no less than the number of regular replicas (represented by the tokens in P_L). Thus, steady-state availability for a VNF is obtained by (6.13) as $t \to \infty$,

$$A_{VNF} = \lim_{t \to +\infty} A_{VNF}(t) = \sum_{j \in J} s_j q_j, \tag{6.14}$$

where q_j is the steady-state probability given by $q_j = \lim_{t \to +\infty} q_j(t)$.

Now, we are able to evaluate the steady-state availability of the whole Network Service, composed by the series of VIM and the 3 VNFs according to the RBD representation in Fig. 6.4.

The resulting Network Service availability A_{NS} can be expressed as the product of the availabilities associated with the VIM and SFC subsystems, namely

$$A_{NS} = A_{VIM} \prod_{m=1}^{3} A_{VNF}^{(m)}, \tag{6.15}$$

where A_{VIM} is derived by (6.12) and A_{VNF} is computed by (6.14), where m denotes the VNF m.

6.3.2 Experimental results

An exemplary application of the proposed approach is now provided, where system parameters assume specific values suggested by the technical literature (e.g. [78]). The values of parameters associated to VIM and to VNF are reported in Tables 6.6 and 6.7, respectively. In order to respect the notation used in Figs. 6.5 and 6.6, we use uppercase subscripts for the VIM parameters and lower case subscripts for the VNF parameters. The present availability analysis has been carried out with the help of SHARPE [79], a tool that allows to analyze SRN availability models. According to elasticity concepts typical of cloud environments, I target two exemplary different operating conditions, namely c_1 and c_2 associated to a dynamically variable load managed by the system, that results in a different number of deployed replicas. I suppose that in condition c_1 (low load) VNFs share the load among two replicas ($L_{c1} = 2$), while in condition c_2 (high load) they share the load

Table 6.6: Input parameters for the SRN representing the VIM

Parameter	Description	Value [a)]
$1/\lambda_{VM}$	mean time to (DB,FB,HA) failure	3000 hours
$1/\lambda_{VMM}$	mean time to hypervisor failure	5000 hours
$1/\lambda_{HW}$	mean time to hardware failure	60000 hours
$1/\mu_{VM}$	mean time to (DB,FB,HA) repair	1 hour
$1/\mu_{VMM}$	mean time to hypervisor repair	2 hours
$1/\mu_{HW}$	mean time to hardware repair	8 hours

[a)] I assume the same failure/repair rate values (being deployed on similar VMs) for database, functional blocks and HAproxy.

Table 6.7: Input parameters for the SRN representing the VNF

Parameter	Description	Value
$1/\lambda_{hw}$	mean time to hardware failure	60000 hours
$1/\lambda_{sw}$	mean time to software failure	3000 hours
$1/\lambda_{vmm}$	mean time to hypervisor failure	5000 hours
$1/\mu_{sw}$	mean time to easy software repair	7 minutes
$1/\mu_{sw1}$	mean time to tough software repair	2 hours
$1/\mu_{vmm}$	mean time to hypervisor repair	10 minutes
$1/\alpha_p$	mean time to provisioning	20 minutes
$1/\alpha_m$	mean time to migration	20 minutes
$1/\alpha_s$	mean time to scaling (S-I/S-O) procedures	12 hours
p_{sw}	probability of successful software repair	0.98
p_{vmm}	probability of successful hypervisor repair	0.99

Table 6.8: Availability Results of the whole Network Service

Setting	Redundancy Level	A_{NS}
S_1	$L_{c1} = 2$, $L_{c2} = 3$, $M_{c1} = 1$, $M_{c2} = 1$, $N = 3$	0.99998444
S_2	$L_{c1} = 2$, $L_{c2} = 3$, $M_{c1} = 1$, $M_{c2} = 2$, $N = 3$	0.99999080
S_3	$L_{c1} = 2$, $L_{c2} = 3$, $M_{c1} = 2$, $M_{c2} = 2$, $N = 3$	0.99999084
S_4	$L_{c1} = 2$, $L_{c2} = 3$, $M_{c1} = 1$, $M_{c2} = 1$, $N = 4$	0.99998686
S_5	$L_{c1} = 2$, $L_{c2} = 3$, $M_{c1} = 1$, $M_{c2} = 2$, $N = 4$	0.99999323
S_6	$L_{c1} = 2$, $L_{c2} = 3$, $M_{c1} = 2$, $M_{c2} = 2$, $N = 4$	0.99999326

among three replicas ($L_{c2} = 3 \geq L_{c1}$).

Accordingly, it is possible to characterize the system in terms of the number of extra replicas M guaranteeing the "five nines" availability requirement, namely M_{c1} and $M_{c2} \geq M_{c1}$ in the two conditions of VNFs. I investigate also the influence of the number of VIM replicas N. It is useful distinguish 6 settings $S_1, ..., S_6$ representing different configurations, as reported in table 6.8 with their corresponding availability value A_{NS}. Figure 6.7 reports the main results where, for visual comfort, the steady-state unavailability of the Network Service $(1 - A_{NS})$ in said exemplary settings

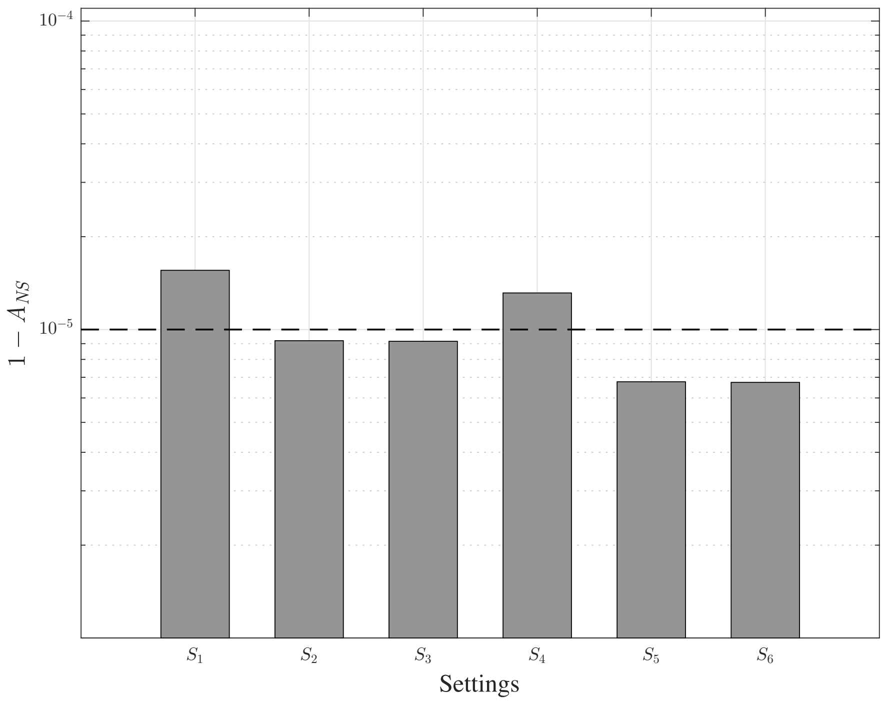


Figure 6.7: Unavailability $1 - A_{NS}$ of the Network Service for 6 settings $S_1, ..., S_6$ representing various replicas arrangements. The horizontal dashed line represents the "five nines" requirement ($1 - A_{NS} = 10^{-5}$).

is shown. By comparing S_1, S_2, S_3 with S_4, S_5, S_6, it is evident that an increase from $N = 3$ to $N = 4$ improves the NS availability. For $N = 3$, instead of considering greater values of N for $M_{c1} = M_{c2} = 1$ (setting S_1), it is possible to achieve the "five nines" requirement by deploying $M_{c2} = 2$ replicas (setting S_2). By incrementing M_{c1} (setting S_3), a negligible increment of NS availability is obtained. Similar considerations can be provided for the $N = 4$ case (setting S_4, S_5, S_6). Given a "five nines" availability constraint, S_2 is the minimal cost setting in terms of deployed replicas.

Sensitivity Analysis

Generally, a sensitivity analysis concerns the robustness of the system with respect to deviations of some system parameters from their nominal values. I analyze the influence of two crucial parameters on the availability of the minimal cost setting S_2, such as:

- λ_{SW}, influencing the transition T_{fsw} of the VNF model reported in Fig. 6.6 governing the software failures;
- $\lambda_{VMM}(= \lambda_{vmm})$, influencing both the transition T_{fVMM} of the VIM model (Fig. 6.5), and the transition T_{fvmm} of the the VNF model (Fig. 6.6) (since I assume the same hypervisor for both).

Figure 6.8 shows that the reciprocal of failure rate of the software part can be relaxed from 3000 hours (nominal value) to 2500

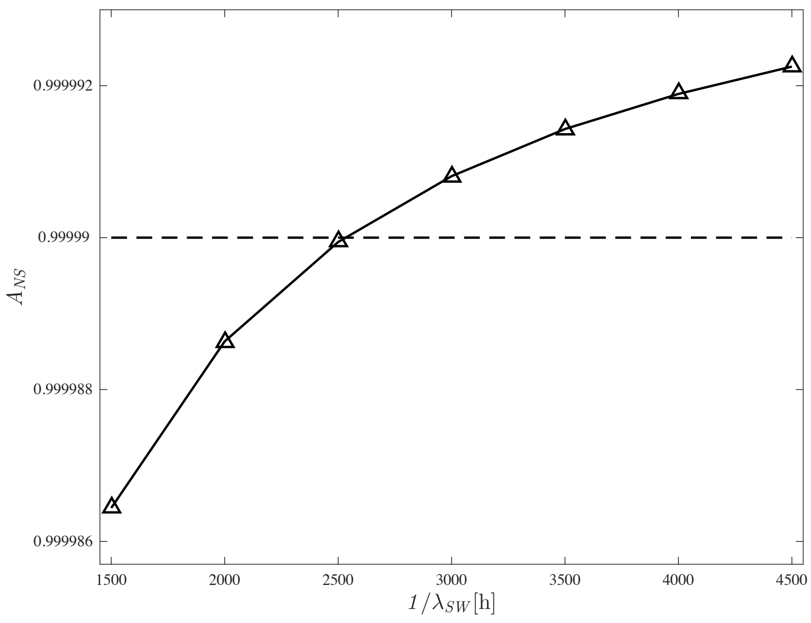


Figure 6.8: Influence of the software failure rate on the overall system.

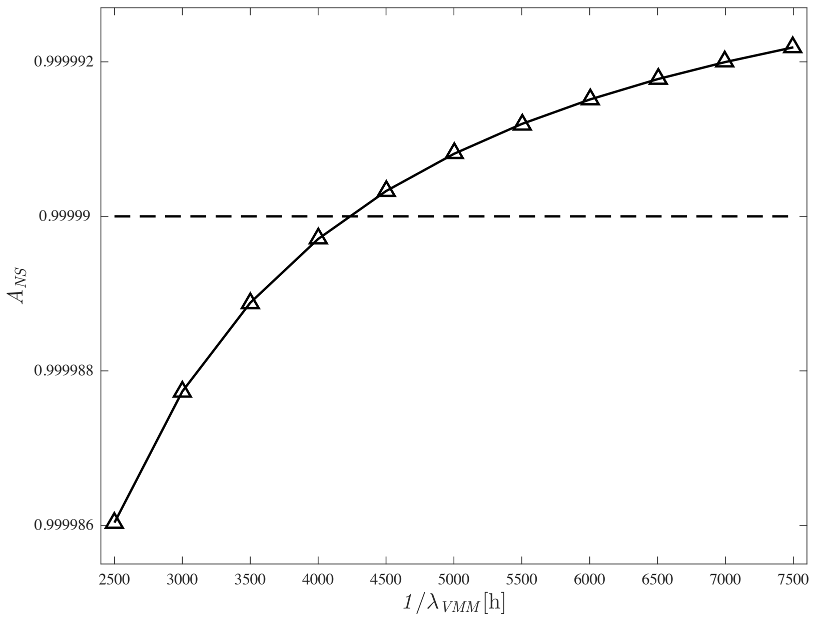


Figure 6.9: Influence of the hypervisor failure rate on the overall system (same hypervisor for both VIM and VNFs).

hours by still guaranteeing the "five nines" requirement indicated with a horizontal dashed line. Similarly, Fig. 6.9 shows that the working value of 5000 hours for $1/\lambda_{VMM}$ can be reduced to 4300 hours with no side effects on the desired availability condition.

Chapter 7

Concluding remarks

Nowadays, the problem of characterizing, modeling, revealing, and mitigating network massive attacks is intriguing and challenging. Within the present work, the candidate intends to offer a statistical perspective about the aforementioned aspects. Actually, three main macro contributions emerge in this work. The first twos concern the security aspects and models that characterize *i)* propagative and *ii)* distributed attacks, considered as two sides of the same coin, even if they exhibit different peculiarities. The third contribution pertains to the resilience and availability concepts *iii)* of network infrastructures as a general prevention strategy (not only against network attack).

As regards the first contribution, the candidate proposes to adopt Kendall′s birth-and-death process as an analytical model to characterize the threat propagation phenomenon. In particular, two properties emerge: firstly, it represents a stochastic scheme useful to account for the complexity of real-world data networks; secondly, it is able to catch the basic features of threat propagation effect by means of some parameters with a meaningful physical interpretation. By using notable properties of the adopted Kendall′ model, the optimal resource allocation problem (namely, the optimal mitigation strategy) has been solved in two conditions: known and unknown (estimated via MLE) infection parameters.

As regards the second contribution, the candidate introduces a formal model for a randomized DDoS attack, where a botnet is able to emulate a normal traffic by picking legitimate messages from a dictionary built by a botmaster. Such a model allows to characterize the botnet *learning ability*, and to discover the nature of a node (malicious or normal) through an indicator called MIR (Message Innovation Rate). By exploiting such a MIR, an algorithm allowing to reveal a possibly hidden botnet in the network has been designed from the scratch. Actually, the presented contribution has been enriched by considering a further sophistication of the randomized DDoS attack, where a botnet is spread among many clusters with its own emulation dictionary. Accordingly, the aforementioned algorithm has been extended to take into account the multi-clustered setting.

As regards the third (and more recent) contribution, the candidate proposes a formalization of the network availability issue, broadly intended as designing an optimal redundant strategy for the network infrastructure useful to account for undesirable events (failures, attacks, etc.). At this aim, two frameworks are introduced: *a)* the Universal Generating Function (UGF), where an extension to cope with multi-dimensional quantities referred to as Multi-dimensional UGF (MUGF) has been devised; *b)* the Stochastic Reward Networks (SRNs), useful to model complex state-spaced systems. UGF and SRN techniques have been evaluated in a Software Defined Networking scenario and in a Network Function Virtualization (NFV) environment, respectively.

The problem of analyzing and mitigating distributed attacks is interesting and multifaceted, so, many open issues emerge to address future works. As regards to the propagative nature of such attacks, further investigation could be devoted at exploring the connection between the threat diffusion mechanisms and the underlying network topology where interesting results derived from percolation theory could be exploited. As regards to the distributed nature of threats, a possible extension of the proposed randomized DDoS model could concern the adoption of more sophisticated distributions to characterize the transmission scheduling activity of the botnet that, in this work, is assumed to follow a Poisson law. Another possibility is to consider a more challenging multi-clustered setting, where the clusters could have the property of being partially or almost totally overlapped. Finally, as regards the availability issues, an interesting progress could regard the adoption of more realistic models (e.g. Weibull, Lognormal) to characterize failures and repairs during the transient regimes.

Appendix A

Appendices

A.1 Appendix 1

Proof of Property 2. The various claims of the theorem will be proved following the route traced in [80]. Specifically, for a given random process to converge in distribution, we must prove that: *i*) the MGF, for each t, exists for $|x| < r_1$, and that *ii*) the MGF converges to a limiting function for all $|x| \leq r_2 < r_1$. Under these conditions, it is shown in [80] that the limiting function defines, for $|t| \leq r_2$, the MGF of the limiting distribution.

We start by considering the case $\rho < 1$. We know that the range for the MGF is $x < \ln(1/\pi_t)$. We observe that:

$$\ln(1/\pi_t) \quad = \quad \ln(1/\rho) + \ln \frac{1 - \rho e^{\Delta t}}{1 - e^{\Delta t}} > \ln(1/\rho). \qquad \text{(A.1)}$$

Accordingly, we can set $r_1 = \ln(1/\rho)$. Since in view of (3.12):

$$\lim_{t \to \infty} \pi_t = \lim_{t \to \infty} q_t = \rho. \qquad \text{(A.2)}$$

we conclude that, for all $|x| \leq r_2 < r_1$, the MGF in (3.13) converges to:

$$\lim_{t \to \infty} \Psi(x; t) = \left(\frac{1 - \rho}{1 - \rho\, e^x} \right)^\eta \qquad \text{(A.3)}$$

which follows easily by noting that, The MGF in (3.13) converges to the MGF of a negative binomial random variable with parameters η and ρ [61]. Let us move on considering the case $\rho > 1$. We must therefore focus on weak convergence of the *scaled* random process $I(t)e^{-\Delta t}$. For ease of notation, we set $\delta_t \triangleq e^{-\Delta t}$, and introduce the MGF of the scaled process, namely,

$$\begin{aligned}\tilde{\Psi}(x;t) &\triangleq \mathbb{E}[e^{xI(t)\delta_t}] = \Psi(x\delta_t;t) \\ &= \left(\frac{1-\pi_t}{1-\pi_t\, e^{x\delta_t}}\right)^{\eta+n_0}\left(\frac{1-q_t\, e^{x\delta_t}}{1-q_t}\right)^{n_0}. \end{aligned} \tag{A.4}$$

The range for the MGF of the scaled process is $x < e^{\Delta t}\ln(1/\pi_t)$. We have that:

$$e^{\Delta t}\ln(1/\pi_t) = e^{\Delta t}\ln\frac{1-e^{-\Delta t}/\rho}{1-e^{-\Delta t}} > (1-1/\rho). \tag{A.5}$$

where we used the inequality $\ln\frac{1-ax}{1-x} > (1-a)x$ that holds when $0 < a < 1$ and $x > 0$. Accordingly, we set $r_1 = 1 - 1/\rho$. Since δ_t vanishes as $t \to \infty$, we use the known Taylor's approximation $e^{x\delta_t} \approx 1 + x\delta_t$, yielding:

$$\begin{aligned}\tilde{\Psi}(x;t) \approx &\left(\frac{1-\pi_t}{1-\pi_t-\pi_t x\delta_t}\right)^{\eta+n_0}\left(\frac{1-q_t-q_t x\delta_t}{1-q_t}\right)^{n_0} \\ &\left(\frac{1}{1-\pi_t x\frac{\delta_t}{1-\pi_t}}\right)^{\eta+n_0}\left(1-q_t x\frac{\delta_t}{1-q_t}\right)^{n_0}. \end{aligned} \tag{A.6}$$

We see from the definitions in (3.12) that:

$$1-\pi_t \triangleq \frac{1-1/\rho}{e^{\Delta t}-1/\rho}, \quad 1-q_t \triangleq \frac{\rho-1}{e^{\Delta t}-1}, \tag{A.7}$$

and, hence, we have that:

$$\lim_{t\to\infty}\frac{\delta_t}{1-\pi_t} = \frac{\rho}{\rho-1}, \quad \lim_{t\to\infty}\frac{\delta_t}{1-q_t} = \frac{1}{\rho-1}, \tag{A.8}$$

yielding:

$$\lim_{t\to\infty} \tilde{\Psi}(x;t) = \left(\frac{1}{1-\frac{x\rho}{\rho-1}}\right)^{\eta+n_0} \left(1-\frac{x}{\rho-1}\right)^{n_0}. \tag{A.9}$$

which is the MGF in (3.19), with choices: $r = \eta, s = \rho$, and $m = m_0$. Finally, the result for the intermediate regime $\rho = 1$ can be obtained as done for the case $\rho > 1$, using the expressions for π_t and q_t reported in (3.16), and defining $\delta_t = 1/(\lambda t)$. In this case, the range for the MGF of the scaled process is $x < (\lambda t)\ln(1/\pi_t)$, and we have that:

$$(\lambda t)\ln(1/\pi_t) = (\lambda t)\ln\frac{1}{1+1/(\lambda t)} > 1. \tag{A.10}$$

Accordingly, we set $r_1 = 1$. Then, using 3.16 we obtain:

$$\lim_{t\to\infty}\frac{\delta_t}{1-\pi_t} = \lim_{t\to\infty}\frac{\lambda t+1}{\lambda t} = 1, \quad \lim_{t\to\infty}\frac{\delta_t}{1-q_t} = \frac{\lambda t}{\lambda t} = 1, \tag{A.11}$$

finally yielding:

$$\lim_{t\to\infty}\Psi(x\delta_t;t) = \frac{1}{(1-x)^{\eta}}, \tag{A.12}$$

which corresponds to a unit-mean gamma random variable with shape parameter equal to η. □

Lemma 1. *Let $D > 0$. For all $D \geq \Delta$, the random process $I(t)e^{-Dt}$ admits almost surely a (finite) limit.*

Proof. Let us preliminarily evaluate the expectation of $I(t)$, namely,

$$m(t) \triangleq \mathbb{E}[I(t)]. \tag{A.13}$$

Note that $m(t)$ can be obtained by taking the first derivative of the moment generating function in (3.13), and evaluating it for $x = 0$. However, it is perhaps simpler to obtain it in the following direct manner. We have that:

$$\mathbb{E}[I(t+\epsilon)|I(t)=n] = n + (n+\eta)\lambda\epsilon - n\mu\epsilon. \tag{A.14}$$

Taking expectation with respect to $I(t)$, and letting $\epsilon \to 0$, we get:

$$\frac{dm}{dt} = \Delta m + \lambda \eta \tag{A.15}$$

Solving (A.15) with the initial condition $m(0) = I(0) = n_0$ gives:

$$m(t) = \left(n_0 + \frac{\rho\eta}{\rho - 1}\right) e^{\Delta t} - \frac{\rho\eta}{\rho - 1}. \tag{A.16}$$

Due to Markovianity, Eq. (A.16) implies also that, for $0 \leq s < t$:

$$\mathbb{E}[I(t)|\{I(\tau)\}_{\tau \leq s}] = \left(I(s) + \frac{\rho\eta}{\rho - 1}\right) e^{\Delta(t-s)} - \frac{\rho\eta}{\rho - 1}. \tag{A.17}$$

Let us now introduce the following shifted-scaled version of $I(t)$, namely,

$$Z(t) = \left(I(t) + \frac{\rho\eta}{\rho - 1}\right) e^{-Dt}. \tag{A.18}$$

with $D > 0$ and $D \geq \Delta$. Using (A.17), it is readily verified that:

$$\mathbb{E}[Z(t)|\{Z(\tau)\}_{\tau \leq s}] = Z(s) e^{-(D-\Delta)(t-s)} \leq Z(s), \tag{A.19}$$

which means that the process $Z(t)$ is a *supermartingale*. Using now (A.16) into (A.18), we also see that, by application of the triangle inequality:

$$\begin{aligned} \mathbb{E}[|Z(t)|] &\leq m(t) e^{-Dt} + \frac{\rho\eta}{|\rho - 1|} e^{-Dt} \\ &= \left(n_0 + \frac{\rho\eta}{\rho - 1}\right) e^{-(D-\Delta)t} \\ &+ \rho\eta e^{-Dt} \left(\frac{1}{|\rho - 1|} - \frac{1}{\rho - 1}\right). \end{aligned} \tag{A.20}$$

Now, according to the *Martingale Convergence Theorem*, since $\sup_{t \geq 0} \mathbb{E}[|Z(t)|] < \infty$, we can conclude that the limit of $Z(t)$ exists and is finite almost surely [73]. Such result immediately implies that also the limit of $I(t)e^{-Dt}$ exists and is finite almost surely. $\square$

A.2 Appendix 2

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 $\bar{X}$.

Proposition 1 (Useful recursion)**.** *Let $a, c > 0$, $b \in \mathbb{R}$, $n \in \mathbb{N}$, $\eta_n = 1 - 1/(c + an)$, and $f_n = \eta_n f_{n-1} + b$. We have:*

$$f_n = f_0 \prod_{\ell=1}^{n} \eta_\ell + b \left(1 + \sum_{k=2}^{n} \prod_{\ell=k}^{n} \eta_\ell \right), \tag{A.21}$$

or:

$$f_n = f_0 \prod_{\ell=1}^{n} \eta_\ell + \frac{ab}{1+a} \left[n + \left(1 + \frac{c}{a}\right) \left(1 - \prod_{\ell=1}^{n} \eta_\ell \right) \right], \tag{A.22}$$

and the following limit holds:

$$\lim_{n\to\infty} \frac{f_n}{n} = \frac{ab}{1+a} \tag{A.23}$$

Proof. First, observe that:

$$f_1 = f_0\eta_1 + b, \qquad f_2 = f_0\eta_1\eta_2 + b(1+\eta_2), \ldots \tag{A.24}$$

which yields (A.21) by recursion. Let now $\hat{f}_n$ denote the RHS in (A.22). By the induction principle, the claim in (A.22) will be proved if we show that $f_1 = \hat{f}_1$, and that

$$f_n = \hat{f}_n \Rightarrow f_{n+1} = \hat{f}_{n+1} \tag{A.25}$$

Making explicit the definition of η_1 where needed, we have:

$$\hat{f}_1 = \eta_1 f_0 + \frac{ab}{1+a}(1 + 1/a) = \eta_1 f_0 + b = f_1. \tag{A.26}$$

Assuming now that $f_n = \hat{f}_n$, we can write

$$\begin{aligned} f_{n+1} = \eta_{n+1}\,\hat{f}_n + b = f_0 \prod_{\ell=1}^{n+1} \eta_\ell - \frac{ab}{1+a}\left(1+\frac{c}{a}\right)\prod_{\ell=1}^{n+1}\eta_\ell \\ + \quad \underbrace{\frac{ab}{1+a}\left(n+1+\frac{c}{a}\right)\left(1-\frac{1}{c+a(n+1)}\right)}_{=\frac{ab}{1+a}\left[n+1+\left(1+\frac{c}{a}\right)\right]} + b = \hat{f}_{n+1}. \end{aligned} \tag{A.27}$$

Finally, the claim in (A.23) follows by observing that the term $\prod_{\ell=1}^{n} \eta_\ell$ in (A.22), vanishes as $n \to \infty$. $\square$

Corollary 1 (Small perturbations)**.** *Let $a, b > 0$, $n \in \mathbb{N}$, and let f_n be a nonnegative sequence such that:*

$$f_n \leq f_{n-1}\left(1 - \frac{1}{an + o(n)}\right) + b + o(1). \tag{A.28}$$

Then:

$$\limsup_{n\to\infty} \frac{f_n}{n} \leq \frac{ab}{1+a}. \tag{A.29}$$

If the inequality in (A.28) is reversed, the constant b can be relaxed to be an arbitrary real number, and:

$$\liminf_{n\to\infty} \frac{f_n}{n} \geq \frac{ab}{1+a}. \tag{A.30}$$

Proof. Clearly, it suffices to prove (A.29). In the following, $\epsilon > 0$ is an arbitrarily small constant. For n large enough, and for all $c \in \mathbb{R}$, we have:

$$0 < 1 - \left(\frac{1}{an + o(n)}\right) \leq 1 - \frac{1-\epsilon}{c + an}. \tag{A.31}$$

Moreover, we have $b + o(1) \leq b + \epsilon$. Since f_n is nonnegative by assumption, a certain n_0 exists, such that, for all $n > n_0$:

$$f_n \leq f_{n-1}\left(1 - \frac{1-\epsilon}{c + an}\right) + b + \epsilon. \tag{A.32}$$

Introducing, for $m = 1, 2, \dots$, the definition

$$\eta_m = 1 - \frac{1-\epsilon}{c + a(n_0 + m)} = 1 - \frac{1}{\frac{c+an_0}{1-\epsilon} + \frac{a}{1-\epsilon}\, m}. \tag{A.33}$$

from (A.32) we get, by recursion:

$$f_{n_0+m} \leq f_{n_0} \prod_{\ell=1}^{m} \eta_\ell + (b+\epsilon)\left(1 + \sum_{k=2}^{m} \prod_{\ell=k}^{m} \eta_\ell\right). \tag{A.34}$$

In view of (A.33), Proposition 1 allows to conclude that:

$$\limsup_{n\to\infty} \frac{f_n}{n} \leq \frac{\frac{a}{1-\epsilon}(b+\epsilon)}{1 + \frac{a}{1-\epsilon}}, \tag{A.35}$$

and, hence, the claim in (A.29) follows from arbitrariness of ϵ. □

Proof of Theorem 1. First, we prove the claim for the synchronous scheduling, where all bots transmit regularly at intervals of constant duration $\tau = 1/\lambda$. Accordingly, we consider a *slotted* system with discrete time index $n \geq 0$, and introduce the quantities:

$$\mathscr{D}_n \triangleq \mathscr{D}_{\mathcal{B}}(n\tau),\ M_n \triangleq |\mathscr{D}_n|,\ \mathscr{E}_n \triangleq \mathscr{E}(n\tau),\ e_n \triangleq |\mathscr{E}_n|, \tag{A.36}$$

where we further observe that:

$$\lim_{n\to\infty} \frac{e_n}{n\tau} = \alpha \Rightarrow e_n = \alpha\tau\, n + o(n). \tag{A.37}$$

Now, for the synchronous case, it suffices to show that:

$$\frac{M_n}{n\tau} \xrightarrow{\mathrm{p}} \frac{\alpha B \lambda}{\alpha + B\lambda} \Leftrightarrow \frac{M_n}{n} \xrightarrow{\mathrm{p}} \frac{\alpha\tau\, B}{\alpha\tau + B} \triangleq \rho, \tag{A.38}$$

where B is the cardinality of subnet $\mathcal{B}$. Observe preliminarily that, by the orthogonality principle, we can write:

$$\mathbb{E}\left[\left(\frac{M_n}{n}-\rho\right)^2\right]=\mathbb{E}\left[\left(\frac{M_n-\bar{M}_n}{n}\right)^2\right]+\left(\frac{\bar{M}_n}{n}-\rho\right)^2, \quad (A.39)$$

and, since mean-square convergence implies convergence in probability [66], it suffices to show that, as $n \to \infty$, both terms on the RHS in (A.39) vanish.[1] We start by showing that $\bar{M}_n/n \to \rho$. At time n, the probability that k bots out of B pick a message outside $\mathscr{D}_{n-1}$ is (conditionally on M_{n-1}):

$$\binom{B}{k}\left(1-\frac{M_{n-1}}{e_n}\right)^k\left(\frac{M_{n-1}}{e_n}\right)^{B-k}. \quad (A.40)$$

Let us introduce the binomial random variable $\hat{X}_n$, with probability mass function given by (A.40), whose (conditional) expectation and variance are:

$$\mathbb{E}[\hat{X}_n|M_{n-1}]=B\left(1-\frac{M_{n-1}}{e_n}\right), \quad (A.41)$$

and

$$\mathrm{VAR}[\hat{X}_n|M_{n-1}]=B\left(1-\frac{M_{n-1}}{e_n}\right)\frac{M_{n-1}}{e_n}. \quad (A.42)$$

In order to build $\mathscr{D}_n$, we must select all the *distinct* messages among the k available ones. Ignoring repetitions, we can write:

$$M_n \le M_{n-1}+\hat{X}_n, \quad (A.43)$$

and, taking expectations:

$$\bar{M}_n \le \bar{M}_{n-1}\left(1-\frac{1}{\alpha\tau n/B+o(n)}\right)+B, \quad (A.44)$$

having used (A.41) and the expression of e_n appearing on the RHS

[1] In fact, we prove a stronger result in terms of *mean-square* convergence.

in (A.37). Direct application of Corollary 1 now yields:

$$\limsup_{n\to\infty} \frac{\bar{M}_n}{n} \leq \frac{\alpha\tau\, B}{\alpha\tau + B}. \tag{A.45}$$

Let us now prove the above (reversed) inequality for the lim inf. To this aim, we split $\mathscr{E}_n$ into C non-overlapping cells:

$$\mathscr{E}_n = \bigcup_{c=1}^{C} \mathscr{E}_{c,n}, \quad \left\lfloor \frac{|\mathscr{E}_n|}{C} \right\rfloor \leq |\mathscr{E}_{c,n}| \leq \left\lfloor \frac{|\mathscr{E}_n|}{C} \right\rfloor + 1, \tag{A.46}$$

where C is an arbitrary integer.

Since we focus on the regime where $n \to \infty$, it can be safely assumed that the initial number of words in the emulation dictionary obeys: $e_0 \geq C$. Let now:

$$\mathscr{D}_n = \bigcup_{c=1}^{C} \mathscr{D}_{c,n}, \quad M_{c,n} \triangleq |\mathscr{D}_{c,n}|, \quad M_n = \sum_{c=1}^{C} M_{c,n}, \tag{A.47}$$

and the events, for $j = 1, 2, \ldots, B$, and $c = 1, 2, \ldots, C$:

$$\mathcal{A}_{j,c} \triangleq \{\text{bot } j \text{ picks a message belonging to } \mathscr{E}_{c,n} \setminus \mathscr{D}_{c,n-1}\}. \tag{A.48}$$

Then we have, for any j:

$$\mathbb{P}[\mathcal{A}_{j,c} | M_{c,n-1}] = \frac{|\mathscr{E}_{c,n}| - M_{c,n-1}}{|\mathscr{E}_n|} \triangleq p_{c,n}, \tag{A.49}$$

with the dependence of $p_{c,n}$ upon $M_{c,n-1}$ being suppressed for ease of notation. From (A.46), we have:

$$\frac{1}{C} - \frac{1}{e_n} - \frac{M_{c,n-1}}{e_n} \leq p_{c,n} \leq \frac{1}{C} + \frac{1}{e_n} - \frac{M_{c,n-1}}{e_n}. \tag{A.50}$$

Now, $M_{c,n-1}$ increases by *at least* 1 whenever *at least* one bot picks a new message belonging to the c-th cell. This implies:

$$\mathbb{E}[M_{c,n} | M_{c,n-1}] \geq M_{c,n-1} + B p_{c,n} - (B p_{c,n})^2, \tag{A.51}$$

where we used the inequality $(1-p)^B \leq 1 - Bp + (Bp)^2$. On the other hand, for large n and small $\epsilon > 0$, from (A.50), we get $p_{c,n}^2 \leq (1/C + 1/e_n)^2 \leq C^{-2} + \epsilon$, and, hence, from (A.51):

$$\mathbb{E}[M_{c,n}|M_{c,n-1}] \geq M_{c,n-1} + Bp_{c,n} - \left(\frac{B}{C}\right)^2 - \epsilon', \tag{A.52}$$

for a certain small ϵ'. Conversely, using the lower bound in (A.50), and averaging over $M_{c,n-1}$, for large n we get:

$$\bar{M}_{c,n} \geq \bar{M}_{c,n-1}\left(1 - \frac{B}{e_n}\right) + \frac{B}{C}\left(1 - \frac{B}{C}\right) - \epsilon'', \tag{A.53}$$

for a certain small ϵ''. Summing over c, we get:

$$\begin{aligned} \bar{M}_n &\geq \bar{M}_{n-1}\left(1 - \frac{B}{e_n}\right) + \underbrace{B\left(1 - \frac{B}{C}\right) - C\,\epsilon''}_{b} \\ &= \bar{M}_{n-1}\left(1 - \frac{1}{\alpha\tau/B + o(n)}\right) + b, \end{aligned} \tag{A.54}$$

having used e_n in (A.37). Invoking now Corollary 1, we obtain:

$$\liminf_{n\to\infty} \frac{\bar{M}_n}{n} \geq \frac{\alpha\tau\, b}{\alpha\tau + B} \geq \frac{\alpha\tau\, B}{\alpha\tau + B}, \tag{A.55}$$

where the latter inequality follows from the definition of b, since C and ϵ are arbitrary. Equation (A.55), along with (A.45), yields that the second term on the RHS in (A.39) vanishes. Let us switch to the first term in (A.39). In view of the ascertained convergence of expectations, the variance will be proved to vanish if we show that: $\mathbb{E}[M_n^2]/n^2 \to \rho^2$. Now, in the light of (A.43), we can write: $\mathbb{E}[M_n^2|M_{n-1}] \leq M_{n-1}^2 + \mathbb{E}[\hat{X}_n^2|M_{n-1}] + 2\,M_{n-1}\mathbb{E}[\hat{X}_n|M_{n-1}]$, which, using (A.41) and (A.42), yields:

$$\begin{aligned} v_n &\leq v_{n-1}\frac{n-1}{n}\left[1-\frac{2B}{e_n}+\frac{B(B-1)}{e_n^2}\right] \\ &+ B\frac{\bar{M}_{n-1}}{n}\left(2-\frac{2B-1}{e_n}\right)+\frac{B^2}{n}, \end{aligned} \tag{A.56}$$

having also introduced the definition $v_n \triangleq \mathbb{E}[M_n^2]/n$. Now, the first term appearing on the RHS can be represented as

$$v_{n-1}\left(1-\frac{1}{\frac{\alpha\tau}{\alpha\tau+2B}n+o(n)}\right). \tag{A.57}$$

Likewise, the second term appearing on the RHS in (A.56) can be written as $2B\rho + o(1)$. Applying Corollary 1, we get:

$$\limsup_{n\to\infty}\frac{\mathbb{E}[M_n^2]}{n^2}=\limsup_{n\to\infty}\frac{v_n}{n}\leq 2B\rho\,\frac{\frac{\alpha\tau}{\alpha\tau+2B}}{1+\frac{\alpha\tau}{\alpha\tau+2B}}=\rho^2. \tag{A.58}$$

Now, subadditivity of limit superior implies:

$$\begin{aligned} &\limsup_{n\to\infty}\mathbb{E}\left[\left(\frac{M_n-\bar{M}_n}{n}\right)^2\right] \\ &\leq \limsup_{n\to\infty}\frac{\mathbb{E}[M_n^2]}{n^2}+\limsup_{n\to\infty}\left(-\frac{\bar{M}_n^2}{n^2}\right)\leq 0, \end{aligned} \tag{A.59}$$

with the latter inequality coming from (A.58), and from $\bar{M}_n/n \to \rho$. The claim for the synchronous case is so proved.

As regards the Poisson case, we consider again the slotted system in (A.36), but for the fact that τ is now an arbitrarily *small* interval. Let A denote the number of transmission attempts in a single slot, that is, a Poisson random variable with expectation $\bar{A}=\sum_{u\in\mathcal{B}}\lambda_u\tau=\lambda_{\mathcal{B}}\tau$. Since the A transmissions correspond to A independent choices of messages from the emulation dictionary, for small τ the system behaves as if we had A synchronous bots, where A is now *random*. Thus, the proof

$$\hat{\mathcal{B}}_1 = \{1\}, \quad \mathcal{E}_2 = \left\{\hat{\rho}_{\hat{\mathcal{B}}_1 \cup \{2\}} < \gamma(\hat{\mathcal{B}}_1, \{2\})\right\}, \ldots\ \mathcal{E}_B = \left\{\hat{\rho}_{\hat{\mathcal{B}}_{B-1} \cup \{B\}} < \gamma(\hat{\mathcal{B}}_{B-1}, \{B\})\right\},$$

$$\mathcal{E}_{B+1} = \left\{\hat{\rho}_{\hat{\mathcal{B}}_B \cap \{B+1\}} \geq \gamma(\hat{\mathcal{B}}_B, \{B+1\})\right\}, \ \ldots\ \mathcal{E}_N = \left\{\hat{\rho}_{\hat{\mathcal{B}}_B \cap \{N\}} \geq \gamma(\hat{\mathcal{B}}_B, \{N\})\right\}. \tag{A.62}$$

for the Poisson case boils down to modify slightly the previous proof in order to take into account such additional randomness. Specifically, Eq. (A.44) should be modified by considering a random number of bots A, and then taking expectations, yielding:[2] $\bar{M}_n \leq \bar{M}_{n-1}(1 - \bar{A}/e_n) + \bar{A}$. Likewise, Eq. (A.51) becomes: $\mathbb{E}[M_{c,n}|M_{c,n-1}] \geq M_{c,n-1} + 1 - \mathbb{E}[(1-p_{c,n})^A|M_{n-1}]$. Since, for the Poisson random variable A, it is easy to show that $\mathbb{E}[(1-p)^A] = e^{-\bar{A}p} \leq 1 - \bar{A}p + (\bar{A}p)^2$, the conclusion in (A.38) still holds true, with B simply replaced by $\bar{A}$. Finally, the inequality in (A.56) becomes:

$$\begin{aligned} v_n &\leq v_{n-1}\frac{n-1}{n}\left[1 - \frac{2\bar{A}}{e_n} + \frac{\overline{A(A-1)}}{e_n^2}\right] \\ &+ \frac{\bar{M}_{n-1}}{n}\left(2\bar{A} - \frac{\overline{A(2A-1)}}{e_n}\right) + \overline{A^2}. \end{aligned} \tag{A.60}$$

Having shown that all the equations used to prove the pertinent convergence hold true with B replaced by $\bar{A}$, we conclude that: $\frac{M_n}{n} \xrightarrow{\text{p}} \frac{\alpha\tau\bar{A}}{\alpha\tau+\bar{A}} = \frac{\alpha\lambda_{\mathcal{B}}}{\alpha+\lambda_{\mathcal{B}}}$. □

A.3 Appendix 3

Proof of Theorem 2. Let us focus on a single step of the BotBuster loop, i.e., the algorithm behavior for a fixed b_0. Consider first the case that b_0 is a normal user, and introduce, for $j \in \mathcal{N} \setminus \{b_0\}$, the events: $\mathcal{E}_j = \{\hat{\rho}_{\{b_0\}\cup\{j\}} \geq \gamma(\{b_0\}, \{j\})\}$. Eq. (4.29) reveals that,

[2]We implicitly use: *i*) the independence between scheduling policy and message picking, and *ii*) the memoryless property of the Poisson process.

for any j, $\mathbb{P}[\mathcal{E}_j] \to 1$ as $t \to \infty$. But we also have that, for b_0 normal,

$$\mathbb{P}[\text{inner loop outputs } \hat{\mathcal{B}} = \{b_0\}] = \mathbb{P}[\cap_{j \in \mathcal{N} \setminus \{b_0\}} \mathcal{E}_j] \to 1, \quad \text{(A.61)}$$

where the convergence follows by the fact that each of the events has probability converging to one as $t \to \infty$.

In contrast, if b_0 is a bot, we distinguish two cases: *i*) if j is normal, from (4.29) we conclude that $\hat{\rho}_{\{b_0\}\cup\{j\}} \geq \gamma(\{b_0\}, \{j\})$ with probability converging to one as $t \to \infty$, while *ii*) if j is a bot, from (4.28) we conclude that $\hat{\rho}_{\{b_0\}\cup\{j\}} < \gamma(\{b_0\}, \{j\})$ with probability converging to one as $t \to \infty$. Assume now, without loss of generality, that the first B users are bots, that $b_0 = 1$, and that the remaining users are normal. In (A.62), we introduce the events corresponding to the inner loop over index j, as well as the associated botnet estimates at step j, denoted by $\hat{\mathcal{B}}_j$. After noticing that, in the definition of these events, the inequality signs in the threshold comparisons are different for $j \leq B$ and for $j > B$, it is seen that the event $\hat{\mathcal{B}} = \{1, 2, \ldots, B\}$ corresponds to the event $\cap_{j=2}^{N} \mathcal{E}_j$. Since, in view of the above points *i*) and *ii*), we have $\mathbb{P}[\mathcal{E}_j] \to 1$, we conclude that (if $b_0 = 1$ is a bot):

$$\mathbb{P}[\text{inner loop outputs } \hat{\mathcal{B}} = \{1, 2, \ldots, B\}] = \mathbb{P}[\cap_{j=2}^{N} \mathcal{E}_j] \to 1, \quad \text{(A.63)}$$

which implies the validity of (5.8). □

Bibliography

[1] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. DDoS in the IoT: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[2] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti. A survey on the security of stateful SDN data planes. *IEEE Communications Surveys Tutorials*, 19(3):1701–1725, 2017.

[3] S. Lal, T. Taleb, and A. Dutta. NFV: Security threats and best practices. *IEEE Communications Magazine*, 55(8):211–217, 2017.

[4] A. D. Keromytis. A comprehensive survey of Voice over IP security research. *IEEE Communications Surveys Tutorials*, 14(2):514–537, 2012.

[5] M. Di Mauro and M. Longo. Skype traffic detection: A decision theory based tool. In *2014 International Carnahan Conference on Security Technology (ICCST)*, pages 1–6, 2014.

[6] M. Di Mauro and C. Di Sarno. A framework for internet data real-time processing: A machine-learning approach. In *2014 International Carnahan Conference on Security Technology (ICCST)*, pages 1–6, 2014.

[7] M. Di Mauro and M. Longo. Revealing encrypted WebRTC traffic via machine learning tools. In *2015 12th Interna-*

tional Joint Conference on e-Business and Telecommunications (ICETE), pages 259–266, 2015.

[8] M. Di Mauro and M. Longo. A decision theory based tool for detection of encrypted WebRTC traffic. In *2015 18th International Conference on Intelligence in Next Generation Networks*, pages 89–94, 2015.

[9] M. Di Mauro and C. Di Sarno. Improving SIEM capabilities through an enhanced probe for encrypted Skype traffic detection. *Journal of Information Security and Applications*, 38(PP):85–95, 2018.

[10] V. Karyotis and M.H.R. Khouzani. *Malware Diffusion Models for Modern Complex Networks: Theory and Applications.* Morgan Kaufmann, 2016.

[11] C.C. Zou, W. Gong, and D. Towsley. Code Red worm propagation modeling and analysis. In *Proc. of the 9th ACM Conference on Computer and Communications Security*, pages 138–147, 2002.

[12] S. Shin, G. Gu, N. Reddy, and C. P. Lee. A large-scale empirical study of Conficker. *IEEE Transactions on Information Forensics and Security*, 7(2):676–690, 2012.

[13] Worm infects millions of computers worldwide. `http://www.nytimes.com/2009/01/23/technology/internet/23worm.html?_r=1&em`.

[14] N.T.J. Bailey. *The elements of Stochastic Processes with applications to the natural sciences.* John Wiley and Sons, New York, 1964.

[15] W.O. Kermack and A.G. McKendrick. A contribution to the mathematical theory of epidemics. *Proc. of Royal Society of London, Series A*, 115(772):700–721, 1927.

[16] P.J. Costa. *Applied Mathematics for the Analysis of Biomedical Data.* John Wiley and Sons, New York, 2017.

[17] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita. Botnet in DDoS attacks: Trends and challenges. *IEEE Communications Surveys Tutorials*, 17(4):2242–2270, 2015.

[18] S. Wen, W. Zhou, J. Zhang, Y. Xiang, W. Zhou, W. Jia, and C. C. Zou. Modeling and analysis on the propagation dynamics of modern email malware. *IEEE Transactions on Dependable and Secure Computing*, 11(4):361–374, 2014.

[19] V. Karyotis and S. Papavassiliou. Macroscopic malware propagation dynamics for complex networks with churn. *IEEE Communications Letters*, 19(4):577–580, 2015.

[20] S. Eshghi, M. H. R. Khouzani, S. Sarkar, and S. S. Venkatesh. Optimal patching in clustered malware epidemics. *IEEE/ACM Transactions on Networking*, 24(1):283–298, 2016.

[21] T. Wang, C. Xia, Z. Li, X. Liu, and Y. Xiang. The spatial-temporal perspective: The study of the propagation of modern social worms. *IEEE Transactions on Information Forensics and Security*, 12(11):2558–2573, 2017.

[22] S. H. Sellke, N.B. Shroff, and S. Bagchi. Modeling and automated containment of worms. *IEEE Transactions on Dependable and Secure Computing*, 5(2):71–86, 2008.

[23] T.E. Harris. *The Theory of Branching Processes.* Springer-Verlag, 1967.

[24] IA. Farina, C. Fantacci, and M. Frasca. Stochastic filtering of a random Fibonacci sequence: Theory and applications. *Signal Processing*, 104:212–224, 2014.

[25] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical approaches to DDoS attack detection and response. In *Proc. DARPA Information Survivability Conference and Exposition*, pages 303–314 vol.1, 2003.

[26] 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*, 6(2):426–437, 2011.

[27] 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*, 9(7):1069–1083, 2014.

[28] Global DDoS threat landscape. `https://www.incapsula.com/blog/ddos-global-threat-landscape-report-q2-2015.html`.

[29] Layer 7 DDoS. `http://blog.sucuri.net/2014/02/layer-7-ddos-blocking-http-flood-attacks.html`.

[30] ETSI. Network Functions Virtualisation (NFV) reliability; report on models and features for end-to-end reliability. Technical report, 2016.

[31] Y. Yamato, Y. Nishizawa, S. Nagao, and K. Sato. Fast and reliable restoration method of virtual resources on OpenStack. *IEEE Transactions on Cloud Computing (In Press)*, 2015.

[32] H. Khazaei, J. Miic, V. B. Miic, and N. B. Mohammadi. Availability analysis of cloud computing centers. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 1957–1962, 2012.

[33] X. Zhang, C. Lin, and X. Kong. Model-driven dependability analysis of virtualization systems. In *2009 Eighth IEEE/ACIS International Conference on Computer and Information Science*, pages 199–204, 2009.

[34] J. Dantas, R. Matos, J. Araujo, and P. Maciel. An availability model for eucalyptus platform: An analysis of warm-standby replication mechanism. In *2012 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1664–1669, 2012.

[35] R. Ghosh, F. Longo, F. Frattini, S. Russo, and K. S. Trivedi. Scalable analytics for iaas cloud availability. *IEEE Transactions on Cloud Computing*, 2(1):57–70, 2014.

[36] D. S. Kim, J. B. Hong, T. A. Nguyen, F. Machida, J. S. Park, and K. S. Trivedi. Availability modeling and analysis of a virtualized system using Stochastic Reward Nets. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*, pages 210–218, 2016.

[37] E. Ataie, R. Entezari-Maleki, L. Rashidi, K. S. Trivedi, D. Ardagna, and A. Movaghar. Hierarchical stochastic models for performance, availability, and power consumption analysis of IaaS clouds. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2017.

[38] V. Matta, M. Di Mauro, M. Longo, and A. Farina. Cyber-threat mitigation exploiting the birth-death-immigration model (under review). *IEEE Transactions on Information Forensics and Security.*

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

[40] 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*, 12(8):1844–1859, 2017.

[41] V. Matta, M. Di Mauro, and M. Longo. Botnet identification in multi-clustered DDoS attacks. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2171–2175, 2017.

[42] G. Carullo, M. Di Mauro, M. Galderisi, M. Longo, F. Postiglione, and M. Tambasco. *Object Storage in Cloud Computing Environments: An Availability Analysis*, pages 178–190. Springer International Publishing, 2017.

[43] M. Di Mauro, M. Longo, and F. Postiglione. *Reliability analysis of the controller architecture in Software Defined Networks*, pages 1503–1510. Talyor and Francis Group, 2015.

[44] M. Di Mauro, M. Longo, F. Postiglione, R. Restaino, and M. Tambasco. *Availability Evaluation of the virtualized infrastructure manager in network function virtualization environments*, pages 2591–2596. Talyor and Francis Group, 2016.

[45] M. Di Mauro, M. Longo, F. Postiglione, G. Carullo, and M. Tambasco. Service function chaining deployed in an NFV environment: An availability modeling. In *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 42–47, 2017.

[46] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco. *Availability evaluation of a virtualized IP Multimedia Subsystem for 5G network architectures*, pages 2203–2210. Talyor and Francis Group, 2017.

[47] M. Di Mauro, M. Longo, F. Postiglione, G. Carullo, and M. Tambasco. Software defined storage: Availability modeling and sensitivity analysis. In *2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–7, 2017.

[48] M. Di Mauro, M. Longo, F. Postiglione, and M. Tambasco. *Availability Modeling and Evaluation of a Network Service Deployed via NFV*, pages 31–44. Springer International Publishing, 2017.

[49] M. Di Mauro M. Longo and F. Postiglione. Performability evaluation of Software Defined Networking infrastructures. In *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*, pages 88–95, 2017.

[50] D. Kendall. On the generalized birth-and-beath process. 19, 11 1948.

[51] I.A. Ushakov. A universal generating function. *Sov. J. Comput. Syst. Sci.*, 24(5):37–49, 1986.

[52] G. Levitin. *The Universal Generating Function in Reliability Analysis and Optimization.* Springer Publishing Company, Incorporated, 2010.

[53] D. Pollard. *Convergence of Stochastic Processes.* Springer-Verlag, New York, 1984.

[54] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2):93–122, 1984.

[55] R.A. Howard. *Dynamic Probabilistic Systems, Vol II: Semi-Markov and Decision Processes.* John Wiley and Sons, New York, 1971.

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

[57] L. Kleinrock. *Queueing Systems. Volume I: Theory.* John Wiley and Sons, New York, 1975.

[58] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the internet in your spare time. In *Proc. USENIX Security Symposium*, pages 303–314 vol.1, 2003.

[59] S. Yu, G. Gu, A. Barnawi, S. Guo, and I. Stojmenovic. Malware propagation in large-scale networks. *IEEE Transactions on Knowledge and data engineering*, 27(1):170–179, 2015.

[60] R.N. Nucho. Transient behavior of the Kendall birth-death process. Applications to capacity expansion for special services. *The Bell System Technical Journal*, 60(1):57–87, 1981.

[61] N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions, Volume 1.* John Wiley and Sons, Inc., 1994.

[62] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions.* Courier Corporation, 1964.

[63] P. Jagers. Convergence of general branching processes and functionals thereof. *Journal of applied probability*, 11(3):471–478, 1974.

[64] P. Olofsson96. General branching processes with immigration. *Journal of applied probability*, 33(4):940–948, 1996.

[65] T.R. Maltus. *An Essay on the principle of population.* J. Johnson in St. Paul's Church-yard, London, 1798.

[66] H. Shao. *Mathematical Statistics.* Springer-Verlag, New York, 2003.

[67] A. Gut. *Probability: a Graduate course, 2nd ed.* Springer, New York, 2013.

[68] F.W. Crawford. Estimation for general birth-death process. *Journal of the American Statistical Association*, 109(506):730–747, 2014.

[69] N. Keiding. Estimation in the birth process. *Biometrika*, 61(1):71–80, 1974.

[70] S. Marano, V. Matta, and L. Tong. Distributed detection in the presence of Byzantine attacks. *IEEE Transactions on Signal Processing*, 57(1):16–29, 2009.

[71] M. Barni and B. Tondi. The source identification game: An information-theoretic perspective. *IEEE Transactions on Information Forensics and Security*, 8(3):450–463, 2013.

[72] M. Mardani and G.B. Giannakis. Estimating traffic and anomaly maps via network tomography. *IEEE/ACM Transactions on Networking*, 24(3):1533–1547, 2016.

[73] S. Ross. *Stochastic Processes.* John Wiley and Sons, Inc., 1996.

[74] I.F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou. A roadmap for traffic engineering in Software Defined Networks. *Computer Networks*, 71:1–30, 2014.

[75] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, 2008.

[76] 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*, 227(3):290–301, 2013.

[77] ETSI. Network Functions Virtualisation: An introduction, benefits, enablers, challenges and call for action. Technical report, 2012.

[78] R. De S. Matos, P. Maciel, F. Machida, K. Dong Seong, and K.S. Trivedi. Sensitivity analysis of server virtualized system availability. *IEEE Transactions on Reliability*, 61(4):994–1006, 2012.

[79] R.A. Sahner and K.S. Trivedi. Reliability modeling using SHARPE. *IEEE Transactions on Reliability*, 36(2):186–193, 1987.

[80] J.H. Curtiss. A note on the theory of Moment Generating Functions. 13(4):430–433, 12 1942.