A Latency-driven Availability Assessment for Multi-Tenant Service Chains

Luigi De Simone, *Member, IEEE,* Mario Di Mauro, *Senior Member, IEEE,* Roberto Natella, Fabio Postiglione

Abstract—Nowadays, most telecommunication services adhere to the Service Function Chain (SFC) paradigm, where network functions are implemented via software. In particular, container virtualization is becoming a popular approach to deploy network functions and to enable resource slicing among several tenants. The resulting infrastructure is a complex system composed by a huge amount of containers implementing different SFC functionalities, along with different tenants sharing the same chain. The complexity of such a scenario lead us to evaluate two critical metrics: the steady-state availability (the probability that a system is functioning in long runs) and the latency (the time between a service request and the pertinent response). Consequently, we propose a latency-driven availability assessment for multi-tenant service chains implemented via Containerized Network Functions (CNFs). We adopt a multi-state system to model single CNFs and the queueing formalism to characterize the service latency. To efficiently compute the availability, we develop a modified version of the Multidimensional Universal Generating Function (MUGF) technique. Finally, we solve an optimization problem to minimize the SFC cost under an availability constraint. As a relevant example of SFC, we consider a containerized version of IP Multimedia Subsystem, whose parameters have been estimated through fault injection techniques and load tests.

Index Terms—Availability; Reliability; Queueing Model; Container Virtualization; IP Multimedia Subsystem; Redundancy Optimization; Multi-State Systems; Universal Generating Function; Network Function Virtualization.

1 Introduction

Today, service providers conceive modern network infrastructures by taking into account cloud-centric paradigms such as Network Function Virtualization (NFV), which remodels classic network nodes (routers, switches, firewalls, and others) as virtual entities called Virtualized Network Functions (VNFs). VNFs can be chained to realize Service Function Chains (SFCs), which represent the modern way of composing and providing new services quickly and flexibly, especially when coupled with the Software Defined Networking [1]–[3]. Many applications adopt the SFC paradigm [4], [5] with some examples shown in Fig. 1: the Data Center domain (upper panel), where the chain is made of systems such as Intrusion Detection, Firewall, and Router in charge of processing the data flow between a Server and the Internet; the cellular domain (middle panel), where the mobile traffic is managed by a chain including: enhanced Node-B (e-NB) to handle the radio link, and Signal and Packet Gateways (S-GW, P-GW) to manage the signaling and the data content, respectively; IP Multimedia Subsystem (IMS) (lower panel) which relies on a chain of network nodes providing multimedia services. Across such domains, virtualization enables a flexible and efficient resource utilization, since resources can be allocated and shared among several service providers (*tenants*) at a fine grain. Examples of multi-tenant commercial and standard-based solutions include: softwarized chains in the Evolved Packet Core domain [6], software-based IMS shared among different providers [7], and the Gateway Core Network (GWCN) for infrastructure sharing among different providers [8]. All the aforementioned systems must satisfy quality-of-service requirements, both in terms of steady-state availability (the probability that a system is functioning in long runs, i.e., when stationary conditions are reached) and latency (the time between a service request and the pertinent response).

From a technological point of view, we are witnessing the adoption of *Containarized Network Functions* (CNFs) to implement VNFs [9], [10]. Differently from traditional virtualization technology, containers are a lightweight solution, as they do not emulate a full computer machine, and do not run a dedicated operating system. Moreover, containers can be quickly deployed and orchestrated using dedicated management platforms (e.g., Docker [11]). Remarkably, lightweight containers allow designers to achieve a great flexibility, in terms of a fine-grained allocation of resources among various tenants. On the other hand, the complexity of managing a huge number of container replicas could negatively affect the computational cost of many availability techniques. In addition, exploiting containerized solutions in real-time environments requires a particular attention to achieve the low latency objectives. It is the case of IMS, whose latency must be below few tens of milliseconds [12]–[15]. Therefore, there is a need for new assessment techniques that are computationally efficient, and that can address both high availability and low latency constraints.

Aimed at dealing with the aforementioned concerns, in this work we advance:

- *M. Di Mauro and F. Postiglione are with the Department of Information and Electrical Engineering and Applied Mathematics (DIEM), University of Salerno, Italy, (E-mail: {mdimauro,fpostiglione}@unisa.it).*
- *L. De Simone and R. Natella are with the Department of Electrical Engineering and Information Technologies (DIETI), University of Napoli Federico II, Italy, (E-mail: {luigi.desimone,roberto.natella}@unina.it).*

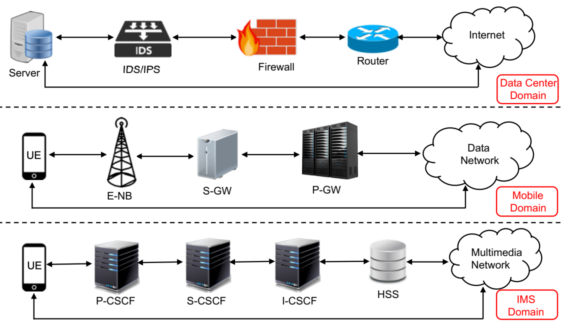


Fig. 1. Examples of domains embracing the Service Chains paradigm: Data Center (upper panel), Mobile Network (middle panel), IP Multimedia Subsystem (lower panel).

1) A novel latency-driven modeling approach for a Containerized Network Function (CNF), a three-layered (Software, Docker, Infrastructure layers) structure representing the elementary block of a service chain, and which has been modeled in terms of a Multi-State System (MSS). The MSS formalism is helpful to encode the interplay among the various nested layers in a containerized node, in terms of failures and repair actions. To take into account latency, which is typically neglected in the technical literature, we enhance this representation with a *delay model*, based on queues with non-exponential service times and time-varying serving facilities ruled by the failure/repair process.

2) A technique for the efficient analysis of service chains composed by several interconnected CNFs, based on the *Universal Generating Function* (UGF) technique. This paper extends our previous work on a multidimensional version of UGF (MUGF) [17] that supports multi-tenant service chains, where different operators (or tenants) share the same infrastructure. In this work, we revised the MUGF technique to take into account the novel latency-based metric, in order to support the analysis over a chain of CNFs.

3) A detailed case study on a containerized multi-tenant IP Multimedia Subsystem (cIMS) platform, a service chain-like infrastructure crucial to manage multimedia content within the 5G core network. The case study shows the feasibility of the proposed approach in the context of a relevant use case. In particular, it presents an extensive set of experiments on the *Clearwater* cloud-based IMS platform [18], through i) load test experiments, to estimate the empirical service times, and ii) fault injection experiments, to estimate repair times. Remarkably, fault injection techniques turned out to be useful when empirical data are lacking, and revealed that time-to-recovery is much longer than what is typically assumed by most model-based studies.

The rest of the paper is organized as follows: Section 2 discusses relevant work on availability assessment in cloud environments. In Section 3 we provide an overview of the IMS case study, with a discussion about its containerized deployment. In Section 4 we present the availability and queueing models of a CNF, being the elementary structure of a service chain. In Section 5 we address the chain availability concern through the MUGF technique and the related optimization problem. Section 6 presents the testbed and offers details about the experimental trials. Section 7 concludes the paper. For the sake of readability, Table 1 summarizes the notation adopted across the paper.

TABLE 1
Notation

m	(Containerized) IMS tier (P,S,I,H)
ℓ	CNF redundancy index
CNF$^{(m,\ell)}$	Parallel CNF ℓ associated to tier m
i; k	Tenant i; number of tenants using the cIMS
η_i	Number of working containerized instances controlled by tenant i
$\boldsymbol{\eta}$	CNF State vector
$N^{(m,\ell)}$	Number of states of CTMC performance model of CNF ℓ of tier m
$g_{i,\boldsymbol{\eta}}$	Capacity level exposed by CNF for tenant i in state $\boldsymbol{\eta}$
γ	Serving capacity
$\boldsymbol{g}_{\boldsymbol{\eta}}$	Capacity level vector in state $\boldsymbol{\eta}$
$\boldsymbol{\delta}_{\boldsymbol{\eta}}$	Mean delay performance vector in state $\boldsymbol{\eta}$
$p_{\boldsymbol{\eta}}$	Steady-state probability of being in state $\boldsymbol{\eta}$
$\boldsymbol{\Delta}(t)$	Vector stochastic process including all tenants mean delays per CNF
$\boldsymbol{W}^c(t)$	Maximum tolerated value for the mean CSD
$A^c(\boldsymbol{w}^c)$	Steady-state availability of the cIMS
$\boldsymbol{J}^c$	Number of states of the cIMS
$u^{(m)}(\boldsymbol{z}); u^c(\boldsymbol{z})$	MUGF for tier m; MUGF for the cIMS
$\Omega_S, \Omega_D, \Omega_I$	State spaces of software, docker, infrastructure layers
$\lambda_C, \lambda_D, \lambda_I$	Failure rate of containerized instances, docker, and infrastructure layers
μ_C, μ_D, μ_I	Repair rate of containerized instances, docker, and infrastructure layers
α_i	Arrival rate of requests at tenant i
β_m	Service rate of requests at tier m

2 Related Work

Availability issues represent a hot topic when dealing with *softwarized* networks, where the presence of virtualized entities (e.g. hypervisors, containerized environments, etc.) pose new intriguing challenges for telecom operators that are called for adhering to strict Service Level Agreements (SLAs) [19]–[22]. Due to the vastness of the topic, technical studies adopt different angles to face the availability problems, including: designing available virtualized infrastructures to manage traffic problems [23], [24], optimizing the allocation of virtualized infrastructures to maximize the resiliency [25], [26], optimizing the availability scheduling of virtual resources [27], [28], managing the state of virtualized services in resiliency problems [29], [30]. On the other hand, in our work we mainly focus on a modeling methodology for the availability issues in softwarized networks. Thus, we are going to explore some affine literature more in detail, by highlighting the main differences with our work.

Fan et al. [31] faced an availability problem concerning the optimal deployment of an SFC infrastructure. In particular, their aim is to find the minimum number of backup VNFs that guarantees a desired availability level. A similar problem is tackled by Kong et al. [32], where a heuristic algorithm has been conceived to maximize the availability of an SFC through an optimal distribution of backup VNFs across primary and backup paths. Alameddine et al. [33] focused on virtual machine redundancy in a multi-tenant environment by adopting an optimal primary/backup logic. All of these works did not consider, or only partially considered, a failure/repair model that is instead accurately examined in our availability setting.
Other studies focused on more compact formalisms to model network availability aspects. It is the case of Sebastio

et al. [34], where an availability assessment of exemplary containerized architectures is faced through the Stochastic Reward Networks (SRNs) framework. An SRN-based approach has been profitably adopted also by Bruneo [35], where a stochastic model to typify some aspects of an Infrastructure-as-a-Service framework has been considered. Similarly, a technique relying on Stochastic Petri Networks (SPNs) has been exploited by Sousa et al. [36], where an availability analysis of cloud-based deployments has been carried out. Interestingly, both SRN/SPN and the proposed MUGF rely on a common underlying Markov model. While MUGF prefers an open analytical approach, SRN/SPN offers a more compact representation (through the formalism of places, arcs, tokens) that can be more convenient for some users. However, in the case of multi-tier systems such as SFCs, as further discussed in this paper, having the underlying Markov model hidden by the SPN/SRN hinders the computation of the availability.

Another track of works exploits the UGF methodology, which, although historically adopted to cope with availability issues in the field of industrial systems, has found fertile ground in networking management. Some examples include the work by Sun et al. [37], where a UGF-based technique is assessed for modeling physical and virtual system failures, and Yu et al. [38], where the UGF has been applied in the field of service requests in cloud scenarios. Both studies focused on single-tenant environments, not calling for the application of a multidimensional form of UGF.

In the present paper, by exploiting the properties of the multidimensional UGF (MUGF) at first conceived by our previous work [17], we propose a new availability assessment method for multi-tenant environments. A set of clear novelties emerge with respect to the original proposal. First, in this work we consider a containerized infrastructure (in place of a traditional virtualized architecture considered in previous work [17]), which is reflected in the three-layered structure of our model, where the containers do not embed an OS and are deployed on top of the Docker container manager. This model directly translates into a novel experimental testbed (missing in the previous work [17]), based on the Clearwater project, a realistic cIMS deployment which allows to estimate the value of key quantities such as repair rates, call setup latencies, and mean service times. Moreover, in this work each tenant is modeled in accordance to a sophisticated queueing model (there was no queueing model in the previous work [17]), which allows to analyze the *call setup delay* (CSD), a critical latency metric for 5G networks as specified by ETSI [12]. Finally, the MUGF structure (in particular, series and parallel operators) is totally different from the one introduced in previous work due to the different metric adopted: the number of sessions in [17], and the CSD in this new proposal.

We remark that, from a scalability viewpoint, the MUGF technique exhibits interesting results compared to other methods in similar fields. For example, Petri net structures (e.g., SPN/SRN), beyond requiring very specific tools to be solved, can suffer when dealing with large and nested systems, as also highlighted by Peterson [39] and Herzog [40]. In particular, SRNs offer a system state representation in terms of the "token" distributions (a.k.a. markings), where each marking is representative of a particular state of the system at a given time t. On one hand, this approach provides a comfortable interface to automatically specify the token distribution by hiding technical details about the underlying state model. On the other hand, such an approach does not allow to easily retrieve the MSS state distribution which is needed by the analytical formulation of the MUGF.

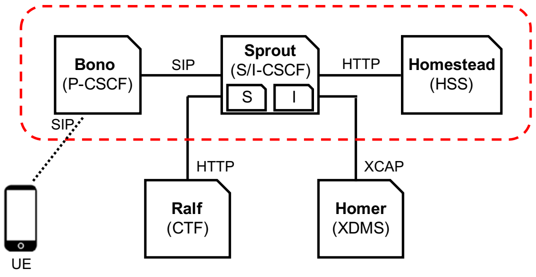


Fig. 2. Overview of the Clearwater IMS.

Yet, the classic Continuous-Time Markov Chain (CTMC) representation would lead to a space state explosion, since it requires the entire cIMS to be modeled (monolithic approach), whereas the MUGF uses a combination of performance distributions of single nodes to achieve the cIMS performance distribution (decomposition approach).

3 Containerized IMS case study

In this study we present the proposed approach in the context of a cIMS case study, based on Clearwater [18], a real IMS product that fully leverages containers and cloud computing technology. The cIMS consists of a chain of *softwarized* functions running within containers. Such containers are managed by a container engine (in our case Docker) which is installed on a physical machine (a *node*). In turn, the nodes can be replicated to form a *tier*, in order to achieve higher performance and availability. Figure 2 shows a sketch of the Clearwater IMS characterized by the following functions: *Bono*, which is the P-CSCF (Proxy-Call Session Control Function), and acts as anchor point for clients relying on the Session Initiation Protocol (SIP); *Sprout* simultaneously acts as S-CSCF (Serving-CSCF) in charge of managing SIP registrations, and as I-CSCF (Interrogating-CSCF) for handling associations between UEs and a specific S-CSCF; *Homestead* represents the HSS (Home Subscriber Server) for users authentication; *Ralf* acts as CTF (Charging Trigger Function), for charging and billing operations; *Homer* manages service setting documents per user, acting as XML Document Management Server (XDMS). A red dashed rectangle in Fig. 2 indicates the mandatory functions of the cIMS architecture that we consider in our analysis.

The IMS is called to satisfy real-time constraints such as delay, jitter, packet loss. In this regard, a metric called Call Setup Delay (CSD) has been elected as a critical Key Performance Indicator (KPI) [13]–[15], being strongly related to the end-user experience. Formally, CSD is a time-based metric defined as the time interval between Invite message sent from the caller and the received Ringing message (code 180) [16], and well approximates the average time that user requests spend in the cIMS (due to the processing time needed by each node to handle the requests), where the propagation

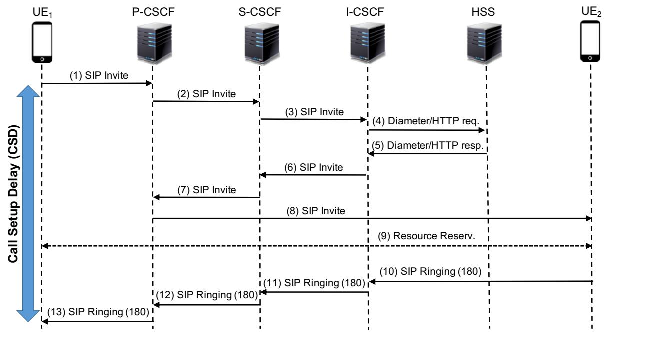


Fig. 3. IMS (single domain) call setup, where the Call Setup Delay metric is represented on the left.

delays are neglected. Figure 3 shows a simplified scenario of a SIP call flow (IMS single domain), where the CSD is accordingly represented as a vertical double arrow between the initial sent message, (1) SIP Invite, and the last received message, (13) SIP Ringing.

3.1 Containerized Network Function Model

The network functions of a service chain can be deployed in dedicated containers, and decoupled from the underlying infrastructure, according to a three-layer structure:

- *Software layer*: Its role is to run application software that implements the business logic of the service chain, to be deployed as containers (for example, in our testbed, the *Bono* and the *Sprout* applications). We assume that a CNF hosts containers of the same type;
- *Docker layer*: Its role is to provide a run-time support (e.g., a service daemon) to build, run, and manage OS containers; this layer is also exploited in other container management technologies, such as *Linux Container Daemon* and *Rocket*;
- *Infrastructure layer*: It represents the underlying physical layer that, for the sake of simplicity, includes only the operating system (OS) and the hardware (HW) components (e.g., CPU, RAM, etc.).

A single CNF represents the elementary block of each tier of the service chain (i.e., a node in the system that includes the three layers). As it will be clear in the following, a cIMS tier can be made of several redundant CNFs (typically, of one and the same type), in order to increase capacity and to meet performance and availability objectives. A stylized representation of a CNF is depicted in Fig. 4, where the i-th tenant ($i = 1, \dots, K$) manages η_i containerized instances. For each tenant, a containerized instance can manage, in turn, a number of service requests amounting to γ (*serving capacity*), at the same time. In practice, a containerized instance is supposed to be composed of processes, each one handling a single request. The serving capacity represents the number of requests handled by each tenant. The resulting queueing model of a single CNF for tenant i is an infinite queue $M/G/\gamma\eta_i$ as in Fig. 4, where $\gamma\eta_i$ depends on the actual working conditions of CNF (see the forthcoming Sect. 4.2 for more details). Moreover, all tenants share the Docker and Infrastructure layers, as typical of modern cloud

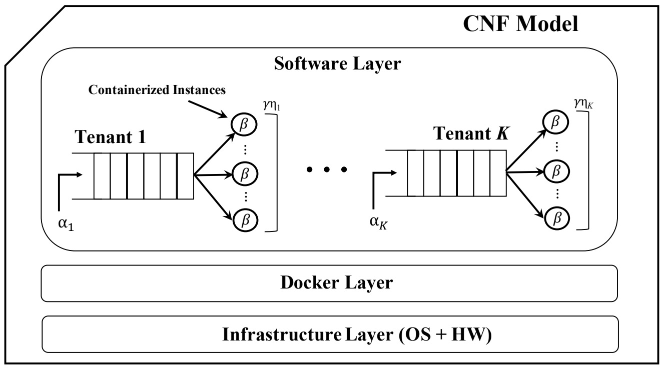


Fig. 4. A CNF hosting K tenants. Each tenant can be represented through a $M/G/\gamma\eta_i$ queueing model managing a set of containerized instances in the Software layer. Quantities α_i and β are the arrival and service rates, respectively. The Docker layer manages containers. The Infrastructure layer embodies the host OS and the hardware.

deployments. Note that Fig. 4 represents the system from a conceptual perspective, as in its current implementations the load among container instances is balanced by communication mechanisms (such as message queues, or DNS-based load balancing) managed by the underlying Docker and infrastructure layers.

4 CNF Modeling

In this section, we analyze the behavior of a single CNF according to a double perspective. The first one concerns the availability characterization of a CNF in terms of a multi-state model, where the failure/repair behaviors of the three layers (Software, Docker, and Infrastructure) are represented and evaluated (Sect. 4.1). The second one pertains to the latency characterization by means of CNF queueing modeling (Sect. 4.2). The CNF availability model and the CNF queueing model can both be applied on a given CNF configuration, in order to get an assessment of its availability and latency. These results will be used in the next section for capacity planning of the service chain.

4.1 CNF availability model

The stochastic behavior (in terms of failure and repair actions) of the three CNF layers can be captured through the concept of *state*. A state reflects a particular condition (e.g. up/down) of: one or more containerized instances (belonging to the Software layer), the Docker layer, the Infrastructure layer. Accordingly, the CNF model of Fig. 4 can be *translated* in terms of the transition-state diagram reported in Fig. 5, where a state is labeled with a K-dimensional vector $\boldsymbol{\eta} = (\eta_1, ..., \eta_K) \in \prod_{i=1}^{K}\{0, \dots, n_i\}$ and $\eta_i \in \{0, 1, ..., n_i\}$ represents the number of working container instances that belong to tenant i. The initial state vector $(n_1, ..., n_K)$ represents a fully-working system that runs the maximum number of container instances per tenant (reported in red in Fig. 5). As faults occur, the system moves to states with lower values in the vector. In the worst case, all container instances are failed (all values in the vector are zero). Similarly, as container instances are recovered, the system moves to states with higher values in the vector. For instance, the vector $(n_1, ..., n_i - 1, ..., n_K)$

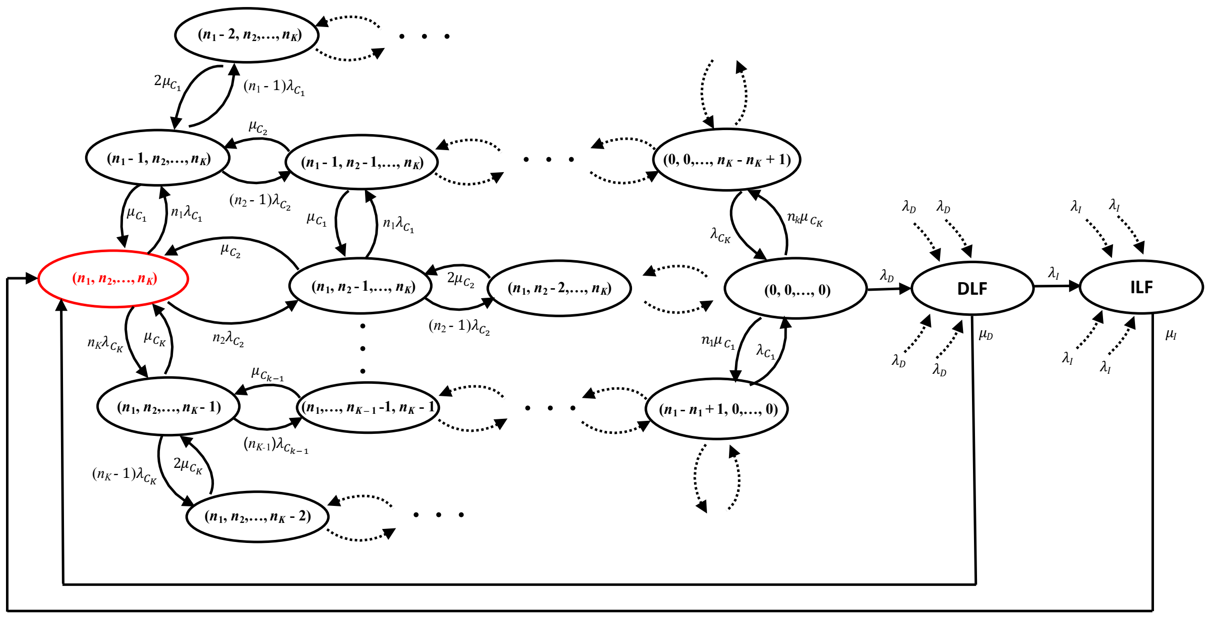


Fig. 5. Transition-state diagram of CNF multi-state model. States are labeled with a vector that represents, for each tenant $1 \dots K$, how many container instances are available. The state vector $(n_1, ..., n_K)$ in red indicates a fully-working system that runs the maximum number of container instances per tenant. States DLF and ILF indicate Docker and Infrastructure layers failures, respectively.

indicates a state where one-out-of-n_i container instances of the i-th tenant is failed. All the failure interarrivals are supposed to be independent and identically distributed (*iid*) random variables according to an exponential distribution with parameter λ_{C_i}. The duration of a repair action of a failed containerized instance is assumed to be an exponential random variable with parameter μ_{C_i}. All the failure and repair times are supposed to be independent of each other. Thus, the transition rates are proportional to the number of container instances that can fail and that can be recovered at each state. The state *Docker Layer Failure* (DLF) indicates the Docker failure condition which, in turn, causes the failure of all the containerized instances, and the corresponding state vector is $(0, ..., 0)_D$. For Docker too, failure interarrivals are supposed to be *iid* according to an exponential distribution with parameter λ_D, and independent from containerized instances failures. The duration of a repair action of this layer is assumed to be an exponential random variable with parameter μ_D. When Docker restarts, all the containerized instances are assumed to be restarted. Such a condition is taken into account by the transition from DLF state to initial (completely working system) state with rate μ_D.

The state *Infrastructure Layer Failure* (ILF) is associated with a crash of HW/OS part provoking a failure of the entire system. Again, in this case the state vector is $(0, ..., 0)_I$. Also for the Infrastructure layer, failure interarrivals are supposed to be *iid* according to an exponential distribution with parameter λ_I, and independent from upper layers failures. The duration of a repair action of this layer is assumed to be an exponential random variable with parameter μ_I. Similar to the previous case, when Infrastructure is restored, both Docker and Software layers are restored. Such a condition is taken into account by the transition from ILF state to initial (completely working system) state with rate μ_I.

It is useful to highlight that, the assumptions on the considered repair rates stem from the fact that mean-time-to-repairs (namely $1/\mu$) are approximately constant over time. It is especially valid for the software infrastructures (as in our case) where repair actions are meant to be reboot actions, as also confirmed by credited literature [41]–[43].

To derive the steady-state availability, we formally model the CNF as a Multi-State System (MSS). Let $\Omega_S = \prod_{i=1}^{K}\{0, 1, \dots, n_i\}$ be the state space of the Software layer, being $\{0, 1, \dots, n_i\}$ the state space of the tenant i, $i = 1, \dots, K$; let $\Omega_D = \{0, 1\}_D$ and $\Omega_I = \{0, 1\}_I$ be the state spaces of Docker and Infrastructure layers, respectively, where 0 indicates the failure condition whereas 1 refers to the working condition. Thus, the state space of the overall CNF is $\Omega = \Omega_S \times \Omega_D \times \Omega_I$.

Moreover, we define the *capacity level* $g_{i,\boldsymbol{\eta}}$ exposed by the CNF for tenant i in state $\boldsymbol{\eta}$ as

$$g_{i,\boldsymbol{\eta}} = \gamma \cdot \eta_i, \tag{1}$$

where γ has been previously defined as the serving capacity of a containerized instance for each tenant. Since the capacity of CNF can vary over time due to failures, the capacity level $g_{i,\boldsymbol{\eta}}$ is time-varying. The set including all possible capacity levels $\boldsymbol{g_\eta} = (\boldsymbol{g}_{1,\boldsymbol{\eta}}, ..., \boldsymbol{g}_{K,\boldsymbol{\eta}})$ for each CNF is

$$\mathcal{G} = \left\{ \gamma \cdot \boldsymbol{\eta} \;\middle|\; \boldsymbol{\eta} \in \prod_{i=1}^{K}\{0, \dots, n_i\} \right\} \cup \{(0, \dots, 0)_D, (0, \dots, 0)_I\}, \tag{2}$$

where $(0, \dots, 0)_D$ and $(0, \dots, 0)_I$ refer to the capacity levels of the DLF and ILF states, respectively. The total number of states in Fig. 5 is

$$N = |\Omega| = \prod_{i=1}^{K}(n_i + 1) + 2. \tag{3}$$

To characterize the considered MSS, it is useful to introduce the *structure function* $\varphi : \Omega \rightarrow$

$\left\{\boldsymbol{\eta} \;\middle|\; \boldsymbol{\eta} \in \prod_{i=1}^{K}\{0,\dots,n_i\}\right\} \cup \{(0,\dots,0)_D, (0,\dots,0)_I\}$ defined as follows:

$$\varphi(\boldsymbol{\eta}, x_D, x_I) = \begin{cases} \boldsymbol{\eta}, & x_D = 1, x_I = 1 \\ (0,\dots,0)_D, & x_D = 0, x_I = 1 \\ (0,\dots,0)_I, & x_I = 0, \end{cases} \tag{4}$$

where $x_D \in \Omega_D$ and $x_I \in \Omega_I$. It is a deterministic function [60] useful to specify the relation between the states of single elements (layers) and the state of the overall system (CNF). Now, the MSS of the CNF is completely described by the state space Ω, the serving capacity levels $\left\{\boldsymbol{\eta} \;\middle|\; \boldsymbol{\eta} \in \prod_{i=1}^{K}\{0,\dots,n_i\}\right\} \cup \{(0,\dots,0)_D, (0,\dots,0)_I\}$, and the structure function φ. Let now be $\boldsymbol{X}(t) = (X_1(t),\dots,X_K(t))$ a Ω_S-valued stochastic process which denotes the failure-repair process of the Software layer; $X_D(t)$ a Ω_D-valued stochastic process which denotes the failure-repair process of Docker layer; $X_I(t)$ a Ω_I-valued stochastic process which denotes the failure-repair process of Infrastructure layer, all defined for $t \geq 0$. Each of the above processes is Markovian on its state space, being all the underlying random variables exponentially distributed and independent of each other. Thus, the stochastic process $\{\varphi(\boldsymbol{X}(t), X_D(t), X_I(t)), t \geq 0\}$ is represented by the transition-state diagram in Fig. 5. Moreover, the CNF capacity level at time $t \geq 0$ is expressed in terms of the vector stochastic process

$$\boldsymbol{G}(t) = (G_1(t), ..., G_K(t)) = \gamma \cdot \varphi(\boldsymbol{X}(t), X_D(t), X_I(t)), \tag{5}$$

with values in $\mathcal{G}$, and with (state) probability vector $\boldsymbol{p}(t)$ at time t collecting all the state probabilities $p_{\boldsymbol{\eta}}(t)$, being $p_{\boldsymbol{\eta}}(t) = \Pr\{\boldsymbol{G}(t) = \boldsymbol{g}_{\boldsymbol{\eta}}\}$. We note that $\boldsymbol{G}(t)$ is a CTMC process described by a transition-state diagram equal to the one in Fig. 5, except that each state vector must be multiplied by γ.

We see that $\boldsymbol{G}(t)$: *i)* has a finite state space with cardinality N given by (3); *ii)* is irreducible, since every state is reachable from every other state (see Fig. 5); *iii)* is homogeneous since its infinitesimal generator matrix $\mathbf{Q}$ has constant elements (given constant parameters of failure and repair random variables). From the above properties, $\boldsymbol{G}(t)$ is an ergodic CTMC with a unique steady-state probability vector $\boldsymbol{p} = \lim_{t\to\infty} \boldsymbol{p}(t)$, with $\boldsymbol{p}(t)$ given by the solution of

$$\frac{\mathrm{d}\boldsymbol{p}(t)}{\mathrm{d}t} = \boldsymbol{p}(t)\mathbf{Q}, \tag{6}$$

along with the normalization condition $\sum_{\boldsymbol{\eta}} p_{\boldsymbol{\eta}}(t) = 1$.

Again, $\boldsymbol{p}$ is a vector collecting the state-state probability $p_{\boldsymbol{\eta}}$ for each state $\boldsymbol{\eta}$, that is given by:

$$p_{\boldsymbol{\eta}} = \lim_{t\to\infty} p_{\boldsymbol{\eta}}(t) = \lim_{t\to\infty} \Pr\{\boldsymbol{G}(t) = \boldsymbol{g}_{\boldsymbol{\eta}}\}. \tag{7}$$

Accordingly, the (discrete) random vector $\boldsymbol{G} = (G_1, ..., G_K) \in \mathcal{G}$ corresponds to the asymptotic behavior of $\boldsymbol{G}(t)$ (in the limit of $t \to \infty$) and admits values in the set (2) with probabilities (7). In conclusion, the set of pairs $\{p_{\boldsymbol{\eta}}, \boldsymbol{g}_{\boldsymbol{\eta}}\}$ determines the steady-state behavior of a CNF in terms of serving capacity.

4.2 CNF Queueing model

Our queueing model reflects the typical behavior of real systems, where the requests arriving to the CNF are served according to a queue characterized by non-exponential service times. As regards the choice of the queueing model system, we lie not so far from technical literature where SIP-based servers are characterized through $M/M/1$ or $M/M/k$ infinite queueing systems (see [44]–[48]) with a fixed number of servers. Such models are based on the exponential assumption of service times that, in the field of network communications, might be quite unrealistic. In our work, indeed, we are able to directly measure the service times across our testbed (see Sect. 6.1), and we find out that the empirical distribution of service times is not well fitted by an exponential distribution. Accordingly, we adopt an $M/G/G_i(t)$ model accounting also for the time-varying number of servers due to the failure/repair process for each tenant i. For a given state of the model, $G_i(t) = \gamma\eta_i$ and thus $M/G/G_i(t)$ is equivalent to $M/G/\gamma\eta_i$.

Furthermore, we want to highlight an important relationship between queueing and failure/repair models in terms of time scales. As observed in [55], in the field of communication networks it is possible to distinguish various and different time scales, such as the failure time scales (FTS) and the service time scales (STS). The former governs the failure/repair processes, and the latter governs the typical queueing metrics (e.g. the service times). When a time scale completely dominates another one, it is possible to neglect the transient effects produced by the dominated time scale. In service chains such as the cIMS, FTS $\gg$ STS, since FTS is in the order of thousands of hours, while STS is in the order of milliseconds (see also the parameters in Table 2). Thus, a decoupling between FTS and STS can be reasonably assumed. This behavior leads to claim that, given a state $\boldsymbol{\eta}$, tenant queues reach their steady-states very quickly compared to the occurrence of faults. In summary, for each state $\boldsymbol{\eta}$, we use a steady-state $M/G/\gamma\eta_i$ queue model for tenant i to derive the delay model of a CNF.

We assume that call setup requests arriving to tenant i follow an arrival Poisson process (a common assumption in technical literature [49]–[51]) with parameter α_i. Let $1/\beta$ be the mean value of service times derived from the experiments, that we suppose to be one and the same for all requests and tenants. Now, since no closed forms are available to evaluate the mean delay introduced by the $M/G/\gamma\eta_i$ model to requests, we proceed in two steps: *i)* we derive the mean delay for $M/M/\gamma\eta_i$; *ii)* then, we compute the mean delay for the $M/G/\gamma\eta_i$ model by using the approximating formula known as the Kingman's law of congestion (see [52], [53]), that exploits the coefficients of variation of the measured service times.

Along with the first step, according to [54], the mean number of requests (or jobs) $\nu_{i,\boldsymbol{\eta}}$ for the $M/M/\gamma\eta_i$ model for tenant i in state $\boldsymbol{\eta}$ is:

$$E[\nu_{i,\boldsymbol{\eta}}] = \gamma\eta_i \cdot \rho_{i,\boldsymbol{\eta}} + \rho_{i,\boldsymbol{\eta}} \frac{(\gamma\eta_i \cdot \rho_{i,\boldsymbol{\eta}})^{\gamma\eta_i}}{\gamma\eta_i!} \frac{\pi_{i,\boldsymbol{\eta}}}{(1-\rho_{i,\boldsymbol{\eta}})^2}, \tag{8}$$

where *utilization factor* amounts to:

$$\rho_{i,\boldsymbol{\eta}} = \frac{\alpha_i}{\beta \cdot \gamma\eta_i},$$

and

$$\pi_{i,\boldsymbol{\eta}} = \left[\sum_{h=0}^{\gamma\eta_i - 1} \frac{(\gamma\eta_i \cdot \rho_{i,\boldsymbol{\eta}})^h}{h!} + \frac{(\gamma\eta_i \cdot \rho_{i,\boldsymbol{\eta}})^{\gamma\eta_i}}{\gamma\eta_i!} \frac{1}{1-\rho_{i,\boldsymbol{\eta}}} \right]^{-1}.$$

By virtue of Little's law, the mean delay is

$$E[d_{i,\boldsymbol{\eta}}] = \frac{E[\nu_{i,\boldsymbol{\eta}}]}{\alpha_i}. \tag{9}$$

In the second step, since the first and second moments of service times distribution are finite, we apply the Kingman's approximation to derive the mean delay introduced by the considered $M/G/\gamma\eta_i$ system (say $\delta_{i,\boldsymbol{\eta}}$) of tenant i in state $\boldsymbol{\eta}$, namely

$$\delta_{i,\boldsymbol{\eta}} \approx E[d_{i,\boldsymbol{\eta}}] \cdot \frac{1+CV_s^2}{2}, \tag{10}$$

being CV_s the coefficient of variation of the empirical service time[1]. As pointed by Whitt (see [57]), such an excellent approximation can be considered a special case of Allen-Cunneen approximation [58]. Obviously, $\delta_{i,\boldsymbol{\eta}}$ increases as $\gamma\eta_i$ reduces due to failures. On the other hand, $\delta_{i,\boldsymbol{\eta}}$ decreases as $\gamma\eta_i$ increases by virtue of repair actions. It is worth noting that, the Kingman's approximation can be easily generalized also to the case of generic arrival times, with a little modification of eq. (10) which can be rewritten as: $\delta_{i,\boldsymbol{\eta}} \approx E[d_{i,\boldsymbol{\eta}}] \cdot (CV_a^2 + CV_s^2)/2$, being CV_a^2 the coefficient of variation of the distribution of the arrivals.

Being $\mathcal{B}^K$ the Cartesian product of a set $\mathcal{B}$ for itself K times, to characterize the MSS describing the delay model, it is useful to introduce the structure function $\varphi_\Delta : \Omega \to \{\mathbb{R}^+ \cup \{+\infty\}\}^K$ defined as follows:

$$\varphi_\Delta(\boldsymbol{\eta}, x_D, x_I) = \begin{cases} (\delta_{1,\boldsymbol{\eta}}, \ldots, \delta_{K,\boldsymbol{\eta}}), & x_D = 1, x_I = 1 \\ (+\infty, \ldots, +\infty), & x_D = 0, x_I = 1 \\ (+\infty, \ldots, +\infty), & x_I = 0, \end{cases} \tag{11}$$

where $\delta_{i,\boldsymbol{\eta}}$ is given by (10), $i = 1, \ldots, K$, and where the infinite delay arises when there are no working containerized instances. The mean delays introduced by a single CNF at time $t \geq 0$ is the vector stochastic process

$$\boldsymbol{\Delta}(t) = (\Delta_1(t), ..., \Delta_K(t)) = \varphi_\Delta(\boldsymbol{X}(t), X_D(t), X_I(t)). \tag{12}$$

Similarly to the stochastic process $\boldsymbol{G}(t)$ in (5), $\boldsymbol{\Delta}(t)$ is an ergodic CTMC, and the random vector $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_K)$ corresponds to the asymptotic behavior of $\boldsymbol{\Delta}(t)$ as $t \to \infty$. Given $\boldsymbol{\delta}_{\boldsymbol{\eta}} = (\delta_{1,\boldsymbol{\eta}}, ..., \delta_{K,\boldsymbol{\eta}})$ the mean delays vector for each state $\boldsymbol{\eta}$, the set of pairs $\{p_{\boldsymbol{\eta}}, \boldsymbol{\delta}_{\boldsymbol{\eta}}\}$ determines exhaustively the steady-state performance behavior of the CNF in terms of mean delay, being $p_{\boldsymbol{\eta}}$ given by (7).

5 Modeling of the CNF service chain

Based on the CNF modeling from the previous section (which represents an individual CNF in the service chain), we here build a model for the whole service chain of multiple CNFs. Two important aspects emerge. First, the chain is made of tiers *connected in series* (e.g., see Fig. 6 in the context of the cIMS). Thus, the entire chain is supposed to be working when every tier m in the chain is working (e.g., $m \in$ {P, S, I, H} for the cIMS, where P, S, I, H, indicate for brevity P-CSCF, S-CSCF, I-CSCF, HSS, respectively). Second, each tier m consists of redundant CNFs *connected in parallel*, in order to improve performance and availability. We remark that a tier m acts as a *logical* entity, by dividing the load among the replicas in the tier, according to the *flow dispersion* hypothesis (any CNF is able to handle service requests - see [59]). Every replica includes all of the three layers of the CNF structure (Software, Docker, and Infrastructure).

We denote with CNF$^{(m,\ell)}$ the ℓ-th parallel CNF associated to tier m ($\ell = 1, \ldots, L_m$).

Now, we start to evaluate the mean CSD introduced by tier m, where each CNF$^{(m,\ell)}$ is modeled as an $M/G/G_i^{(m,\ell)}(t)$ queue for tenant i. Indeed, tier m is given by L_m parallel CNFs (see Fig. 6), and can be analyzed as a single $M/G/G_i^{(m)}(t)$ queue as consequence of the flow dispersion hypothesis. Similarly to the vector stochastic process defined in (12), let

$$\boldsymbol{\Delta}^{(m)}(t) = \left(\Delta_1^{(m)}(t), ..., \Delta_K^{(m)}(t)\right) \tag{13}$$

be the vector stochastic process containing the mean CSD introduced by tier m for each tenant. Remarkably, $\Delta_i^{(m)}(t)$ is the stochastic process describing the $M/G/G_i^{(m)}(t)$ queue, that can be computed like in Section 4.2, by replacing $\gamma\eta_i$ with $G_i^{(m)}(t) = \sum_{\ell=1}^{L_m} G_i^{(m,\ell)}(t)$ in equations from (8) to (10).

Since the call flow traverses the service chain, the overall mean CSD is the sum of mean CSDs introduced by each single tier, namely

$$\boldsymbol{\Delta^c}(t) = (\Delta_1^c(t), ..., \Delta_K^c(t)) = \sum_m \boldsymbol{\Delta}^{(m)}(t). \tag{14}$$

Similarly to the derivation of $\{p_{\boldsymbol{\eta}}, \boldsymbol{\delta}_{\boldsymbol{\eta}}\}$ previously obtained for a single CNF, $\boldsymbol{\Delta}^{(m)}(t)$ and $\boldsymbol{\Delta}^c(t)$ are ergodic CTMCs, and $\boldsymbol{\Delta}^{(m)}$ and $\boldsymbol{\Delta}^c$ correspond to their asymptotic behaviors as $t \to \infty$. More technical details about the derivation of $\boldsymbol{\Delta}^{(m)}(t)$ and $\boldsymbol{\Delta}^c(t)$ (obtained by introducing the series and parallel structure functions) are provided in the Appendix A.

Accordingly, given $\boldsymbol{\delta}_{\boldsymbol{\eta}}^{(m)} = \left(\delta_{1,\boldsymbol{\eta}}^{(m)}, ..., \delta_{K,\boldsymbol{\eta}}^{(m)}\right)$ the mean delays vector of tier m in state $\boldsymbol{\eta}$, and $p_{\boldsymbol{\eta}}^{(m)} = \lim_{t\to\infty} \Pr\{\boldsymbol{\Delta}^{(m)}(t) = \boldsymbol{\delta}_{\boldsymbol{\eta}}^{(m)}\}$ the corresponding limiting probability, the set of pairs $\left\{p_{\boldsymbol{\eta}}^{(m)}, \boldsymbol{\delta}_{\boldsymbol{\eta}}^{(m)}\right\}$ represents the steady-state mean CSD distribution of tier m. Likewise, given $\boldsymbol{\delta}_{\boldsymbol{\eta}}^c = \left(\delta_{1,\boldsymbol{\eta}}^c, ..., \delta_{K,\boldsymbol{\eta}}^c\right)$ the mean delays vector of system in state $\boldsymbol{\eta}$, and $p_{\boldsymbol{\eta}}^c = \lim_{t\to\infty} \Pr\{\boldsymbol{\Delta}^c(t) = \boldsymbol{\delta}_{\boldsymbol{\eta}}^c\}$ the corresponding limiting probability, the set of pairs $\{p_{\boldsymbol{\eta}}^c, \boldsymbol{\delta}_{\boldsymbol{\eta}}^c\}$ is the steady-state mean CSD distribution of the entire service chain.

Letting $J^{(m)} = \prod_{\ell=1}^{L_m} N^{(m,\ell)}$ be the number of states of tier m for each parallel CNF ℓ, with $N^{(m,\ell)}$ given by (3), the number of states corresponding to the service chain is

$$J^c = \prod_{m \in \{P,S,I,H\}} J^{(m)}. \tag{15}$$

We consider the multi-tenant infrastructure as *available* when *every* operator (or tenant) guarantees a mean CSD less than a (maximum) tolerated value for its customers.

1. Such an approximation holds either for individual queues and for open non-Markovian network of queues (see [52], [56]).

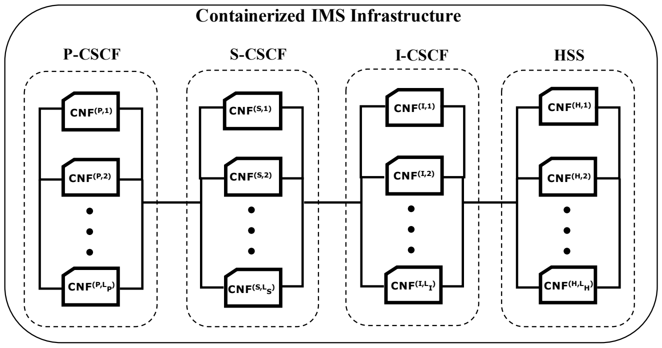


Fig. 6. Multi-tenant cIMS system, where parallel CNFs are introduced for redundancy purposes. CNF$^{(m,\ell)}$ refers to parallel CNF ℓ ($\ell = 1,\ldots,L_m$) of tier m, with $m \in${P-CSCF, S-CSCF, I-CSCF, HSS}.

Let $\boldsymbol{W}^c(t) = (W_1^c(t),...,W_K^c(t))$ be a K-dimensional vector containing the maximum tolerated values per tenant i at time t. The *instantaneous availability* $A^c\left[t, \boldsymbol{W}^c(t)\right]$ is defined (see [59], [60]) as the probability that mean CSD of the service chain for each tenant i (at $t > 0$) is not greater than $W_i^c(t)$, $i = 1,...,K$, viz.

$$A^c\left[t, \boldsymbol{W}^c(t)\right] = \Pr\{\Delta_i^c(t) - W_i^c(t) \leq 0, \ \ \forall i = 1,...,K\}. \tag{16}$$

Given constant maximum values $\boldsymbol{W}^c(t) = \boldsymbol{w}^c = (w_1^c,...,w_K^c)$, the *steady-state availability* $A^c\left(\boldsymbol{w}^c\right)$ can be derived from (16) for $t \to \infty$, as

$$A^c(\boldsymbol{w}^c) = \sum_{\boldsymbol{\eta}\in\Omega^c} p_{\boldsymbol{\eta}}^c \cdot \mathbf{1}\left(\delta_{i,\boldsymbol{\eta}}^c \leq w_i^c, \forall i = 1,...,K\right), \tag{17}$$

where $\mathbf{1}(\cdot)$ amounts to 1 if condition holds true and 0 otherwise, and $\Omega^c = \prod_m \Omega^{L_m}$.

5.1 Steady-state availability using the MUGF technique

The Multidimensional Universal Generating Function (MUGF) technique [17] provides an efficient method to evaluate the steady-state availability of (17). Being the MUGF a special case of probability generating function of a multivariate random variable, the steady-state distribution of an MSS can be expressed through a polynomial-shape form. More precisely, the MUGF of the steady-state mean CSD distribution pertaining to the tier m is

$$u^{(m)}(\boldsymbol{z}) = \sum_{\boldsymbol{\eta}\in\Omega^{L_m}} p_{\boldsymbol{\eta}}^{(m)} \prod_{i=1}^{K} z_i^{\delta_{i,\boldsymbol{\eta}}^{(m)}}, \tag{18}$$

a function of the vector indeterminate $\boldsymbol{z} = (z_1,\ldots,z_K)$.

From the generating functions theory, the sum of multivariate independent random variables has a generating function given by the product of the generating functions of single variables. Accordingly, by recalling that mean CSD of the service chain is the sum of mean CSDs introduced by each tier, the MUGF $u^c(\boldsymbol{z})$ is the product of the MUGFs of single tiers computed by (18), viz.

$$u^c(\boldsymbol{z}) = \prod_m \left[\sum_{\boldsymbol{\eta}\in\Omega^{L_m}} p_{\boldsymbol{\eta}}^{(m)} \prod_{i=1}^{K} z_i^{\delta_{i,\boldsymbol{\eta}}^{(m)}}\right]. \tag{19}$$

Thus, $u^c(\boldsymbol{z})$ represents a polynomial-shape function in $z_1,\ldots,z_K$. The $u^c(\boldsymbol{z})$ can be easily computed by combining the individual MUGFs through products and sums. The resulting expression of $u^c(\boldsymbol{z})$ provides the mean CSD vector $\boldsymbol{\delta}_{\boldsymbol{\eta}}^c$ and the corresponding steady-state probabilities $p_{\boldsymbol{\eta}}^c$ of the whole service chain. For each state $\boldsymbol{\eta}$, vector $\boldsymbol{\delta}_{\boldsymbol{\eta}}^c$ collects all the exponents of $z_1,\ldots,z_K$, while $p_{\boldsymbol{\eta}}^c$ is the multiplicative coefficient. Such quantities are used in (17) to compute the steady-state availability $A^c(\boldsymbol{w^c})$ of the multi-tenant chain.

5.2 Redundancy optimization of the service chain

The proposed availability assessment method is useful to solve network design problems, such as the selection of an optimal configuration that satisfy a given availability objective. The problem of practical interest is to identify the configuration(s) minimizing the number of CNF replicas for each tier of the service chain.

We denote with the vector $\boldsymbol{\ell}$ a configuration of the multi-tenant service chain, that is, the number L_m of replicas for each tier m. In the case of the cIMS, $m \in$ {P, S, I, H}, and $\boldsymbol{\ell} = (L_P, L_S, L_I, L_H)$. Obviously, this approach can be easily applied to other SFC architectures by changing the set of elements belonging to the chain itself (namely the components in the series availability model).

We define $C^{(m,\ell)}$ as the cost of parallel CNF ℓ belonging to tier m. Thus, the cost of the configuration $\boldsymbol{\ell}$ of the multi-tenant service chain is

$$C^c(\boldsymbol{\ell}) = \sum_m \sum_{\ell=1}^{L_m} C^{(m,\ell)}. \tag{20}$$

By considering d_{max} the maximum tolerated value for the mean CSD for each tenant (which is typically provided by international standards, such as for the IMS [71]), namely $\boldsymbol{w}^c = (d_{max},\ldots,d_{max})$, the steady-state availability of the configuration $\boldsymbol{\ell}$, in terms of mean CSD, is given by (17), viz.

$$A^c(\boldsymbol{w}^c, \boldsymbol{\ell}) = \sum_{\boldsymbol{\eta}\in J^c} p_{\boldsymbol{\eta}}^c \cdot \mathbf{1}\left(\delta_{i,\boldsymbol{\eta}}^c \leq d_{max}, \forall i = 1,...,K\right). \tag{21}$$

Given an availability constraint A_0 (for example, $A_0 = 0.99999$ also known as "five nines" availability), we define the set of configurations satisfying such a constraint as $\mathcal{L}^c = \{\boldsymbol{\ell} : A^c(\boldsymbol{w}^c, \boldsymbol{\ell}) \geq A_0\}$.

Then, the system configurations with minimum deployment cost which satisfy the availability constraint A_0 are provided by solving the following optimization problem:

$$\boldsymbol{\ell}^* = \arg\min_{\boldsymbol{\ell}\in\mathcal{L}^c} C^c(\boldsymbol{\ell}). \tag{22}$$

In summary, the optimization problem (22) contains elements both from availability and queueing models, where the latter allows to estimate the latency introduced by a chain. Indeed, the first step consists in computing the steady-state availability of chains, defined in terms of latency (see (21)), and in building the set of configurations $\mathcal{L}^c$ guaranteeing the given availability constraint. The second step consists in identifying the configuration(s) minimizing the cost (20) represented by the solution(s) of (22).

The process allowing to build various configurations relies on a greedy stage. By starting from a baseline configuration with 1 CNF per tier, the routine automatically adds 1 CNF per node up to a given threshold and evaluates the configuration availability. Since it is impossible to know beforehand which is the number of CNFs needed to obtain at least one solution, we set a configurable threshold value (namely, an integer number of CNFs) which represents the maximum number of CNFs a node can host. Obviously, depending on failure/repair parameters, it may happen that a given availability target (i.e., five nines) is never reached. In such a case, the network designer must relax the availability constraint (i.e. four nines). Once the routine has terminated its run, it produces a set of possible configurations, and it will choose the one ($\boldsymbol{\ell}^*$) satisfying the required condition. Interestingly, our routine allows to retain also the (sub-optimal) configurations, to leave the network designer the possibility of choosing a different setting (e.g. a configuration with $A^c(\boldsymbol{w}^c) = 0.99998$, which is barely far from the five nines requirement).

6 Experimental Results

This section consists of two parts: the first one contains a detailed description of the testbed deployed to derive realistic parameters such as: repair rates of various components by adopting fault injection techniques, mean service times employed by containerized software instance to manage cIMS requests, mean call setup delays experimented by multimedia calls. The second part pertains to the availability assessment performed through MUGF technique by exploiting the estimated parameters.

6.1 The cIMS testbed

We deploy from scratch an experimental testbed aimed at validating the proposed technique in a realistic NFV-based environment. Our testbed is composed of hardware and software technologies commonly adopted in cloud datacenters such as: operating systems based on Linux kernel 4.4.0, Docker engines (version 19.03.5) running on 16-Core 1.80GHz Intel Xeon machines with 64GB RAM, two 500GB SATA HDD, four 1-Gbps Intel Ethernet NICs, and one NetApp Network Storage Array with 32TB of storage space and 4GB of SSD cache. The hosts are connected to a 1-Gbps Ethernet network switch. The testbed includes 4 machines (each of which equipped with the aforementioned HW/SW), and relies on Clearwater (release 130), an open-source platform (later embodied in a commercial product [18]) which allows emulating a fully working IMS architecture in a container-based environment. We deploy the IMS functionalities (Bono, Sprout, Homestead) on top of Docker as depicted in Fig. 7. For each CNF we manage two different tenants. We use an external machine equipped with *SIPp* tool [76] (as a workload generator) and with HAproxy [77] to perform traffic balancing among all instances of Bono.

As in our previous studies [10], [62], we adopt *fault injection* to emulate faults and to measure the recovery times, in order to estimate representative model parameters. To assure the occurrence of failures, we emulate faults through the injection of their effects, which is also referred to as *error* or *failure injection* in some studies [63], [64]. In our experiments, we injected the following three types of faults.

Software layer faults: responsible for software crashes of the CNF upper layer which embeds the specific IMS

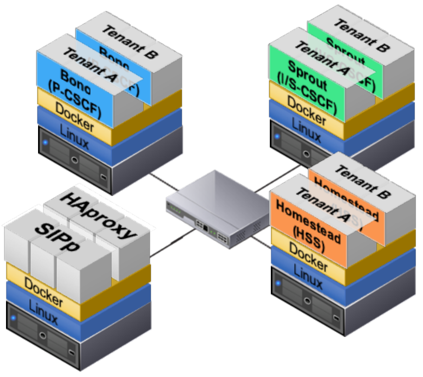


Fig. 7. The deployed cIMS architecture.

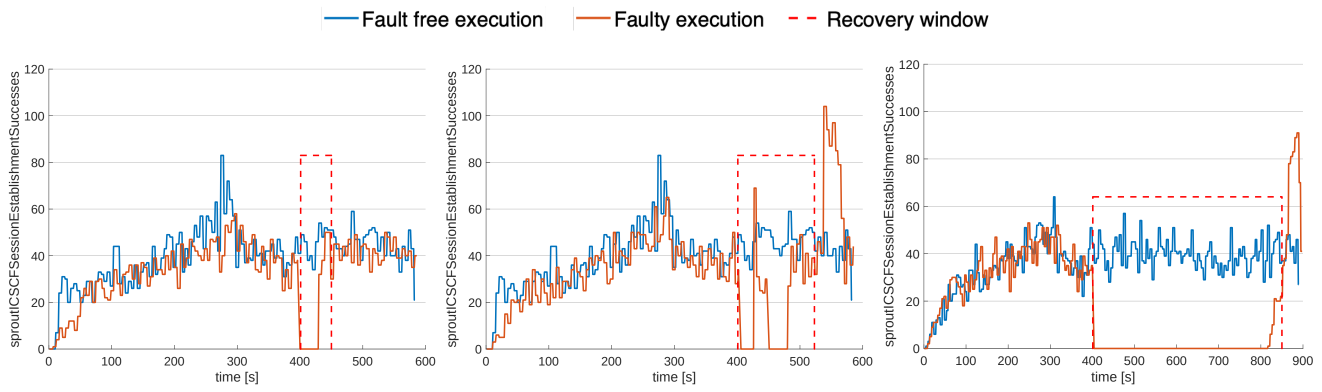
service logic. Typically, such faults include race condition bugs, resource exhaustion due to software aging bugs, I/O and exception handling bugs [75]. The Clearwater IMS is no exception, as a number of such failures have been reported by users on its issue tracker and mailing lists [74]. We inject these faults by forcing the abrupt termination of a container.

Docker layer faults: similarly for the containers, the Docker engine is affected by software faults related to timing, resource management, and other environmental conditions. Both academic research and end-users report recurring failures in Docker and in similar management software [65], [72], [73], causing the unavailability of containers along with virtual networks and storage volumes. We inject these faults by forcing the termination of container management services (i.e., the `dockerd` process) that, in turn, results in the termination of all containers running on top.

Infrastructure layer faults: software and hardware faults related to the crash of the hypervisor and the underlying physical infrastructure, respectively [10], [72]. In turn, these faults cause the unavailability of Docker and the whole set of containers. We inject these faults by forcing the abrupt shutdown of the machine.

We performed 30 fault injection experiments per CNF and per fault type, amounting to 360 experiments in total. Each fault injection experiment takes about 10 minutes both for the software layer and for the Docker engine, whereas it takes about 15 minutes for the infrastructure layer. Before injecting faults, we wait for an initial warm-up period (400 seconds) to let the system to reach a regime level. We automate fault injection experiments through ad-hoc routines developed to manage operations such as: start/stop fault injection, trigger the shutdown and the recovery of containers and hosts, collect metrics for each CNF through SNMP protocol as detailed in the following.

- **P-CSCF**: we analyze the number of SIP events (SIP messages) successfully passed to a Bono worker thread per unit time, reported in the SNMP object `bonoQueueSuccessFailSuccesses`.
- **S-CSCF**: we analyze the number of successful outgoing SIP transactions (INVITE messages) per unit time, reported in the SNMP object `sproutSCSCFOutgoingSIPTransactionsSuccesses`.
- **I-CSCF**: we analyze the number of successful terminating request attempts over



(a) Software layer fault.

(b) Docker layer fault.

(c) Infrastructure layer fault.

Fig. 8. Sprout I-CSCF under fault injection.

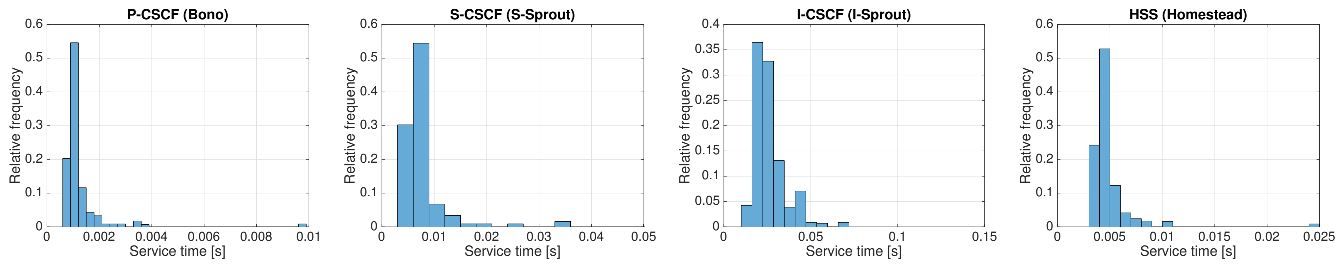


Fig. 9. Empirical service time distribution of Clearwater nodes (from the left: P-CSCF, S-CSCF, I-CSCF, HSS).

the period, reported in the SNMP object `sproutICSCFSessionEstablishmentSuccesses`.

- **HSS**: we analyze the number of enqueued Memcached requests per unit time, reported in the SNMP object `homesteadCacheQueueSizeCount`.

We use the aforementioned metrics to estimate the *recovery time* of the cIMS, in line with the previous NFV dependability benchmark study [10]. We measure the recovery time as the period during which a CNF is unavailable: it starts at a given fault injection time when the considered metric drops to zero, and it ends when the metric raises up to its regime value after recovery is completed. Each metric is evaluated at the output of a five-sample moving average filter, introduced to smooth fluctuations occurring during recovery. We consider the recovery procedure as completed when the metric overcomes a given threshold set to 90% of the fault-free metric level [78].

For instance, the recovery times of Sprout I-CSCF CNF after different injections are shown in panel of Figs. 8. Precisely, Fig. 8a, Fig. 8b, Fig. 8c report the recovery time in the presence of a Software layer fault, a Docker layer fault, and an infrastructure layer fault, respectively. We note that the differences between the curves is due to the typical variability of the experiments. Moreover, it is possible to receive a slightly different amount of traffic across executions, since load balancing cannot be perfectly uniform across replicas. The panel of Figs. 9 reports the service times distributions that we measure for each cIMS node. We remarked that these empirical distributions of service times are important to analyze the latency, namely to derive the coefficients of variation for each CNF to be used in (10).

Interestingly, our fault injection trials provided unexpected findings on the failure and recovery behavior of containerized network functions. Precisely, most studies adopt too simplistic assumptions about the containers time-to-recovery, by only considering the time to perform a restart action on a container (in the order of few seconds for a Linux container on a modern hardware machine). In contrast, our experiments reveal a longer time (in the order of minutes) to restore performance. A critical factor is represented by the restart of application software. Since the IMS is developed in the Java language, a new instance of the JVM needs to be allocated and initialized. Moreover, the application itself needs to manage allocations and data initialization (e.g., to start a thread pool). Afterwards, the performance is gradually restored, due to enqueueing and caching effects.

6.2 Availability assessment of cIMS

We perform a steady-state availability assessment of cIMS through the MUGF technique supported by the experimental data. Let us assume that: *i)* each CNF composing the system in Fig. 6 exhibits one and the same availability model namely, we suppose identical failure/repair parameters for each CNF), and, *ii)* all nodes exhibit one and the same cost amounting to 1, or equivalently, $C^{(m,\ell)} = 1$, $\forall \ell \in \{1, ..., L_m\}$, with $m \in \{\mathrm{P, S, I, H}\}$. Needless to say, the considered assumptions can be easily tailored across a variety of scenarios where network providers operate with different costs and performance features. Let us refer to an exemplary setting with $K = 2$ network providers (tenants), where the first one has $n_1 = 2$ containerized instances, and the second one has $n_2 = 3$ containerized instances. As a result, the MSS model of the CNF can be directly derived from the transition-state diagram depicted

TABLE 2
Input parameters

Parameter	Description	Value
$1/\lambda_C$	mean time for container failure[†]	1258 hours
$1/\lambda_D$	mean time for docker failure[†]	2516 hours
$1/\lambda_I$	mean time for infrastructure failure[†]	60000 hours
$1/\mu_C$	mean time for container repair[‡]	30 s
$1/\mu_D$	mean time for docker repair[‡]	60 s
$1/\mu_I$	mean time for infrastructure repair[‡]	5 min
α_1	IMS request arrival rate at tenant 1[‡]	$100\ \mathrm{s}^{-1}$
α_2	IMS request arrival rate at tenant 2[‡]	$200\ \mathrm{s}^{-1}$
$1/\beta_P$	P-CSCF empirical mean service time per request[‡]	$1.1 \cdot 10^{-3}$ s
$1/\beta_S$	S-CSCF empirical mean service time per request[‡]	$7.2 \cdot 10^{-3}$ s
$1/\beta_I$	I-CSCF empirical mean service time per request[‡]	$4.1 \cdot 10^{-2}$ s
$1/\beta_H$	HSS empirical mean service time per request[‡]	$4.6 \cdot 10^{-3}$ s
CV_P	P-CSCF coefficient of variation (Kingman's approx.)[‡]	0.7538
CV_S	S-CSCF coefficient of variation (Kingman's approx.)[‡]	0.9826
CV_I	I-CSCF coefficient of variation (Kingman's approx.)[‡]	0.5581
CV_H	HSS coefficient of variation (Kingman's approx.)[‡]	0.4631
d_{max}	Maximum tolerated CSD	50 ms

[†] From scientific literature
[‡] From experiments

in Fig. 10 with a number of different states amounting to $N = (n_1+1)(n_2+1)+2 = 14$ according to (3). The steady-state probability distribution of a single CNF can be directly obtained by solving the system of differential equations (6) for $t \to \infty$, as detailed in Appendix B.

Table 2 summarizes the input parameters adopted for the assessment, where: failure and repair rates have been, in part, derived from the deployed testbed, and, in part, chosen according to the technical literature (see [34], [66]–[70]). Let us provide clarifications about some parameters estimated through the testbed. First, the presence of multiple containers does not dramatically affect the infrastructure reboot ruled by the μ_I parameter. Such a situation is commonly encountered in practice, where the network designer adopts some strategies to control the parameter variability (i.e., smart allocation of hardware resources, snapshots usage, CPU pinning).

Moreover, the α parameters are influenced by the capacity of the system. We calibrated the workload generator to run new subscribers and to generate traffic such that the system approaches its capacity, without degrading latency and without saturating CPU and memory. This scenario assumes that the number of containers in the system is scaled according to the workload, which is typical for services deployed on cloud computing infrastructures. Again, the empirical service times distributions (Figs. 9) are evaluated by analyzing the service rates of each node. From such data, we also derive the coefficient of variation per node, useful to obtain the Kingman's approximation (10). Finally, the d_{max} value is a pessimistic estimate chosen in accordance to the ITU-T standard specifications [71], where acceptable values are in the order of seconds, since they account for propagation delays in geographic networks that are obviously negligible in our testbed.

In keeping with the "five nines" availability requirement, we set $A_0 = 1 - 10^{-5}$ and solve the optimization problem (22). A routine in Mathematica® (available upon request) computes the MUGF (19) and serves to finally evaluate the steady-state availability of cIMS expressed through (17). Then, the routine selects, among all feasible configurations, the one(s) exhibiting the minimum cost.

Table 3 reports the results of our experimental evaluation, where 12 exemplary configurations have been shown.

TABLE 3
Steady-state Availability under 12 configurations

Config.	Redundancy Level	$C^c(\boldsymbol{\ell})$	$A^c(\boldsymbol{w}^c)$
$\boldsymbol{\ell^*}$	$[CNF^{(P)}=2, CNF^{(S)}=1, CNF^{(I)}=3, CNF^{(H)}=2]$	8	0.999992
$\boldsymbol{\ell_1}$	$[CNF^{(P)}=2, CNF^{(S)}=2, CNF^{(I)}=2, CNF^{(H)}=2]$	8	0.999944
$\boldsymbol{\ell_2}$	$[CNF^{(P)}=2, CNF^{(S)}=3, CNF^{(I)}=2, CNF^{(H)}=2]$	9	0.999944
$\boldsymbol{\ell_3}$	$[CNF^{(P)}=3, CNF^{(S)}=3, CNF^{(I)}=2, CNF^{(H)}=3]$	11	0.999945
$\boldsymbol{\ell_4}$	$[CNF^{(P)}=1, CNF^{(S)}=1, CNF^{(I)}=3, CNF^{(H)}=3]$	8	0.999984
$\boldsymbol{\ell_5}$	$[CNF^{(P)}=1, CNF^{(S)}=1, CNF^{(I)}=2, CNF^{(H)}=1]$	5	0.999919
$\boldsymbol{\ell_6}$	$[CNF^{(P)}=2, CNF^{(S)}=1, CNF^{(I)}=2, CNF^{(H)}=1]$	6	0.999927
$\boldsymbol{\ell_7}$	$[CNF^{(P)}=2, CNF^{(S)}=2, CNF^{(I)}=2, CNF^{(H)}=1]$	7	0.999936
$\boldsymbol{\ell_8}$	$[CNF^{(P)}=3, CNF^{(S)}=3, CNF^{(I)}=3, CNF^{(H)}=1]$	10	0.999994
$\boldsymbol{\ell_9}$	$[CNF^{(P)}=2, CNF^{(S)}=2, CNF^{(I)}=3, CNF^{(H)}=2]$	9	0.9999999
$\boldsymbol{\ell_{10}}$	$[CNF^{(P)}=2, CNF^{(S)}=2, CNF^{(I)}=2, CNF^{(H)}=4]$	10	0.999968
$\boldsymbol{\ell_{11}}$	$[CNF^{(P)}=2, CNF^{(S)}=3, CNF^{(I)}=2, CNF^{(H)}=4]$	11	0.999968

The optimal configuration is $\boldsymbol{\ell^*}$, which exhibits a steady-state availability amounting to $A^c(\boldsymbol{w}^c) = 0.999992$, with a cost amounting to $C^c(\boldsymbol{\ell^*}) = 8$ CNFs. Such a configuration is obtained by considering (see the second column of Table 3 where CNF$^{(m)}$ denotes the number of CNFs for tier m): 2 redundant CNFs for P-CSCF and HSS, 3 CNF replicas for I-CSCF, and no redundancy for the S-CSCF. For the sake of clarity, we want to highlight that the same cost and availability values are obtained by exchanging the redundancy role of P-CSCF, S-CSCF, and HSS. This is due to the fact that the most critical network function turns to be the I-CSCF, in view of its higher service time (see pertinent values in Table 2). By exploring the remaining configurations, we can observe other interesting facts. Configuration $\boldsymbol{\ell_1}$ is obtained as a rearrangement of $\boldsymbol{\ell^*}$, where replicas have been differently distributed across the network functions (obviously, this results in the same cost). But, as can be noticed, this redistribution does not allow to meet the desired high availability requirement. In configuration $\boldsymbol{\ell_2}$, the redundancy of S-CSCF is empowered by 2 replicas w.r.t. $\boldsymbol{\ell^*}$, whereas one less replica is considered for I-CSCF. Also in this case, the steady-state availability fixes on "four nines", and the overall configuration cost increases to 9. Interestingly, configurations $\boldsymbol{\ell_1}$ and $\boldsymbol{\ell_2}$ exhibit the same value of availability even if an additional S-CSCF replica characterizes $\boldsymbol{\ell_2}$ w.r.t. $\boldsymbol{\ell_1}$. This behavior can be ascribed to the high efficiency of S-CSCF in handling IMS requests (see $1/\beta_S$ value in Table 2), translating in a very scarce sensitivity in improving its availability when the number of the corresponding CNF replicas exceeds the value of 2.

Even adding one more replica to P-CSCF and to HSS w.r.t. $\boldsymbol{\ell_2}$, the steady-state high availability target is not reached, and the whole cost jumps to 11 (configuration $\boldsymbol{\ell_3}$). Again, such a behavior can be ascribed to the weakness introduced by I-CSCF. This notwithstanding, in case we preserve a high redundancy degree for I-CSCF with 3 replicas (as occurs for the optimal configuration $\boldsymbol{\ell^*}$), the availability target remains unsatisfactory if two nodes are not replicated at all as shown in $\boldsymbol{\ell_4}$. Thus, guaranteeing a strong redundancy degree for the I-CSCF it is not enough.

A set of "cheap" configurations includes $\boldsymbol{\ell_5}$ - $\boldsymbol{\ell_7}$, whose costs range from 5 to 7. They show that when a more relaxed availability constraint (e.g. "four nines") has to be satisfied, it might be no necessary to add many CNF replicas. A limiting case is given by $\boldsymbol{\ell_5}$, whose availability value amounts to 0.999919 at a very cheap cost of 5. In

$$\begin{aligned} u^c(\boldsymbol{z}) = {} & 0.9997\, {z_1}^{0.0255} {z_2}^{0.0255} + 2.649 \times 10^{-5}\, {z_1}^{0.0255} {z_2}^{0.0255} + 8.773 \times 10^{-11}\, {z_1}^{0.0255} {z_2}^{0.0255} \\ & + 1.135 \times 10^{-29}\, {z_1}^{0.0304} {z_2}^{0.0329} + 4.013 \times 10^{-34}\, {z_1}^{0.0417} {z_2}^{0.0329} + \cdots + \ldots \\ & + 1.061 \times 10^{-44}\, {z_1}^{0.0587} {z_2}^{0.0329} + 2.121 \times 10^{-44}\, {z_1}^{0.0255} {z_2}^{0.0329} + \cdots + \ldots \\ & + 3.498 \times 10^{-49}\, {z_1}^{0.0450} {z_2}^{0.0589} + 2.867 \times 10^{-74}\, {z_1}^{0.0416} {z_2}^{0.0589} + \cdots + \ldots \\ & + 2.142 \times 10^{-58}\, {z_1}^{0.0418} {z_2}^{0.0570} + 1.654 \times 10^{-75}\, {z_1}^{0.0281} {z_2}^{0.0137} + \cdots + \ldots \end{aligned} \quad (24)$$

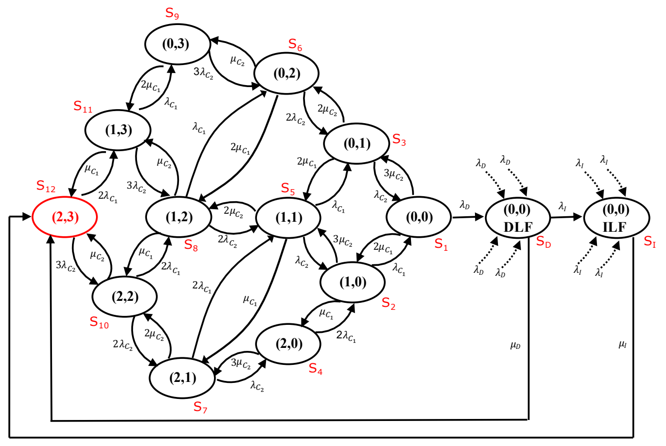


Fig. 10. Transition-state diagram of the CNF with two tenants.

contrast, the "five nines" requirement is met by $\boldsymbol{\ell}_8$ which cannot be the optimal configuration since its cost amounts to 10 (more expensive than $\boldsymbol{\ell}^*$). Furthermore, configuration $\boldsymbol{\ell}_9$ is obtained by adding one more replica to S-CSCF w.r.t. $\boldsymbol{\ell}^*$. In such a case, cost increases by one, but, the steady-state availability jumps to 0.9999999, a very challenging value sometimes required by mission-critical services. Interestingly, $\boldsymbol{\ell}_9$ exhibits a greater availability value than $\boldsymbol{\ell}_8$, but at a cheaper cost (amounting to 9). This behavior depends on the fact that in $\boldsymbol{\ell}_8$ no redundant CNF is allocated to HSS, and confirms that the best trade-off between availability and costs can be obtained through smart allocation of CNF replicas among the tiers. Finally, configurations $\boldsymbol{\ell}_{10}$ and $\boldsymbol{\ell}_{11}$ report two cases of CNF redundancy greater than 3 (for a tier). Precisely, $\boldsymbol{\ell}_{10}$ can be considered as an enhancement of configuration $\boldsymbol{\ell}_1$ with 2 more replicas on HSS tier. Likewise, $\boldsymbol{\ell}_{11}$ is derived by $\boldsymbol{\ell}_2$ adding 2 more CNFs to the HSS. In both cases, we do not observe any significant improvement in the availability values (both amounting to 0.999968 for $\boldsymbol{\ell}_{10}$ and $\boldsymbol{\ell}_{11}$) which are still far from the five nines availability target. Even, the costs of $\boldsymbol{\ell}_{10}$ and $\boldsymbol{\ell}_{11}$ increase by two units with respect to $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$, respectively. Once again, we can notice that a wrong redundancy strategy contributes to make the costs grow but not the availability values.

In (24) we report the MUGF of cIMS system in the optimal configuration $\boldsymbol{\ell}^*$, where most terms have been suppressed due to space constraints. The first term has a coefficient equal to 0.9997, which is the steady-state probability for a state corresponding to a mean CSD equal to 0.0255 s for both tenants, as indicated by the exponents of z_1 and z_2. Moreover, the terms in red refer to states where one or both mean CSD values do not respect the mean CSD constraint d_{max}, namely are greater than 50 ms, and their coefficient does not contribute to the value of the steady-state availability of cIMS according to (17).

From a computational complexity perspective, it is useful to highlight that the MUGF approach mitigates by far the computational load required by monolithic approaches which attempt to find the steady-state probability distribution of a single CTMC describing the whole system without decomposing it in simpler subsystems. Indeed, as regards the proposed example with $N^{(m,\ell)} = N = 14$, the state space of a single cIMS CTMC model amounts to $J^c = 14^{\sum_{m \in \{P,S,I,H\}} L_m}$ (by virtue of (15)). In particular, the optimization problem (22) requires to solve r^m systems of equations whose number ranges from 14^4 to J^c, being r the maximum redundancy level considered in the optimization algorithm. For instance, the optimal configuration $\boldsymbol{\ell}^* = (2, 1, 3, 2)$ requires the solution of a system with $14^{2+1+3+2} = 14^8$ equations. Conversely, our approach requires three steps: $i)$ we find the steady-state distribution of the CTMC of a single CNF with 14 states (namely we need to solve a system of 14 equations for each tier as detailed in Appendix B); $ii)$ we compute the steady-state distribution of the mean CSD due to tier m, for each m, and the corresponding MUGF by (18); $iii)$ we combine the distributions computed in step $ii)$ to obtain the mean CSD probability distribution of cIMS. Then, we get the steady-state availability (17). To compute the MUGF corresponding to the optimal configuration for the cIMS, we need about 360 s on a PC equipped with an Intel Quad-Core Xeon E5 CPU@3.7GHz.

6.3 Limitations

This work presented an availability model for multi-tenant service chains, tailored to the recent architectural trend towards containerized services. Although our efforts to match real systems (e.g., non-exponential service times assumptions, parameters from real-world experiments, etc.), some limitations necessarily remain: $i)$ for the sake of simplicity, in our model we consider CNFs having the same performance, but in real architectures, they might slightly differ. This notwithstanding, this assumption holds for many systems (including our cIMS case study), where all of the CNFs have been developed using the same software technology, have been assigned the same amount of resources (e.g., in terms of virtual CPUs), and share the same underlying layers (Docker and Infrastructure) in a cloud-based deployment; $ii)$ some parameters (e.g., the failure rates) are derived from the technical literature rather than from experiments, due to the large observation scale of failure events (up to some years). The other parameters are estimated using well-assessed fault injection techniques; $iii)$ we assume exponential distributions for failure/repair times: such an assumption is the most common and accepted across the technical literature, and in our work, it is necessary to manage complications arising from the joint availability and queueing modeling. Moreover, our proposed method sacrifices some high-level expressiveness achieved by other approaches (e.g., SPN/SRN), in order to benefit from visibility on the underlying analytical model. For example, this occurs in the eq. (18) where the MUGF expression of the mean delay distribution pertaining to the tier m is made explicit in terms of the pair $(p_{\boldsymbol{\eta}}, \boldsymbol{\delta_{\eta}})$. Benefiting from such a decomposition, the MUGF of the overall chain can be easily evaluated through a simple product as shown in eq. (19). Finally, we note that the MUGF method is intended to be applied "one-shot", using a fixed set of parameters, reflecting the expected workload, mean time to failure, mean time to repair, etc. This typically reflects the SLA-based approach of service providers, which are called to guarantee specific performance levels by fixing some constraints. When such constraints are violated, the SLA must be renegotiated, implying that the MUGF method has to be run again in order to return the best configuration guaranteeing the new performance level.

7 Conclusion

We propose an availability assessment approach to fit the modern Service Function Chain paradigm adopting: a Multi-State System model to represent the complex hardware and software stack in Containerized Network Functions; a queueing model, to include latency aspects; an extended version of multidimensional UGF technique, to efficiently analyze an infrastructure running several CNFs over multiple tenants, by combining their steady-state probability distributions through algebraic procedures.

We used an experimental fault-injection testbed to estimate parameters for the model, such as the repair rates of CNF layers and the service rates of containerized instances. The proposed approach allowed us to efficiently solve the availability optimization problem within few minutes.

This work might be extended in several directions such as: *i)* considering more and different service chains (e.g. mobile/broadband networks, data center chains) where the network manager is typically interested at finding the best redundant chain configuration at a minimal cost; *ii)* differentiating (by priority or importance) the requests entering a chain, so as to deploy a service chain able to satisfy Quality-of-Service constraints, as well.

References

[1] G. Davoli, W. Cerroni, C. Contoli, F. Foresta, F. Callegati, "Implementation of service function chaining control plane through OpenFlow," in *Proc. IEEE NFV-SDN*, 2017, pp.1–4.

[2] M. Gharbaoui, C. Contoli, G. Davoli, G. Cuffaro, B. Martini, F. Paganelli, W. Cerroni, P. Cappanera, P. Castoldi, "Experimenting latency-aware and reliable service chaining in Next Generation Internet testbed facility," in *Proc. IEEE NFV-SDN*, 2018, pp.1–4.

[3] D. Borsatti, G. Davoli, W. Cerroni, C. Contoli, F. Callegati, "Performance of Service Function Chaining on the OpenStack Cloud Platform," in *Proc. IEEE CNSM*, 2018, pp.432–437.

[4] IETF, "Service Function Chaining (SFC) Use Cases," [Online]. https://tools.ietf.org/id/draft-liu-sfc-use-cases-03.html.

[5] IETF, "Service Function Chaining Use Cases in Mobile Networks," [Online]. https://tools.ietf.org/id/draft-haeffner-sfc-use-case-mobility-00.html.

[6] NEC White Paper, "NEC virtualized evolved packet core - vEPC," [Online]. https://networkbuilders.intel.com/docs/vEPC_white_paper_w.cover_final.pdf.

[7] Ericsson Review, "Virtualizing network services - the telecom cloud," [Online]. https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/virtualizing-network-services---the-telecom-cloud.

[8] ETSI, "TS 123-251," [Online]. https://www.etsi.org/deliver/etsi_ts/123200_123299/123251/13.01.00_60/ts_123251v130100p.pdf.

[9] Cziva, R. & Pezaros, D. Container network functions: Bringing NFV to the network edge. *IEEE Commun. Mag.* . **55**, 24-31 (2017)

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

[11] Docker Inc., Docker Platform. [Online] https://www.docker.com/resources/what-container.

[12] ETSI, "TS 101-563," [Online]. https://www.etsi.org/deliver/etsi_ts/101500_101599/101563/01.03.01_60/ts_101563v010301p.pdf.

[13] A. Elnashar, M.A. El-Saidny, and M. Mahmoud, "Practical Performance Analyses of Circuit-Switched Fallback and Voice Over LTE," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1748–1759, 2017.

[14] J. E. Vargas Bautista, S. Sawhney, M. Shukair, I. Singh, V. K. Govindaraju, S. Sarkar, "Performance of CS Fallback from LTE to UMTS," *IEEE Commun. Mag.*, vol. 51, no. 9, pp. 136–143, 2013.

[15] H. Nemati, A. Singhvi, N. Kara and M. E. Barachi, "Adaptive SLA-based elasticity management algorithms for a virtualized IP multimedia subsystem," in *Proc. IEEE Globecom*, 2014, pp. 7-11.

[16] K. Al-Begain, A. Ali, *Multimedia Services and Applications in Mission Critical Communication Systems*. Hershey (PA), IGI Global, 2017.

[17] M. Di Mauro, M. Longo and F. Postiglione, "Availability Evaluation of Multi-tenant Service Function Chaining Infrastructures by Multidimensional Universal Generating Function," *IEEE Trans. Services Comput.*, vol. 14, no. 5, pp. 1320-1332, 2021.

[18] Clearwater Project, 2018 [Online]. https://clearwater.readthedocs.io/en/stable/.

[19] B. Tola, G. Nencioni, B. E. Helvik, Y. Jiang, "Modeling and Evaluating NFV-Enabled Network Services under Different Availability Modes," in *Proc. DRCN. IEEE*, 2019, pp.1–5.

[20] B. Tola, Y. Jiang, B. E. Helvik, "Failure process characteristics of cloud-enabled services," in *Proc. IEEE RNDM*, 2017, pp.1–7.

[21] B. Tola, G. Nencioni, B. E. Helvik, "Network-Aware Availability Modeling of an End-to-End NFV-enabled Service," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1389–1403, 2019.

[22] L. Qu, C. Assi, K. Shaban, M. J. Khabbaz, "A Reliability-Aware Network Service Chain Provisioning With Delay Guarantees in NFV-Enabled Enterprise Datacenter Networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 554–568, 2017.

[23] S. Sharma, A. Engelmann, A. Jukan, and A. Gumaste, "VNF Availability and SFC Sizing Model for Service Provider Networks," in IEEE Access, vol. 8, pp. 119768–119784, 2020.

[24] M. Wang, B. Cheng, S. Wang, and J. Chen, "Availability- and Traffic-Aware Placement of Parallelized SFC in Data Center Networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 182-194, 2021.

[25] A. Alleg, T. Ahmed, M. Mosbah, and R. Boutaba, "Joint Diversity and Redundancy for Resilient Service Chain Provisioning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1490-1504, 2020.

[26] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu, "Recent Advances of Resource Allocation in Network Function Virtualization," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 2, pp. 295-314, 2021.

[27] R. Kang, F. He, and E. Oki, "Robust Virtual Network Function Allocation in Service Function Chains with Uncertain Availability Schedule," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp.2987-3005, 2021.

[28] S. Yang, F. Li, R. Yahyapour, and X. Fu, "Delay-Sensitive and Availability-Aware Virtual Network Function Scheduling for NFV," *IEEE Trans. Services Comput.*, 2019.

[29] S. G. Kulkarni, G. Liu, K. K. Ramakrishnan, M. Arumaithurai, T. Wood, and X. Fu, "REINFORCE: Achieving Efficient Failure Resiliency for Network Function Virtualization-Based Services," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 695-708, 2020.

[30] S. G. Kulkarni, K. K. Ramakrishnan, and T. Wood, "Managing State for Failure Resiliency in Network Function Virtualization," in *Proc. IEEE LANMAN*, 2020, pp.1–6.

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

[32] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, and J. P. Jue, "Guaranteed-availability network function virtualization with network protection and VNF replication," in *Proc. IEEE GLOBECOM*, 2017.

[33] H. A. Alameddine, S. Ayoubi, and C. Assi, "An efficient survivable design with bandwidth guarantees for multi-tenant cloud networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 357–372, 2017.

[34] S. Sebastio, R. Ghosh, and T. Mukherjee, "An availability analysis approach for deployment configurations of containers," *IEEE Trans. Services Comput.*, vol. PP, no. 99, pp. 1–1, 2018.

[35] D. Bruneo, "A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems, " *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 560–569, 2014.

[36] E. Sousa, F. Lins, E. Tavares, P. Cunha, and P. Maciel, "A modeling approach for cloud infrastructure planning considering dependability and cost requirements," *IEEE Trans. Syst., Man, Cybern.*, vol. 45, no. 4, pp. 549–558, 2015.

[37] P. Sun, D. Wu, X. Qiu, L. Luo, and H. Li, "Performance analysis of cloud service considering reliability," in *Proc. IEEE QRS-C*, 2016, pp. 339–343.

[38] S. Yu, H. Chen, and Y. Xiang, "Maximal Service Profit in MAS-Based Cloud Computing Considering Service Security," in *Lecture Notes in Electrical Engineering, vol 355. Springer,* 2015.

[39] J. L. Peterson *Petri Net Theory and the Modeling of Systems*. USA, Prentice Hall, 1981.

[40] U. Herzog, "Formal Methods for Performance Evaluation," in *Lecture Notes in Computer Science, vol 2090. Springer,* 2001.

[41] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley & Sons, 2016.

[42] K. S. Trivedi, J. K. Muppala, S. P. Woolet, B. R. Haverkort, "Composite performance and dependability analysis," *Performance Evaluation*, vol. 14, no. 3-4, pp. 197–215, 1992.

[43] K. S. Trivedi, A. Bobbio, "Reliability and Availability Analysis in Practice," *Handbook of Advanced Performability Engineering. Springer, Cham*, 2021.

[44] 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," *Proc. Inst. Mech. Eng. O J. Risk Reliab.*, vol. 3, pp. 290–301, 2013.

[45] H. Shulzrinne, S. Narayanan, J. L. Doyle, "SIPstone - benchmarking SIP server performance", Tech. Rep., 2002.

[46] S. V. Subramanian, R. Dutta, "Measurements and Analysis of M/M/1 and M/M/c Queuing Models of the SIP Proxy Server," in *Proc. IEEE ICCCN*, 2009, pp. 1-7.

[47] S. V. Subramanian, R. Dutta, "A study of performance and scalability metrics of a SIP proxy server a practical approach," in *Journ. of Comp. System and Science*, vol. 77, no. 5, pp. 884–897, 2011.

[48] M. Di Mauro, A. Liotta, "Statistical Assessment of IP Multimedia Subsystem in a Softwarized Environment: a Queueing Networks Approach," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1493–1506, 2019.

[49] C. Shen, H. Schulzrinne, and E. Nahum, "Session Initiation Protocol (SIP) Server Overload Control: Design and Evaluation," in *IPTComm 2008. LNCS, vol 5310*, pp.149-173, 2008.

[50] T. Eyers, H. Schulzrinne, "Predicting Internet telephone call setup delay," in *Internet Telephony Workshop*, 2000

[51] G. Faraci, G. Schembra, "An Analytical Model to Design and Manage a Green SDN/NFV CPE Node," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 3, pp. 435–450, 2015.

[52] T. Kimura, "Approximations for multi-server queues: System interpolations," in *Queueing Systems*, vol. 17, no. 3, pp. 347–382, 1994.

[53] N. Gans, G. Koole, A. Mandelbaum,"Telephone Call Centers: Tutorial, Review, and Research Prospects," in *Manufacturing & Service Operations Management*, vol. 5, no. 2, pp. 79–141, 2003.

[54] L. Kleinrock, *Queueing systems, vol.2: computer applications*. New York, John Wiley & Sons, 1976.

[55] W. Whitt, *Stochastic-Process Limits: An Introduction to Stochastic-Process Limits and Their Application to Queues*. N.Y., Springer, 2001.

[56] W. Whitt,"The queueing network analyzer," in *Bell Syst. Techn. J.*, vol. 62, pp. 2779–2815, 1983.

[57] W. Whitt,"Approximations for the GI/G/m queue," in *Production and Operations Management*, vol. 2, pp. 114–161, 1993.

[58] A. O. Allen, *Probability, Statistics, and Queueing Theory*. Academic Press, Inc., San Diego, 2 ed., 1990.

[59] G. Levitin and A. Lisnianski, *Multi-state system reliability: assessment, optimization and applications*. Singapore, WS, 2003.

[60] G. Levitin, "A Universal Generating Function in the Analysis of Multi-state Systems," in: Handbook of Performability Engineering, Springer London, 2008.

[61] I. A. Ushakov, "A universal generating function," *Sov. Journ. of Comp. System and Science*, vol. 24, no. 5, pp. 37–49, 1986.

[62] D. Cotroneo, A.K. Iannillo, R. Natella, and S. Rosiello, "Dependability Assessment of the Android OS through Fault Injection," *IEEE Trans. Rel.*, doi=10.1109/TR.2019.2954384, pp. 1–16, 2019.

[63] R. Natella, D. Cotroneo, H. & Madeira, "Assessing dependability with software fault injection: A survey," *ACM Computing Surveys (CSUR)*. **48**, 1-55 (2016)

[64] A. Avizienis, J. Laprie, B. Randell,& C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*. **1**, 11-33 (2004)

[65] M. Torquato and M. Vieira, "An Experimental Study of Software Aging and Rejuvenation in Dockerd," in *Proc. IEEE EDCC*, 2019, pp.1–6.

[66] R. d. S. Matos, P. R. M. Maciel, F. Machida, D. S. Kim, and K. S. Trivedi, "Sensitivity analysis of server virtualized system availability," *IEEE Trans. Rel.*, vol. 61, no. 4, pp. 994–1006, 2012.

[67] D. A. Patterson, and J. L. Hennessy, *Computer Organization and Design*. Morgan Kaufmann, 4th ed., 2011.

[68] M. Kleppmann, *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media, 1st ed., 2017.

[69] D. Ford, F. Labelle, F. Popovici, M. Stokely, V. A. Truong, L. Barroso, C. Grimes, S. Quinlan "Availability in globally distributed storage systems ," in *Proc. Usenix*, 2010, pp.61–74.

[70] J. Bai, X. Chang, F. Machida, K. S. Trivedi, and Z. Han, "Analyzing Software Rejuvenation Techniques in a Virtualized System: Service Provider and User Views," *IEEE Access*, vol. 8, pp. 6448–6459, 2020.

[71] ITU-T, "G.1028 : End-to-end quality of service for voice over 4G mobile networks," [Online]. https://www.itu.int/rec/T-REC-G.1028/en.

[72] F. Machida, J. Xiang, K. Tadano, and Y. Maeno, "Aging-related Bugs in Cloud Computing Software," in *Proc. IEEE ISSRE*, 2012, pp. 287–292.

[73] R. Matos, J. Araujo, V. Alves, and P. Maciel, "Experimental evaluation of software aging effects in the Eucalyptus elastic block storage," in *Proc. IEEE SMC*, 2012, pp. 1103–1108.

[74] Clearwater project. (2016) ibcf (bono) VM using up a lot of memory. [Online]. https://github.com/Metaswitch/clearwater-mailing-list-archives/blob/master/2016-October.txt

[75] M. Grottke and K. S. Trivedi, "Fighting bugs: Remove, retry, replicate, and rejuvenate," *Computer*, vol. 40, no. 2, pp. 107–109, 2007.

[76] Gayraud, Richard and Jacques, Olivier and Day, Robert and Wright, Charles P.. (2019) SIPp HomePage. [Online]. http://sipp.sourceforge.net/

[77] Willy Tarreau. (2019) HAProxy HomePage. [Online]. http://www.haproxy.org/

[78] K. Kanoun and L. Spainhower, *Dependability benchmarking for computer systems*. Wiley Online Library, 2008, vol. 72.

Luigi De Simone (Ph.D.) is a postdoctoral researcher at the University of Naples Federico II, Italy. His research interests include dependability benchmarking, fault injection testing, virtualization reliability and its application on safety-critical systems.

Mario Di Mauro (Ph.D.) is an assistant professor at the University of Salerno, Italy. His main fields of interest include: network performance, network security and availability characterization, data analysis for novel telecommunication infrastructures.

Roberto Natella (Ph.D.) is an assistant professor at the University of Naples Federico II, Italy. His research interests include dependability benchmarking, software fault injection, software aging and rejuvenation, and their application in OS and virtualization technologies.

Fabio Postiglione (Ph.D.) is an associate professor of statistics at the University of Salerno, Italy. His research interests include statistical characterization of degradation processes, reliability and availability modeling of complex systems, and Bayesian methods.

Appendix A

We introduce two operators (namely series and parallel structure functions) to formally derive $\mathbf{\Delta}^{(m)}(t)$ and $\mathbf{\Delta}^{c}(t)$. For the sake of simplicity, let us start to define the *parallel structure function*:

$$\psi_p : \Omega^{L_m} \rightarrow \mathbb{R}^K \cup (+\infty, \dots, +\infty). \tag{25}$$

Accordingly, the mean CSD introduced by tier $m \in \{P, S, I, H\}$ is:

$$\begin{aligned} \mathbf{\Delta}^{(m)}(t) = \psi_p \Big(& \boldsymbol{X}^{(m,1)}(t), X_D^{(m,1)}(t), X_I^{(m,1)}(t), \dots, \\ & \boldsymbol{X}^{(m,L_m)}(t), X_D^{(m,L_m)}(t), X_I^{(m,L_m)}(t) \Big) = \left(\Delta_1^{(m)}(t), ..., \Delta_K^{(m)}(t) \right), \end{aligned} \tag{26}$$

where $\boldsymbol{X}^{(m,L_m)}(t)$ denotes the Ω_S-valued failure/repair process of the Software layer, $X_D^{(m,L_m)}(t)$ denotes the Ω_D-valued failure/repair process of the Docker layer, and $X_I^{(m,L_m)}(t)$ denotes the Ω_I-valued failure/repair process of the Infrastructure layer. Finally, $\Delta_i^{(m)}(t)$ is the stochastic process describing the $M/G/G_i^{(m)}(t)$ queue, that can be computed like in Section 4.2, by replacing $\gamma\eta_i$ with $G_i^{(m)}(t) = \sum_{\ell=1}^{L_m} G_i^{(m,\ell)}(t)$ in equations from (8) to (10).

It is now useful to recall that, since the call flow traverses the cIMS chain, the overall mean CSD is the sum of mean CSDs introduced by each single tier.

Accordingly, by introducing $L_{\text{tot}} = \sum_{m \in \{P,S,I,H\}} L_m$, we define the *series structure function*:

$$\psi_s : \Omega^{L_{\text{tot}}} \rightarrow \mathbb{R}^K \cup (+\infty, \dots, +\infty). \tag{27}$$

Thus, the overall mean delay $\mathbf{\Delta^c}(t) = (\Delta_1^c(t), ..., \Delta_K^c(t))$ introduced by the cIMS is given by:

$$\begin{aligned} \mathbf{\Delta}^c(t) = \sum_{m \in \{P,S,I,H\}} \mathbf{\Delta}^{(m)}(t) = \psi_s \Big(& \boldsymbol{X}^{(P,1)}(t), X_D^{(P,1)}(t), X_I^{(P,1)}(t), \dots, \ \boldsymbol{X}^{(P,L_P)}(t), X_D^{(P,L_P)}(t), X_I^{(P,L_P)}(t), \dots, \\ & \boldsymbol{X}^{(H,1)}(t), X_D^{(H,1)}(t), X_I^{(H,1)}(t), \dots, \ \boldsymbol{X}^{(H,L_H)}(t), X_D^{(H,L_H)}(t), X_I^{(H,L_H)}(t) \Big) \\ = \sum_{m \in \{P,S,I,H\}} \psi_p \Big(& \boldsymbol{X}^{(m,1)}(t), X_D^{(m,1)}(t), X_I^{(m,1)}(t), \dots, \ \boldsymbol{X}^{(m,L_m)}(t), X_D^{(m,L_m)}(t), X_I^{(m,L_m)}(t) \Big). \end{aligned} \tag{28}$$

APPENDIX B

Let $p_1(t), \dots, p_{12}(t)$ be the state probabilities corresponding to states $S_1, \dots, S_{12}$, and $p_I(t)$ and $p_D(t)$ the state probabilities corresponding to states S$_I$ and S$_D$, respectively, as shown in Fig. 10.

According to (6), by assuming one and the same model for each CNF composing the cIMS system, all the state probabilities at time t can be derived by solving the system of 14 differential equations (29), representative of the 14-state MSS in Fig. 10, with the constraint $\sum_{i=1}^{12} p_i(t) + p_D(t) + p_I(t) = 1$, and the assumption that the node is initially working.

$$\begin{cases}
\frac{dp_I(t)}{dt} = -\mu_I p_I(t) + \lambda_I \sum_{i=1}^{12} p_i(t) + p_D(t) \\
\frac{dp_D(t)}{dt} = -(\mu_D + \lambda_I) p_D(t) + \lambda_D \sum_{i=1}^{12} p_i(t) \\
\frac{dp_1(t)}{dt} = -(2\mu_{C_1} + 3\mu_{C_2} + \lambda_D + \lambda_I) p_1(t) + \lambda_{C_1} p_2(t) + \lambda_{C_2} p_3(t) \\
\frac{dp_2(t)}{dt} = 2\mu_{C_1} p_1(t) - (\lambda_{C_1} + \mu_{C_1} + 3\mu_{C_2} + \lambda_D + \lambda_I) p_2(t) + 2\lambda_{C_1} p_4(t) + \lambda_{C_2} p_5(t) \\
\frac{dp_3(t)}{dt} = 3\mu_{C_2} p_1(t) - (\lambda_{C_2} + 2\mu_{C_1} + 2\mu_{C_2} + \lambda_D + \lambda_I) p_3(t) + \lambda_{C_1} p_5(t) + 2\lambda_{C_2} p_6(t) \\
\frac{dp_4(t)}{dt} = \mu_{C_1} p_2(t) - (2\lambda_{C_1} + 3\mu_{C_2} + \lambda_D + \lambda_I) p_4(t) + \lambda_{C_2} p_7(t) \\
\frac{dp_5(t)}{dt} = 3\mu_{C_2} p_2(t) + 2\mu_{C_1} p_3(t) - (\lambda_{C_1} + \lambda_{C_2} + \mu_{C_1} + 2\mu_{C_2} + \lambda_D + \lambda_I) p_5(t) + 2\lambda_{C_1} p_7(t) + 2\lambda_{C_2} p_8(t) \\
\frac{dp_6(t)}{dt} = 2\mu_{C_2} p_3(t) - (2\lambda_{C_2} + 2\mu_{C_1} + \mu_{C_2} + \lambda_D + \lambda_I) p_6(t) + \lambda_{C_1} p_8(t) + 3\lambda_{C_2} p_9(t) \\
\frac{dp_7(t)}{dt} = 3\mu_{C_2} p_4(t) + \mu_{C_1} p_5(t) - (2\lambda_{C_1} + \lambda_{C_2} + 2\mu_{C_2} + \lambda_D + \lambda_I) p_7(t) + 2\lambda_{C_2} p_{10}(t) \\
\frac{dp_8(t)}{dt} = 2\mu_{C_2} p_5(t) + 2\mu_{C_1} p_6(t) - (\lambda_{C_1} + 2\lambda_{C_2} + \mu_{C_1} + \mu_{C_2} + \lambda_D + \lambda_I) p_8(t) + 2\lambda_{C_1} p_{10}(t) + 3\lambda_{C_2} p_{11}(t) \\
\frac{dp_9(t)}{dt} = \mu_{C_2} p_6(t) - (3\lambda_{C_2} + 2\mu_{C_1} + \lambda_D + \lambda_I) p_9(t) + \lambda_{C_1} p_{11}(t) \\
\frac{dp_{10}(t)}{dt} = 2\mu_{C_2} p_7(t) + \mu_{C_1} p_8(t) - (2\lambda_{C_1} + 2\lambda_{C_2} + \mu_{C_2} + \lambda_D + \lambda_I) p_{10}(t) + 3\lambda_{C_2} p_{12}(t) \\
\frac{dp_{11}(t)}{dt} = \mu_{C_2} p_8(t) + 2\mu_{C_1} p_9(t) - (\lambda_{C_1} + 3\lambda_{C_2} + \mu_{C_1} + \lambda_D + \lambda_I) p_{11}(t) + 2\lambda_{C_1} p_{12}(t) \\
\frac{dp_{12}(t)}{dt} = \mu_{C_2} p_{10}(t) + \mu_{C_1} p_{11}(t) + \mu_D p_D(t) + \mu_I p_I(t) - (2\lambda_{C_1} + 3\lambda_{C_2} + \lambda_D + \lambda_I) p_{12}(t)
\end{cases} \tag{29}$$

The steady-state probability distribution $\boldsymbol{p}$ of the CNF in Fig. 10 can be hence derived by considering the limit for $t \to \infty$ of the solution of system (29). Alternatively, $\boldsymbol{p}$ can be can be simply determined by nullifying the derivative terms in system (29), and solving it along with the constraint $\sum_{i=1}^{12} p_i + p_D + p_I = 1$.