Multi-Provider IMS Infrastructure With Controlled Redundancy: A Performability Evaluation

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

***Abstract*—In modern telecommunication networks, services are provided through Service Function Chains (SFC), where network resources are implemented by leveraging virtualization and containerization technologies. In particular, the possibility of easily adding or removing network resources has prompted service providers to redefine some concepts including performance and availability. In line with this new trend, we propose a performability study of a multi-provider containerized IP Multimedia Subsystem (cIMS), an SFC-like infrastructure used in the core part of 4G/5G networks to handle multimedia sessions. On the one hand, performance issues are tackled by modeling each cIMS node in terms of a *G/G/m* queueing system to derive the Call Setup Delay (CSD), a performance metric related to the user-end experience in multimedia communications. On the other hand, availability issues are addressed through the Multi-State System (MSS) formalism, to take into account different *performance rates* of the system. Then, we devise an algorithm called PE-MUGF (Performability Evaluation through Multidimensional Universal Generating Function) to identify the minimum-redundancy cIMS configuration which meets given performance and availability targets at the same time. Finally, an extensive experimental analysis based on *Clearwater*, a containerized IMS testbed, allows us to estimate most of system parameters whose robustness is evaluated through a sensitivity analysis.**

***Index Terms*—SFC performance, availability, performability analysis, controlled redundancy.**

I. Introduction and Contribution

SERVICE function chains (SFCs) have revolutionized the way to provide telecommunication services thanks to the flexibility to reconfigure on-demand hardware and software resources needed to provide specific functionalities [1], [2]. Since SFCs are arranged as a series of software-based nodes to be traversed in a predefined order, a network operator can decide to insert or remove one or more nodes from the chain aimed at modifying the service provisioning. Likewise, it is possible to add or remove redundant nodes to strengthen or relax performance and availability (or, simply, performability) requirements. Obviously, an increased redundancy implies higher costs, thus, an accurate modeling and planning of additional resources is desirable.

Accordingly, we propose a performability evaluation of a container-based version of IP Multimedia Subsystem (cIMS), a popular framework typically deployed as an SFC-like architecture, which is broadly exploited in the core part of 4G/5G networks to support multimedia communications [3], [4], [5]. Due to its generality, our assessment can be easily employed to characterize performability of different chained structures which today are often implemented via *softwarization* paradigm. Examples includes: *i*) WAN softwarized chains, where the data flow may traverse in sequence softwarized elements such as an intrusion detection system, a load balancer, and a router, before arriving into the Internet core; *ii*) Radio softwarized chains, where the software defined radio paradigm allows to realize, completely in software, also radio access elements resulting in chains made of: base stations, radio network controllers, signalling/packet gateways, data network; *iii)* WLAN softwarized chains, where the wireless local data traffic can sequentially traverse (for example) an access point, a firewall and a Web server.

In line with modern cloud concepts, the considered IMS infrastructure is assumed to be shared among different providers, thus, we refer to a *multi-provider* cIMS. Remarkably, the multi-provider qualification is an opportunity offered by softwarized network systems where, aimed at a cost reduction, part of the infrastructure is in common. Today, multi-provider solutions include: [6], where a configuration known as Gateway Core Network (GWCN) allows different network operators to connect to a shared radio access network; [7], where operators can deploy access or core nodes in an independent way, but sharing with other providers a common infrastructure; [8], where an exemplary multi-operator IMS framework is deployed into a virtual data center, with different services offered by (different) operators having the same common physical infrastructure.

It is useful to decompose our analysis into two parts: the first one concerns the performance aspects that we take into account through the Call Setup Delay (CSD), a performance indicator [9], [10], [11] defined as the time difference between

Manuscript received 9 March 2023; revised 29 May 2023; accepted 1 June 2023. Date of publication 5 June 2023; date of current version 12 December 2023. The work of Luigi De Simone was supported by the European Union FSE-REACT-EU, PON Research and Innovation 201–2020 DM1062/2021 under Contract 18-I-15350-6. The associate editor coordinating the review of this article and approving it for publication was S. Kanhere. *(Corresponding author: Mario Di Mauro.)*

Luigi De Simone and Roberto Natella are with the Department of Electrical Engineering and Information Technologies, University of Napoli Federico II, 80125 Naples, Italy (e-mail: luigi.desimone@unina.it; roberto.natella@unina.it).

Mario Di Mauro, Maurizio Longo, and Fabio Postiglione are with the Department of Information and Electrical Engineering and Applied Mathematics, University of Salerno, 84084 Fisciano, Italy (e-mail: mdimauro@unisa.it; longo@unisa.it; fpostiglione@unisa.it).

Digital Object Identifier 10.1109/TNSM.2023.3282745

the first message sent from a caller (*Invite* message) and the first message received from a callee (*Ringing* message). Obviously, the CSD is influenced by the number of nodes that the messages have to traverse, and by the time spent at each node for the message processing. According to ETSI [9] and ITU-T [12], CSD value should not exceed 400-500 ms.

The second part of the problem concerns the availability aspects, namely, the ability of a system to provide a service despite the occurrence of failures. Precisely, the availability requirement of a technological system (the cIMS in our case) can be measured in terms of number of "nines" that can be translated into a maximum annual downtime (MAD): for instance, a four nines availability (namely, a probability of 0.9999 that the system is working) corresponds to a MAD of 52 minutes and 36 seconds, whereas a five nines availability (namely, a probability of 0.99999 that the system is working) corresponds to a MAD of 5 minutes and 15 seconds. This latter requirement, often known as *high availability*, is usually included into the Service Level Agreement offered by all modern telecom infrastructures [13].

A joint performance and availability assessment allows to pinpoint the optimal-redundancy cIMS setting that: *i*) exhibits the best performance (with CSD below a critical threshold), *ii*) is able to satisfy a given availability requirement (e.g., the five nines); *iii*) has the minimum cost, namely, the minimum number of redundant elements.

The present paper represents a substantial extension over our previous conference paper [14], both from a methodological and an experimental perspective as detailed in the final part of Section II, with key contributions summarized as follows:

- To characterize performance, we propose a *G/G/m* queueing model of a cIMS node by exploiting the Krämer/Lagenbach-Belz approximation.
- To characterize availability, we: *i*) exploit the Multi-State System (MSS) formalism to model a cIMS node; and *ii*) devise an ad hoc algorithm to find the minimum-redundancy cIMS setting which fulfills the performance and availability requirements.
- We put in place an extensive experiment through which we are able: to *i*) estimate some system parameters (e.g., service times, repair rates); and to *ii*) perform a dedicated sensitivity analysis.

The remainder of the paper is organized as follows. Section II proposes an overview of related work both concerning performance and availability aspects, including our previous conference work [14], where we highlight the differences with respect to the work presented here. In Section III we provide an architectural perspective of cIMS and of the *Clearwater* testbed, and we introduce the concept of containerized function (CF), the basic modeling element of a cIMS node. Section IV analyzes in depth the state-space model of a CF through the MSS formalism. In Section V we provide analytical details about the adopted *G/G/m* queueing model and the pertinent approximations. Section VI introduces the PE-MUGF algorithm to evaluate the optimal cIMS settings. Section VII presents the experimental part along with original results, and Section VIII concludes the paper along with some future research hints.

II. Related Work

Recently, there have been many efforts and attempts to model both performance and availability aspects concerning the service function chains in the realm of network management [15], [16], [17], [18]. Thus, we find it more convenient to keep separated the two aspects, so as to highlight differences and improvements of our proposal with respect to the existing technical literature.

A. Performance Issues

As regards the performance issues, a large part of literature is aimed at characterizing the effect of delays introduced by single nodes belonging to a chain, impacting unavoidably the overall SFC delay. Relevant studies include: a performance evaluation of chained services through a solution strategy named MaxZ [19]; a mathematical formulation of an optimization problem which takes into account the delay guarantees provided by SFCs [20]; an SFC orchestration solution with the objective of minimizing the cost of the composing virtual network functions (VNFs) [21]; service rate control problems in SFC requests scheduling [22]; a technique (named Network Queueing Assessment) to detect bottlenecks in SFCs based on the network queue occupation [23]; a solution (called eRESERV) to evaluate performance of SFCs [24]; a delay-based performance of SFC along with the problem of CPU allocation [25]; a reliable SFC placement problem in softwarized 5G networks [26].

All the aforementioned works adopt *M/M/*1 queueing models to characterize the elements belonging to a service chain. On the one hand, such models offer the comfort of a mathematical closed form amenable to be managed. On the other hand, they could fail to represent some real-world situations since they assume predefined (exponential) distributions both for inter-arrivals and service times, and assume single-server nodes, where in many cases each node could be able to manage more than one service request at a time.

Other studies [27], [28], [29], [30] admit the presence of multi-server nodes to model VNFs, but they adopt *M/M/m* queueing systems that restrict the generality of the model. Similarly, previous studies [14], [31] characterized each SFC node in terms of an *M/G/m* queueing model.

Differently from all the mentioned works, we propose a *G/G/m* queueing model to characterize each node of the SFC. It represents the most general case, which can deal with realistic use-cases where classic assumption of exponential distributions (both for inter-arrival and service times) is inaccurate.

B. Availability Issues

As regards the availability aspects of SFCs, the technical literature proposes a number of useful techniques to optimize the redundancy. For example, Petri-based formalisms provide a compact way to model the availability of chained structures through the analysis of the state changes. Among the works which exploit such a formalism we find: [32], including a VNF migration strategy where the underlying SFC has been modeled according to the Petri formalism; stochastic

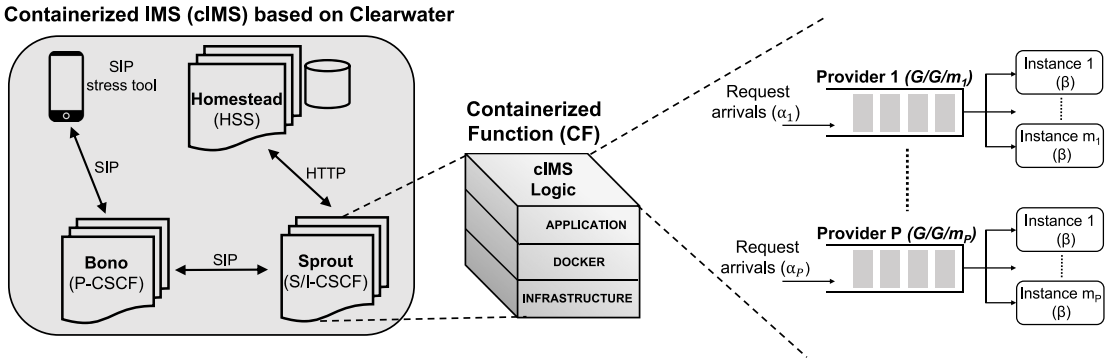


Fig. 1. Architecture overview: the testbed based on the *Clearwater* platform (left panel); the 3-layer Containerized Function (CF) constituting a cIMS node (middle panel); the *G/G/m* queueing model for each provider *p* (right panel).

Petri networks (SPNs) have been exploited in [33] to set an automatic method useful to evaluate the availability of SFCs; authors in [34] propose a comparative analysis of different SFC configurations exploiting the stochastic reward networks (SRNs) formalism, a variant of classic stochastic Petri networks with a reward function; SRN have been used also in [35] to characterize from an availability view point homogeneous and heterogeneous deployments of SFCs; stochastic activity networks (SANs) have been adopted in [36] and [37] to assess the availability of an end-to-end NFV-aware network service; generalized stochastic Petri networks (GSPNs) have been employed in [38] to model availability problems in data centers in charge of managing SFCs.

The compactness of the described methods represents both an advantage and a disadvantage at the same time: they are benefiting since provide high-level expressiveness during the modeling stage, but, make it difficult to access analytical details that we exploit in our MUGF-based approach. Another limitation is that Petri-based techniques typically require specific software tools such as SHARPE [39], SPNP [40], TimeNET [41], WebSPN [42]. The Universal Generating Function (UGF) technique (the non-multidimensional version) has also been exploited to deal with availability aspects of virtualized environments [43], [44]. A limitation is that such a method is not suitable for the application to a multi-provider environment, which we are able to address through the proposed multidimensional UGF technique.

We conclude this section by pinpointing the main methodological and experimental advances over [14]. As to the former: *i*) we adopt a more general queueing model for a cIMS node, namely *G/G/m*, where: the inter-arrivals of cIMS sessions (the first *G*) are assumed to be Gamma-distributed with different shape parameters accounting for a broader set of possibilities; the generic service times (the second *G*) are estimated through experiments; and *m* containerized instances are managed by *P* providers which share a cIMS node; *ii*) we introduce a formalization of series and parallel structures useful to mathematically justify the Multidimensional Universal Generating Function (MUGF) method; *iii*) we devise an effective *ad hoc* algorithm named PE-MUGF (Performability Evaluation through MUGF) to search for the minimum-redundancy cIMS setting which meets given performance and availability requirements. On the experimental side, we conduct an extensive campaign based on *Clearwater*, a container-based IMS deployment, through which we are able to: *i*) obtain *on-field* estimates of relevant model parameters (e.g., service times, repair rates); *ii*) elaborate on possible variations of redundant cIMS configurations; *iii*) conduct a dedicated sensitivity analysis focused on some critical parameters.

III. Motivating Example: Multi-Provider cIMS

As an exemplary SFC infrastructure we consider a container-based version of IP Multimedia Subsystem (cIMS) realized through the open-source project *Clearwater* release 130 [45]. The leftmost panel in Fig. 1 shows the nodes that we have implemented in our testbed:

- Proxy-CSCF (P-CSCF)[1]: the ingress point of the cIMS architecture which exposes its SIP[2]-based interface to the external world. The corresponding Clearwater name is *Bono*.
- Serving-CSCF (S-CSCF): is responsible for the multimedia sessions control, including authentication and routing procedures. The corresponding Clearwater name is *Sprout/S*.
- Interrogating-CSCF (I-CSCF): it enables IMS requests to be routed towards the correct S-CSCF. The corresponding Clearwater name is *Sprout/I* and is co-located with *Sprout/S*.
- Home Subscriber Server (HSS): it stores information about IMS subscribers (including authentication keys). The corresponding Clearwater name is *Homestead*.

In line with the decoupling logic of softwarized infrastructures, each cIMS node is realized through a 3-layer structure that we call Containerized Function (CF) shown in the middle panel of Fig. 1. The CF upper layer (*application* layer) hosts the specific cIMS logic (e.g., Proxy, Serving, etc.) embodied into containers; the middle layer (*docker* layer) provides support for containers and is realized through the popular docker daemon engine; the lower layer (*infrastructure* layer) models

[1]Call Session Control Function: a control functionality useful to manage the multimedia sessions, and distributed across different IMS nodes.

[2]Session Initiation Protocol: the signaling protocol used within IMS-based architectures.

all the physical parts including CPU, power supplies, etc. It is useful to disclose that a cIMS node can be made of one or more redundant CFs connected in parallel to improve the availability, as will be detailed in Section VI. Finally, the rightmost panel of Fig. 1 shows that each CF can be shared by P different providers. Each provider p $(p = 1, \ldots, P)$, modeled as a $G/G/m_p$ queue, represents a set of containerized instances (briefly, instances) each of which able to manage the cIMS requests in queue.

IV. A Multi-State System Approach for the Availability Modeling of a Containerized Function

The Multi-State System (MSS) formalism was introduced to overcome the limitation arising from the binary models [46] where, from an availability perspective, a system can be characterized according to two extreme cases: perfect functioning or complete failure. Conversely, in many real-life situations, the systems and their components can assume a certain range of *performance rates* between the two aforementioned extreme cases [47]. By applying the MSS modeling to service function chains, it is possible to: *i*) evaluate the performance rates of the single components (e.g., the nodes) ruled by failures and repair operations, and *ii*) employ the MUGF to recombine, through simple algebraic operations, the performance rates of single components and derive a macroscopic performance model of the whole chain.

Figure 2 represents the MSS model of a single CF, where each performance rate can be mapped into a given *state*, providing information about the operating (working or failed) condition of a specific component. A failed component is indicated by 0 and a working (or repaired) component is indicated by 1.

In Fig 2, each state is identified by a P-dimensional vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_P) \in \prod_{p=1}^{P}\{0, \ldots, s_p\}$ being $\sigma_p \in \{0, 1, \ldots, s_p\}$ the number of working containerized instances belonging to provider p whose maximum value is s_p. For example, the top-most state in Fig. 2, namely $(s_1, s_2, \ldots, s_P)$, is the most favourable state (all providers with all instances working). In contrast, state $(s_1 - 1, s_2, \ldots, s_P)$ indicates that provider 1 works with one instance less. We also have 3 special states: $(0, 0, \ldots, 0)_C$ indicating that all providers have no working containerized instances; $(0, 0, \ldots, 0)_D$ indicating that docker layer is no longer working (e.g., due to a bug into docker manager), thus, causing the immediate faults of all containerized instances on top; $(0, 0, \ldots, 0)_I$ indicating that infrastructure layer is no longer working (e.g., due to a power interruption), thus, causing the faults of both instances and docker layers.

The inter-arrival failures are treated as independent and identically distributed (iid) random variables, and, more precisely, as exponentially distributed random variables with parameter λ, whereas the times taken for repair are treated as exponentially distributed random variables with parameter μ [48], [49]. For example, by starting from a completely working system with state $(s_1, s_2, \ldots, s_P)$, the failure action observed when one instance of provider 1 fails is ruled by

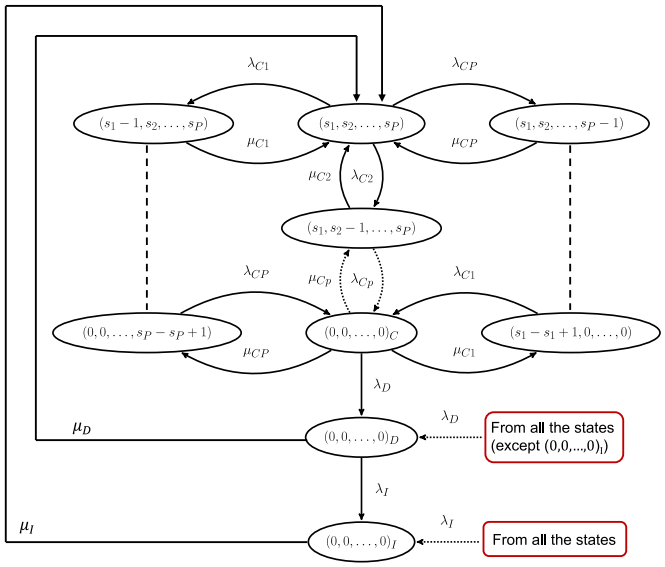


Fig. 2. Multi-State System (MSS) model of a Containerized Function.

failure rate λ_{C1} and brings the system towards the state $(s_1 - 1, s_2, \ldots, s_P)$. In contrast, the system comes back into the completely working state when the failed instance of provider 1 gets repaired with parameter μ_{C1}. Remarkably, when the docker layer fails, each state of the system (excepting for state $(0, 0, \ldots, 0)_I$) is forced to reach the state $(0, 0, \ldots, 0)_D$ with failure rate λ_D. Likewise, when the infrastructure layer fails, each state of the system is forced to reach the state $(0, 0, \ldots, 0)_I$ with failure rate λ_I. Please also note that, as usual in real-world systems, both repairs of docker and infrastructure layers conclude with a recover of the whole system with repair rates μ_D and μ_I, respectively.

The overall state space of the MSS can be defined as $\omega = \omega_C \times \omega_D \times \omega_I$, where:

- $\omega_C = \prod_{i=1}^{P}\{0, 1, \ldots, s_p\}$ represents the state space of the application layer (with containerized instances) of all providers;
- $\omega_D = \{0, 1\}_D$ represents the state space of the docker layer, where 0 indicates the docker failure condition and 1 indicates the docker working condition;
- $\omega_I = \{0, 1\}_I$ represents the state space of the infrastructure layer, where 0 indicates the infrastructure failure condition and 1 indicates the infrastructure working condition.

At this point, to completely characterize the MSS, we need to formally define two descriptors. The first one is the performance rate $r_{p,\boldsymbol{\sigma}} = \gamma\sigma_p$ being γ the so-called *serving capacity*, namely, the number of cIMS requests that containerized instances of provider P can concurrently manage when σ_p instances are currently available. Thus, $\boldsymbol{r}_{\boldsymbol{\sigma}} = (\boldsymbol{r}_{1,\boldsymbol{\sigma}}, \ldots, \boldsymbol{r}_{P,\boldsymbol{\sigma}})$ represents the stochastic vector containing all performance rates included in the set

$$\mathcal{R} = \left\{ \boldsymbol{r}_{\boldsymbol{\sigma}} \;\middle|\; \boldsymbol{\sigma} \in \prod_{p=1}^{P}\{0, \ldots, s_p\} \right\} \cup \{(0, \ldots, 0)_D, (0, \ldots, 0)_I\}. \quad (1)$$

The second descriptor is the *structure function* ψ which provides a mapping between all possible combinations of states of

the system components $\omega = (\sigma, z_D, z_I)$ and the whole system state. Thus, we have:

$$\psi : \omega \rightarrow \left\{ \boldsymbol{\sigma} \;\middle|\; \boldsymbol{\sigma} \in \prod_{p=1}^{P} \{0, \ldots, s_p\} \right\} \cup \{(0, \ldots, 0)_D, (0, \ldots, 0)_I\}, \quad (2)$$

where: $\psi(\boldsymbol{\sigma}, z_D, z_I) = \boldsymbol{\sigma}$ for $z_D = 1$ and $z_I = 1$ (namely, docker and infrastructure layers are both working), $\psi(\boldsymbol{\sigma}, z_D, z_I) = (0, \ldots, 0)_D$ for $z_D = 0$ and $z_I = 1$ (namely, the docker layer is failed), and $\psi(\boldsymbol{\sigma}, z_D, z_I) = (0, \ldots, 0)_I$ for $z_D = 0$ and $z_I = 0$ (namely, docker and infrastructure layers are failed).

We can conclude that the MSS performance rate can be expressed as the vector stochastic process $\boldsymbol{R}(t) = (R_1(t), \ldots, R_P(t)) = \gamma \cdot \psi(\boldsymbol{Z}(t), Z_D(t), Z_I(t))$, where: $\boldsymbol{Z}(t) = (Z_1(t), \ldots, Z_P(t))$ is a ω_C-valued process representing the failure/repair condition of the application layer, and $Z_D(t)$ [$Z_I(t)$] is a ω_D-valued [ω_I-valued] process representing the failure/repair condition of the docker [infrastructure] layer.

Being built on the MSS of Fig. 2, $\boldsymbol{R}(t)$ is a process having a finite number of states amounting to

$$S = \prod_{p=1}^{P} (s_p + 1) + 2, \quad (3)$$

and is *irreducible*, meaning that each state can be reached by each other state. Moreover, since all the λ and μ parameters do not change with time, the process $\boldsymbol{R}(t)$ is an ergodic Continuous-Time Markov Chain (CTMC) whose probability vector $\boldsymbol{\pi}(t)$ can be obtained by solving:

$$\frac{\mathrm{d}\boldsymbol{\pi}(t)}{\mathrm{d}t} = \boldsymbol{\pi}(t)\mathbf{Q}, \quad (4)$$

being $\mathbf{Q}$ the infinitesimal generator matrix [50]. We have also to consider the normalization condition $\sum_{\boldsymbol{\sigma}} \pi_{\boldsymbol{\sigma}}(t) = 1$, where $\pi_{\boldsymbol{\sigma}} = \lim_{t\to\infty} \pi_{\boldsymbol{\sigma}}(t) = \lim_{t\to\infty} \Pr\{\boldsymbol{R}(t) = \boldsymbol{r}_{\boldsymbol{\sigma}}\}$. Since we are interested in the steady-state behavior of the MSS, we can safely conclude that the set $\{\pi_{\boldsymbol{\sigma}}, \boldsymbol{r}_{\boldsymbol{\sigma}}\}$ uniquely describes the steady-state behavior of the containerized function.

V. Performance of a Containerized Function Through a $G/G/m_p$ Queue-Based Model

As shown in the rightmost panel of Fig. 1, each provider P at the application layer is represented through a queueing system made of a set of software instances able to manage the cIMS sessions. In order to treat the problem in the most general way possible, we propose a $G/G/m_p$ $(p = 1, \ldots, P)$ queue modeling of a CF. For this model we have: generic inter-arrival times (the first G), generic service times (the second G), and a number of finite m_p containerized instances per provider p. Before delving into details of such a queueing model we want to highlight an important fact. In principle, the number of m_p instances in charge of managing multimedia sessions can vary across the time since, as specified in the previous section, some of them may fail, thus we would have a $G/G/m_p(t)$ queueing model. Having also defined $\boldsymbol{R}(t)$ as the performance rate of the MSS, the queueing model can be denoted by $G/G/R_p(t)$ so as to stress the dependency from a particular state.

TABLE I
Model Parameters

Parameter	Description	Value
$1/\beta_P$	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
$V_S^{(P)}$	P-CSCF coefficient of variation	0.7538
$V_S^{(S)}$	S-CSCF coefficient of variation	0.9826
$V_S^{(I)}$	I-CSCF coefficient of variation	0.5581
$V_S^{(H)}$	HSS coefficient of variation	0.4631
$1/\mu_C$	mean time to container repair	30 s
$1/\mu_D$	mean time to docker repair	60 s
$1/\mu_I$	mean time to infrastructure repair	5 min
$1/\lambda_C$	mean time to container failure	1258 hours
$1/\lambda_D$	mean time to docker failure	2516 hours
$1/\lambda_I$	mean time to infrastructure failure	60000 hours

Interestingly, we note that failure time scale completely dominates queueing time scale since failure times are in the order of thousand of hours, whereas service times are in the order of few milliseconds (see Table I further ahead). This condition leads to a *decoupling* of the two time scales that, as well explained in [51], allows to neglect the transient effects of the dominated times scale (the queueing times scale in our case). In other words, the steady-state condition of queues are achieved much faster than the steady-state condition of faults. Thus, we can safely assume a $G/G/R_p$ model, where the time dependency is "absorbed". This notwithstanding, when we need to stress the time dependency from a particular state, we will occasionally use the $G/G/R_p(t)$ notation.

At this point, it is useful to recall that the CSD performance indicator is directly related to the amount of time that a cIMS request spends at each CF waiting to be processed. For example, in our IMS case study, the CSD is given by the total latency across the four stages (i.e., the CFs) in the service chain (P-CSCF, S-CSCF, I-CSCF, HSS). Intuitively, higher sojourn times at each CF imply higher CSD that, in turn, implies worse performance.

In line with these considerations, we characterize the average sojourn times of cIMS requests which depend on the provider p and on the particular state σ. Remarkably, $G/G/m_p$ queueing systems do not admit analytical closed forms, thus, some approximating formulas are required. To address this issue, we introduce an *equivalent* $M/M/m_p$ model with Poisson inter-arrivals with rate α, and exponential service rates with mean β. According to the classic queueing theory [52], we can express the mean sojourn time at a single CF as:

$$\delta_{p,\sigma} = q_{p,\sigma} + \frac{1}{\beta}, \quad (5)$$

where: $q_{p,\sigma}$ represents the mean waiting time spent by a cIMS request in the queue of provider p in state σ, and $1/\beta$ is the mean service time spent by the cIMS request to be processed. This latter quantity can be experimentally estimated (see numeric values in Table I further ahead) for each node.

In contrast, $q_{p,\sigma}$ is a random quantity which can be obtained by approximating the corresponding mean waiting time of the equivalent $M/M/m_p$ queueing system with:

$$q_{p,\sigma} \approx \frac{\pi_m}{\beta(1-\rho)} \frac{V_A^2 + V_S^2}{2m} \mathcal{F}, \tag{6}$$

where: π_m is the steady-state probability of the equivalent $M/M/m_p$ queueing system [52], $\rho = \alpha/(\beta \cdot m)$ is the *utilization factor*, V_A and V_S are the coefficients of variation $(\sigma(\cdot)/E(\cdot))$ for inter-arrival and service times, respectively, and $\mathcal{F}$ is a correction factor which implements the Krämer/Lagenbach-Belz approximation formula [53], [54], [55] for a $G/G/m_p$ queueing model obeying to the following relation:

$$\mathcal{F} = \begin{cases} \exp\left[\frac{2(\rho-1)(1-V_A^2)^2}{3\pi_m(V_A^2+V_S^2)}\right], \leq V_A \leq 1, \\ \exp\left[\frac{(\rho-1)(V_A^2-1)}{V_A^2+4V_S^2}\right], \qquad V_A > 1. \end{cases} \tag{7}$$

Thus, by substituting (6) in (5), we obtain the mean sojourn time per CF modeled as a $G/G/m_p$ queueing system. Moreover, since the time spent by a cIMS request also depends on the particular state reached by the MSS in Fig. 2, we can easily define the structure function $\psi_\Delta : \omega \rightarrow \{\mathbb{R}^+ \cup \{+\infty\}\}^P$ specialized to the mean sojourn times. Similarly to the definition introduced in (2), we have that $\psi_\Delta(\boldsymbol{\sigma}, z_D, z_I) = (\delta_{1,\sigma}, \ldots, \delta_{P,\sigma})$ for $z_D = 1$ and $z_I = 1$, $\psi_\Delta(\boldsymbol{\sigma}, z_D, z_I) = (\infty, \ldots, \infty)_D$ for $z_D = 0$ and $z_I = 1$ (when docker layer is not working, we have infinite delay), and $\psi_\Delta(\boldsymbol{\sigma}, z_D, z_I) = (\infty, \ldots, \infty)_I$ for $z_D = 0$ and $z_I = 0$ (when infrastructure and docker layers are not working, we have infinite delay).

Remarkably, the structure function ψ_Δ is useful to characterize the mean sojourn time in each possible state through the vector stochastic process $\boldsymbol{\Delta}(t) = (\Delta_1(t), \ldots, \Delta_P(t)) = \psi_\Delta(\boldsymbol{\sigma}, Z_D(t), Z_I(t))$. Moreover, similarly to the $\boldsymbol{R}(t)$ process, also $\boldsymbol{\Delta}(t)$ is an ergodic CTMC process, where the set of pairs $\{\pi_{\boldsymbol{\sigma}}, \boldsymbol{\delta_\sigma}\}$ determines the steady-state performance behavior of a CF in terms of mean sojourn times.

VI. Performability of a Service Chain: The MUGF Approach

Since we are dealing with a chain of nodes where each node is made of replicated CFs for availability purposes, we want to stress that: *i*) a *series connection* implies that the whole chain is functioning when each node $n \in N$ is functioning, where $N = \{$P-CSCF, S-CSCF, I-CSCF, HSS$\}$ is the set of cIMS nodes; *ii*) a *parallel connection* implies that each node n is made of redundant CFs. Specifically, $\text{CF}^{(n,\ell)}$ represents the parallel CF ℓ $(\ell = 1, \ldots, L_n)$ associated to node n. Since we assume that all CFs composing a node have to share the load among them, the redundancy is supposed to be "hot standby" (this working hypothesis is also known as flow dispersion hypothesis [47]). The resulting series/parallel structure is shown in Fig. 3. Such a model is meant to capture a high-level architectural perspective, by not considering synchronization problems nor links availability (links are supposed to be always-on). At this point, we evaluate the mean CSD introduced by a chain, denoted by $\boldsymbol{\Delta}^c(t)$, through the definition of two operators: the *series structure function* and the *parallel structure function*. We find it more convenient to start by defining the latter operator.

Proposition 1 (Parallel Structure Function): Let $\boldsymbol{\Delta}^{(n)}(t)$ be the vector stochastic process containing the mean sojourn time introduced by node n. Once defined the *parallel structure function* $\psi_{par} : \omega^{L_n} \rightarrow \mathbb{R}^P \cup (+\infty, \ldots, +\infty)$, the mean sojourn time introduced by node n is:

$$\boldsymbol{\Delta}^{(n)}(t) = \psi_{par}\Big(\boldsymbol{Z}^{(n,1)}(t), Z_D^{(n,1)}(t), Z_I^{(n,1)}(t), \ldots, \boldsymbol{Z}^{(n,L_n)}(t), Z_D^{(n,L_n)}(t), Z_I^{(n,L_n)}(t)\Big) = \Big(\Delta_1^{(n)}(t), \ldots, \Delta_P^{(n)}(t)\Big), \tag{8}$$

where $\Delta_p^{(n)}(t)$ is the stochastic process describing the $G/G/R_p^{(n)}(t)$ queue, being $R_p^{(n)}(t) = \sum_{\ell=1}^{L_n} R_p^{(n,\ell)}(t)$, and $\boldsymbol{Z}^{(n,L_n)}(t)$, $Z_D^{(n,L_n)}(t)$, $Z_I^{(n,L_n)}(t)$ denoting, respectively, the ω_C-valued, ω_D-valued, ω_I-valued failure/repair processes of the application layer, docker layer, infrastructure layer of the $\text{CF}^{(n,\ell)}$, $\ell = 1, \ldots, L_n$.

Since the call flow traverses the chain in series, the overall mean CSD $\boldsymbol{\Delta}^c(t)$ can be obtained as the sum of mean CSDs introduced by each single node. Such a quantity can be evaluated by introducing the following:

Proposition 2 (Series Structure Function): We define a *series structure function* $\psi_{ser} : \omega^{\Sigma_n L_n} \rightarrow \mathbb{R}^P \cup (+\infty, \ldots, +\infty)$. The overall mean CSD $\boldsymbol{\Delta^c}(t) = (\Delta_1^c(t), \ldots, \Delta_P^c(t))$ introduced by the chain is given by:

$$\begin{aligned} \boldsymbol{\Delta}^c(t) = \sum_n \boldsymbol{\Delta}^{(n)}(t) = \psi_{ser}\Big(&\boldsymbol{Z}^{(P,1)}(t), Z_D^{(P,1)}(t), Z_I^{(P,1)}(t), \ldots, \\ &\boldsymbol{Z}^{(P,L_P)}(t), Z_D^{(P,L_P)}(t), Z_I^{(P,L_P)}(t), \ldots, \\ &\boldsymbol{Z}^{(H,1)}(t), Z_D^{(H,1)}(t), Z_I^{(H,1)}(t), \ldots, \\ &\boldsymbol{Z}^{(H,L_H)}(t), Z_D^{(H,L_H)}(t), Z_I^{(H,L_H)}(t)\Big) \overset{(8)}{=} \\ \sum_n \psi_{par}\Big(&\boldsymbol{Z}^{(n,1)}(t), Z_D^{(n,1)}(t), Z_I^{(n,1)}(t), \ldots, \\ &\boldsymbol{Z}^{(n,L_n)}(t), Z_D^{(n,L_n)}(t), Z_I^{(n,L_n)}(t)\Big). \end{aligned} \tag{9}$$

From (8) we can derive the steady-state mean distribution of sojourn times for node n, viz.,

$$\left\{\pi_{\boldsymbol{\sigma}}^{(n)}, \boldsymbol{\delta}_{\boldsymbol{\sigma}}^{(n)}\right\}, \tag{10}$$

where: $\boldsymbol{\delta}_{\boldsymbol{\sigma}}^{(n)} = (\delta_{1,\boldsymbol{\sigma}}^{(n)}, \ldots, \delta_{P,\boldsymbol{\sigma}}^{(n)})$ is the mean sojourn times vector of node n in state $\boldsymbol{\sigma}$, and $\pi_{\boldsymbol{\sigma}}^{(n)} = \lim_{t\rightarrow\infty} \Pr\{\boldsymbol{\Delta}^{(n)}(t) = \boldsymbol{\delta}_{\boldsymbol{\sigma}}^{(n)}\}$ the corresponding limiting probability. Likewise, from (9) we can derive the steady-state mean CSD distribution of the whole chain, viz.,

$$\{\pi_{\boldsymbol{\sigma}}^c, \boldsymbol{\delta}_{\boldsymbol{\sigma}}^c\}, \tag{11}$$

where: $\boldsymbol{\delta}_{\boldsymbol{\sigma}}^c = (\delta_{1,\boldsymbol{\sigma}}^c, \ldots, \delta_{P,\boldsymbol{\sigma}}^c)$ is the mean sojourn times vector of system in state $\boldsymbol{\sigma}$, and $\pi_{\boldsymbol{\sigma}}^c = \lim_{t\rightarrow\infty} \Pr\{\boldsymbol{\Delta}^c(t) = \boldsymbol{\delta}_{\boldsymbol{\sigma}}^c\}$ the corresponding limiting probability.

In order to solve the proposed model in a computationally-efficient way, we use an approach based on the Universal Generating Function (UGF), also known as *u*-function [56]. The UGF is a hierarchical technique to compute steady-state

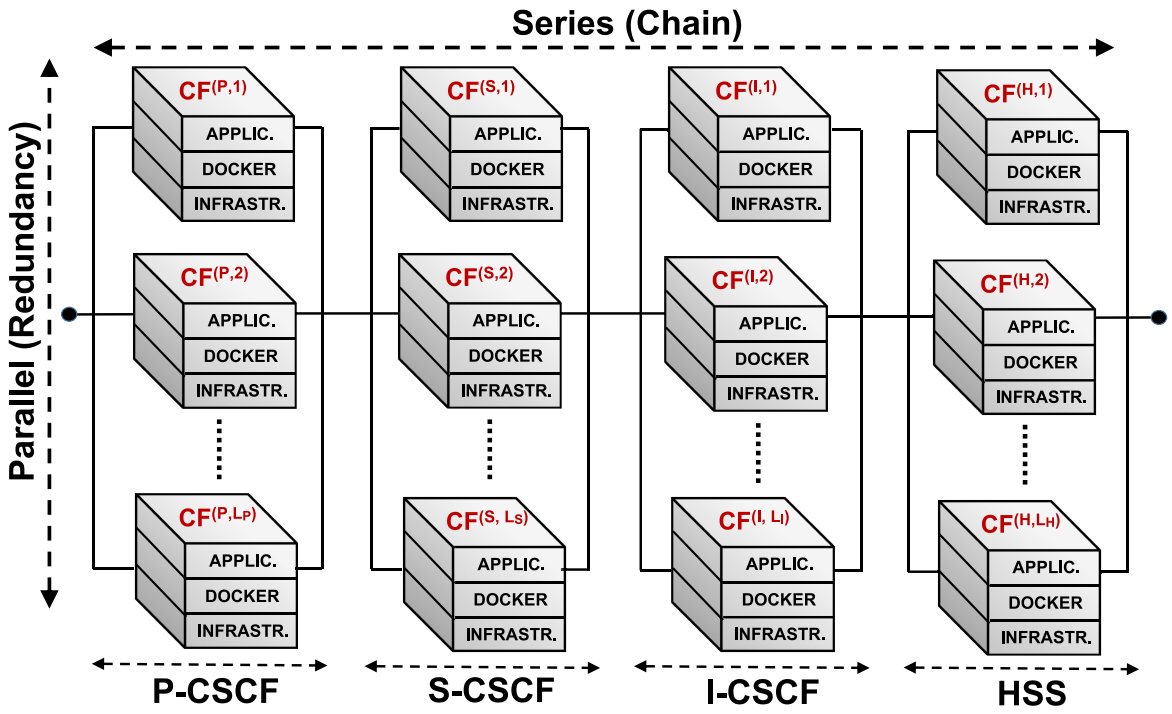


Fig. 3. Series/Parallel cIMS architecture: each node is connected in series and is made of a number redundant CFs connected in parallel.

performance distributions of complex MSSs characterized by series/parallel interconnections among the components. The UGF of the steady-state performance metric Y, whose distribution is given by the set of pairs $\{\pi_j, y_j\}$, is the polynomial-shape function

$$u(z) = \sum_j \pi_j z^{y_j}, \tag{12}$$

being y_j the j-th performance rate, and π_j the corresponding steady-state probability. The UGF of the whole system (i.e., the combination of multiple CFs in series/parallel) can be obtained by combining the UGFs of individual CFs through simple sums and products. Thus, the steady-state probabilities for the entire system can be obtained from the combined UGF.

Since we deal with multiple service providers in the system, we adopt a multidimensional extension of (12) dubbed MUGF [31] along with (8) and (9). Thus, we are able to obtain the MUGF of the whole cIMS chain where: *i*) 4 nodes are connected in series, *ii*) each node is made of ℓ replicated CFs to guarantee redundancy, *iii*) each CF is shared among different P providers and can be in a particular state $\boldsymbol{\sigma}$. In summary, we can write the MUGF $u^c(\boldsymbol{z})$ of the whole chain as the product of MUGFs of single nodes, viz.

$$\begin{aligned} u^c(\boldsymbol{z}) &= \prod_n \underbrace{\left[\sum_{\boldsymbol{\sigma}\in\omega^{Ln}} \pi_{\boldsymbol{\sigma}}^{(n)} \prod_{p=1}^{P} z_p^{\delta_{p,\boldsymbol{\sigma}}^{(n)}}\right]}_{\text{MUGF of node } n\,(u^{(n)}(z))} \\ &= \sum_{\boldsymbol{\sigma}\in\omega^{\text{tot}}} \pi_{\boldsymbol{\sigma}}^c \prod_{p=1}^{P} z_p^{\delta_{p,\boldsymbol{\sigma}}^c}, \end{aligned} \tag{13}$$

with $\omega^{\text{tot}} = \prod_n \omega^{L_n}$. In practice, $u^c(\boldsymbol{z})$ represents a polynomial-shape function in $z_1, \ldots, z_P$ indeterminates, where each term corresponds to the mean CSD vector $\boldsymbol{\delta}_{\boldsymbol{\sigma}}^c$ (exponents of $\boldsymbol{z}$), whereas its steady-state probability $\pi_{\boldsymbol{\sigma}}^c$ is the pertinent coefficient. Such quantities can be used to calculate the steady-state availability of the whole service chain as explained below.

First, we denote by $\boldsymbol{S}$ a particular cIMS setting where each node n is made of a number of redundant CFs ℓ (ℓ=1, …, L_n). Yet, we denote by $\boldsymbol{\delta}^* = (\delta_1^*, \ldots, \delta_P^*)$ a P-dimensional vector which contains the maximum steady-state tolerated values of mean CSD. Thus, we define the steady-state availability of a particular cIMS setting $\boldsymbol{S}$ as

$$A^c(\boldsymbol{\delta}^*, \boldsymbol{S}) = \sum_{\boldsymbol{\sigma}\in\omega^{\text{tot}}} \pi_{\boldsymbol{\sigma}}^c \cdot \mathbf{1}\big(\delta_{p,\boldsymbol{\sigma}}^c \le \delta_p^*, \forall p = 1, \ldots, P\big), \tag{14}$$

where $\mathbf{1}(\cdot)$ is a function which amounts to 1 if condition holds true and 0 otherwise. We note that $\pi_{\boldsymbol{\sigma}}^c$ and $\delta_{p,\boldsymbol{\sigma}}^c$ in (14) are directly derived from the MUGF expression (13).

We also stress the fact that (14) provides the steady-state availability of a generic cIMS setting $\boldsymbol{S}$, but we are interested to find the steady-state availability of the setting with the minimum cost, namely, with the minimum number of redundant CFs.

Accordingly, denoting by $E^{(n,\ell)}$ the cost (or expenditure) of the $\ell-$th CF belonging to node n, we can define the cost of a cIMS setting $\boldsymbol{S}$ as $E^c(\boldsymbol{S}) = \sum_n \sum_{\ell=1}^{L_n} E^{(n,\ell)}$. In summary, we search for the solution of the following optimization problem:

$$\begin{cases} \underset{\boldsymbol{S}\in\mathcal{L}^c}{\text{minimize}} \;\; E^c(\boldsymbol{S}) \\ \text{subject to } A^c(\delta^*, \boldsymbol{S}) \ge A_0, \end{cases} \tag{15}$$

Algorithm 1: PE-MUGF

Input: α, β, V_A, V_S, $\boldsymbol{Q}$, δ^*, L_n, N, P, A_0, $E^{(n,\ell)}$
1 **Function** `BuildSetting` (L_n, N):
2 **if** $N = 0$ **then**
3 **return** {}
4 **else**
5 $\boldsymbol{S}_{N-1} \leftarrow$ `BuildSetting` $(L_n, N-1)$
6 $\boldsymbol{S}_N \leftarrow \emptyset$
7 **for** $s \in \boldsymbol{S}_{N-1}$ **do**
8 **for** $i = 1, 2, \ldots, L_n$ **do**
9 $\boldsymbol{S}_{temp} \leftarrow$ prepose i to s
10 $\boldsymbol{S}_N \leftarrow \boldsymbol{S}_N \bigcup \{\boldsymbol{S}_{temp}\}$
11 **end**
12 **end**
13 **end**
14 Calculate $\boldsymbol{\pi}$ from (4) for $t \to \infty$, and $\boldsymbol{\delta}$ from (5)
15 $u^c(z) = \prod_{n=1}^{N} u^{(n)} = \sum_{\ell=1}^{L_n} \pi_\sigma^c \prod_{p=1}^{P} z_p^{\delta_{p,\sigma}^c}$
16 Evaluate $A^c(\boldsymbol{\delta}^*, \boldsymbol{S}_N)$ through (14)
17 Compute cost per setting $E^c(\boldsymbol{S}_N)$
18 **return**
19 **Output**: Optimal setting $\boldsymbol{S}_N^*$ through (15)

where A_0 is a given availability constraint. Often, $A_0 = 0.99999$ namely, the well-known "five nines" steady-state availability constraint.

A. PE-MUGF Algorithm

Algorithm 1 describes PE-MUGF, an algorithm devised to evaluate all the possible cIMS settings in terms of steady-state availability and cost. Such a choice is due to two reasons: first, it is impossible to pinpoint beforehand the optimal cIMS setting with no knowledge of its composition (in terms of redundant CFs); then, with more cIMS settings, a network provider can make different choices or compare several settings according to customized criteria. Thanks to the MUGF approach, the steady-state availabilities can be computed efficiently even for a large number of combinations. PE-MUGF has been realized with Wolfram Mathematica™ and is available upon request.

The initial line of the algorithm specifies the input parameters. In Section VII, we show how these values can be defined in the context of an experimental use case. Lines 1 – 13 report a function called `BuildSetting` useful to build, in a combinatorial way, all the possible cIMS settings made of N nodes, where each node is made of a maximum number of CFs amounting to L_n. We note in passing that, to highlight the recursion into `BuildSetting` function, we adopt the notation $\boldsymbol{S}_N$ in place of $\boldsymbol{S}$ to indicate that a given setting is made of N nodes. In such a way, except for the case $N = 0$ (line 2), each N-node setting can be obtained by preposing a number i $(i = 1, \ldots, L_n)$ to a $(N-1)$-node setting.

Line 15 is the MUGF per cIMS setting derived as a combination of the MUGF applied per single node. Such an expression allows to evaluate the steady-state availability per setting (line 16). Then, a cost assignment per setting obtained as the sum of costs per CF is performed at line 17, and the optimal cIMS setting is provided as the output (line 19).

As mentioned before, to evaluate the optimal setting $\boldsymbol{S}^*$, the PE-MUGF algorithm must evaluate all the built settings which are a byproduct of the procedure.

From a time complexity perspective, it is useful to notice that the MUGF construction is very fast since it relies on simple algebraic operations such as sums and products. In contrast, the `BuildSetting` function requires more time since it has to build all the possible settings, thus leading the complexity of PE-MUGF to $\mathcal{O}(L_n^N)$. This notwithstanding, for typical values of N and L_n in real-world applications (see Section VII for numerical values), PE-MUGF is reasonably affordable.

Even if not explicitly stated, our evaluation can be easily applied to a single-provider architecture being a special case obtained by posing $P = 1$ in (13), and providing a drastic simplification of the MSS in Fig 2. Indeed, in the single-provider case, we have to take into account only the containerized instances (that can be in working or failed conditions) of the provider under analysis, and the performance rates vectors in (1) reduce to scalars.

VII. Experimental Results

We present results from a complex experiment that incorporates real-world hardware and software technologies. Our testbed is based on the Clearwater IMS that was previously introduced. We deployed the three main Clearwater nodes (Bono, Homestead, Sprout/S-I represented in the leftmost panel of Fig. 1) on three dedicated server machines, each of which equipped with: Intel Xeon™ (16-Core, 1.80 GHz), 64 GB of RAM, 2 SATA HDD each of 500 GB, 1 NetApp Network Storage Array (32 TB of storage and 4GB of SSD cache). The operating system on top of each node is based on Linux kernel 4.4.0 with Docker engine version 19.03.5. All the nodes are connected through an Ethernet network switch supporting a maximum throughput of 1 Gbps. Moreover, an additional node hosts *SIPp*, a SIP stress tool that we use as workload generator.

Now, we find it convenient to split the remainder of this section into three parts: in the first one, we deal with the estimation of parameters to insert into MSS and queueing models (in particular, repair times and service times); in the second one, we describe how to use PE-MUGF to find the best cIMS settings; in the last part we evaluate, through a sensitivity analysis, the robustness of the obtained settings when some critical parameters deviate from their nominal value.

A. Parameters Estimation

Through our testbed we are able to estimate two classes of parameters. The first class pertains to the service times distributions obtained by analyzing the logs of all cIMS nodes. In particular, we have stressed the cIMS architecture with 10000 SIP requests from 10000 subscribers automatically generated via *SIPp*, and we have built the empirical mean service time distributions per node and derived the corresponding mean values. The results are reported into the first part of Table I

expressed as the inverse of service rate $(1/\beta)$. We note that all the mean service times are in the order of 10^{-3} seconds except for the mean service time of the I-CSCF which is in the order of 10^{-2} seconds. We will see experimentally that, in view of (5), such a higher value will adversely affect the availability of the whole cIMS if not adequately contrasted through specific redundancy strategies. From the empirical distributions of service times we are also able to derive the coefficient of variation V_S for each node, being expressed as the ratio between the standard deviation and the mean value calculated from such distributions, whose values are also reported in Table I.

The second class of parameters pertains to the repair (or reboot) times for each layer of a CF. To perform such estimation we have implemented a fault injection routine [57], [58] which automatically injects into a CF three types of faults including: container faults (simulated by I/O exceptions and resource exhaustion to force containers to crash and to reboot), docker faults (simulated by forcing an abrupt termination of the *dockerd* process), infrastructure faults (simulated by a physical machine crash). In total, we have performed 360 fault injection experiments (30 fault injections for 3 layers and for 4 types of CFs). Once a layer restores after a fault, we experimentally measure the pertinent repair time. Specifically, as regards repair times differentiated per layer and per CF type, our experiments reveal how slight differences in repair times for each single layer might arise, as shown in Fig. 4. Such a behavior is obviously due to the technological differences among CFs. For instance, container and docker layers of the HSS-type CF exhibit a slightly longer reboot time (w.r.t. reboot times of container and docker layers of remaining CFs), due to the internal database structure that requires more time to be restored. For the sake of simplicity, in Table I we report only the average values of repair times per layer without specifying the CF type they refer to.

Moreover, through dedicated scalability tests, we have also investigated the behavior of application layer reboot times when multiple containers are deployed on top of a CF. In particular, we have deployed onto a I-CSCF-type CF a number of 16 containers[3] in line with the number of cores in our server machines. Then, we configured the containers to optimize the performance and recovery of the system (following the best practice from real-world systems [59], [60]), by configuring CPU affinity policies to avoid CPU contention between containers, and common-mode failures due to a CPU failure. This configuration makes the recovery time insensitive to the number of containers. As shown in Fig. 5, despite some variability due to shared resources between the containers (e.g., communication with the container manager process), the average recovery time does not significantly vary even if we increase the number of containers. Thus, the estimated container repair time can be reasonably considered constant and scarcely dependable on the number of containers deployed in parallel. The remaining parameters, namely, the mean time to failures per layer $(1/\lambda_C, 1/\lambda_D, 1/\lambda_I)$ have been derived by scientific literature [48], [61], [62], [63].

[3]Similar results were observed on CFs for P-CSCF, S-CSCF, HSS.

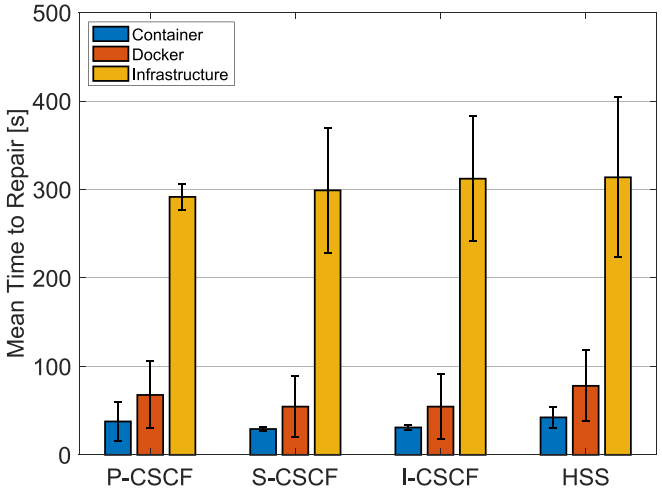


Fig. 4. Mean times to repair per layer and per CF type.

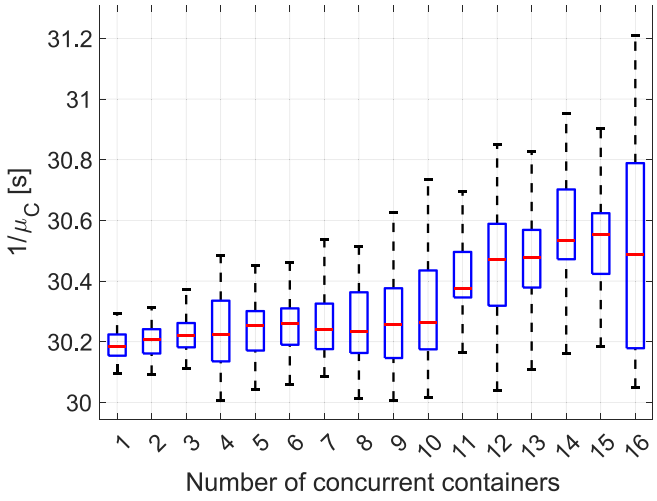
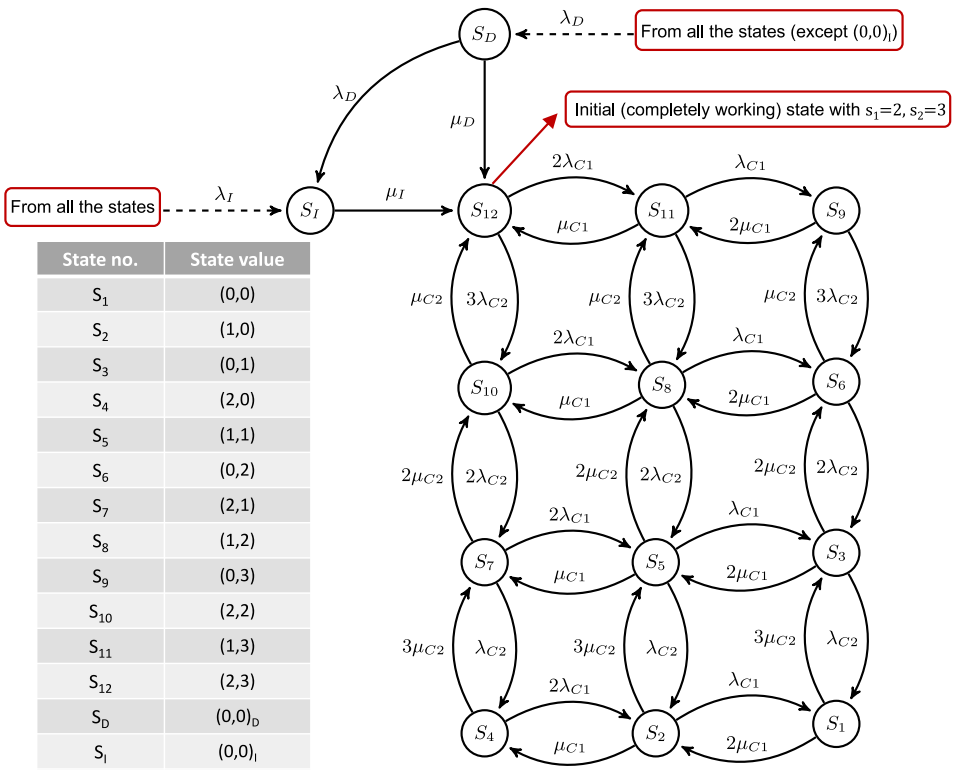


Fig. 5. Stress tests on containers repair times (boxplot representation). The number of concurrent running containers does not dramatically affect $1/\mu_C$.

B. Optimal cIMS Settings

The second part of the experiment aims to demonstrate how to determine the optimal cIMS setting in accordance with (15). We have implemented an exemplary cIMS model where each CF can support $P = 2$ providers and whose corresponding MSS is shown in Fig. 6. We assume that provider 1 is able to support 2 instances, and provider 2 is able to support 3 instances. It is easy to note that, in accordance with (3), the total number of states amounts to 14. Precisely, we have 12 states $(S_1, \ldots, S_{12})$ embodying the failure/working status of each instance and 2 states (S_D and S_I) embodying the failure/working status of docker and infrastructure layers, respectively. Analyzing the MSS in Fig. 6 we can see that, starting from the initial state that is the completely working state S_{12}, we can reach S_{11} with $2\lambda_{C1}$ failure rate, since one of the two working instances of provider 1 may fail. In contrast, when returning into S_{12} from S_{11}, we have a repair rate of μ_{C1} since only one failed instance needs to be repaired. All the pairs $\{\pi_{\boldsymbol{\sigma}}, \boldsymbol{\delta}_{\boldsymbol{\sigma}}\}$ associated to the considered MSS can be found by solving the differential equation (4), along with the



State no.	State value
S_1	(0,0)
S_2	(1,0)
S_3	(0,1)
S_4	(2,0)
S_5	(1,1)
S_6	(0,2)
S_7	(2,1)
S_8	(1,2)
S_9	(0,3)
S_{10}	(2,2)
S_{11}	(1,3)
S_{12}	(2,3)
S_D	$(0,0)_D$
S_I	$(0,0)_I$

Fig. 6. MSS model of the exemplary CF with 2 providers and 14 states.

normalization condition $\sum_{i=1}^{12} \pi_i(t)+p_D(t)+p_I(t)$, and considering the limit $t \to \infty$. The infinitesimal generator matrix $\boldsymbol{Q}$ derived by the MSS in Fig. 6 can be expressed into the compact form (16), shown at the bottom of the page where the

$$\mathbf{Q} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu_I \\
\lambda_I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu_D \\
\lambda_I & \lambda_D & 0 & 2\mu_{C_1} & 3\mu_{C_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\lambda_I & \lambda_D & \lambda_{C_1} & 0 & 0 & \mu_{C_1} & 3\mu_{C_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\lambda_I & \lambda_D & \lambda_{C_2} & 0 & 0 & 0 & 2\mu_{C_1} & 2\mu_{C_2} & 0 & 0 & 0 & 0 & 0 & 0 \\
\lambda_I & \lambda_D & 0 & 2\lambda_{C_1} & 0 & 0 & 0 & 0 & 3\mu_{C_2} & 0 & 0 & 0 & 0 & 0 \\
\lambda_I & \lambda_D & 0 & \lambda_{C_2} & \lambda_{C_1} & 0 & 0 & 0 & \mu_{C_1} & 2\mu_{C_2} & 0 & 0 & 0 & 0 \\
\lambda_I & \lambda_D & 0 & 0 & 2\lambda_{C_2} & 0 & 0 & 0 & 0 & 2\mu_{C_1} & \mu_{C_2} & 0 & 0 & 0 \\
\lambda_I & \lambda_D & 0 & 0 & 0 & \lambda_{C_2} & 2\lambda_{C_1} & 0 & 0 & 0 & 0 & 2\mu_{C_2} & 0 & 0 \\
\lambda_I & \lambda_D & 0 & 0 & 0 & 0 & 2\lambda_{C_2} & \lambda_{C_1} & 0 & 0 & 0 & \mu_{C_1} & \mu_{C_2} & 0 \\
\lambda_I & \lambda_D & 0 & 0 & 0 & 0 & 0 & 3\lambda_{C_2} & 0 & 0 & 0 & 0 & 2\mu_{C_1} & 0 \\
\lambda_I & \lambda_D & 0 & 0 & 0 & 0 & 0 & 0 & 2\lambda_{C_2} & 2\lambda_{C_1} & 0 & 0 & 0 & \mu_{C_2} \\
\lambda_I & \lambda_D & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3\lambda_{C_2} & \lambda_{C_1} & 0 & 0 & \mu_{C_1} \\
\lambda_I & \lambda_D & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3\lambda_{C_2} & 2\lambda_{C_1} & 0
\end{pmatrix} + diag(\mathbf{d}) \tag{16}$$

where:

$$\begin{aligned}
\mathbf{d} = \big[&-\mu_I, -(\lambda_I+\mu_D), -(2\mu_{C_1}+3\mu_{C_2}+\lambda_D+\lambda_I), \\
&-(\lambda_{C_1}+\mu_{C_1}+3\mu_{C_2}+\lambda_D+\lambda_I), -(\lambda_{C_2}+2\mu_{C_1}+2\mu_{C_2}+\lambda_D+\lambda_I), \\
&-(2\lambda_{C_1}+3\mu_{C_2}+\lambda_D+\lambda_I), -(\lambda_{C_1}+\lambda_{C_2}+\mu_{C_1}+2\mu_{C_2}+\lambda_D+\lambda_I), \\
&-(2\lambda_{C_2}+2\mu_{C_1}+\mu_{C_2}+\lambda_D+\lambda_I), \\
&-(2\lambda_{C_1}+\lambda_{C_2}+2\mu_{C_2}+\lambda_D+\lambda_I), -(\lambda_{C_1}+2\lambda_{C_2}+\mu_{C_1}+\mu_{C_2}+\lambda_D+\lambda_I), \\
&-(3\lambda_{C_2}+2\mu_{C_1}+\lambda_D+\lambda_I), -(2\lambda_{C_1}+2\lambda_{C_2}+\mu_{C_2}+\lambda_D+\lambda_I), \\
&-(\lambda_{C_1}+3\lambda_{C_2}+\mu_{C_1}+\lambda_D+\lambda_I), -(2\lambda_{C_1}+3\lambda_{C_2}+\lambda_D+\lambda_I)\big].
\end{aligned} \tag{17}$$

diagonal of the $\boldsymbol{Q}$ has been separately reported in (17), shown at the bottom of the previous page and where all numerical values of parameters are drawn from Table I.

At this point, we need to set the remaining input parameters for the PE-MUGF algorithm. As arrival rates for the two providers we set somehow arbitrarily $\alpha_1 = 100$ s^{-1} and $\alpha_2 = 200$ s^{-1}. Furthermore, we choose one and the same value for the maximum steady-state tolerated mean CSD, i.e., $\delta_1^* = \delta_2^* = \cdots = \delta_P^* = \delta^* = 50$ ms. Such a choice (one order of magnitude less than ETSI values) is justified since, in a local testbed, we neglect all the propagation delay contributions arising in wide geographical networks. As the maximum number of CF redundant replicas L_n, we set 3 for all nodes in the cIMS, and as the availability target A_0, we set the classic five nines 0.99999. Yet, $N = 4$ (we have 4 cIMS nodes).

The CF cost parameter is an arbitrary value that can be customized with no loss of generality. For the sake of simplicity, we assume that each CF has a unitary cost $E^{(n,\ell)} = 1$.

The last parameter to be provided to PE-MUGF algorithm is the coefficient of variation of inter-arrivals V_A that, we recall, in a *G/G/m* queueing system, depends on the particular shape of the inter-arrivals distribution. Differently from the coefficient of variation of service times V_S that we have estimated from the empirical service time distributions, empirical inter-arrivals cannot be simply emulated since they strongly depend on the behavior of users. This notwithstanding, also in line with some credited literature [64], [65], generic inter-arrivals can be modeled by exploiting the versatility offered by the Gamma distribution.

In particular, we employ the distribution $\Gamma(\theta, 1)$, namely a Gamma distribution with a variable shape parameter θ and the scale parameter set to 1 (as suggested in [64]). By varying the shape parameter of the Gamma distribution, we observe different distribution shapes of inter-arrivals, including the exponential distribution obtained for $\theta = 1$ (corresponding to $V_A = 1$) which represents the *M/G/m* queueing model. Figure 7 shows a set of inter-arrival cIMS request distributions corresponding to 7 different values of θ as much as of coefficient of variations. We choose the exponential case as the benchmark (black dashed curve with $\theta = 1$ and $V_A = 1$) and we spanned some values around such a benchmark value. We note that, for $\theta < 1$ the coefficient of variation decreases and the corresponding inter-arrival distributions stretches out. In contrast, for $\theta > 1$ the coefficient of variation increases and, as expected according to (6), this increase will adversely affect the availability as we will numerically show in a while.

We run PE-MUGF as many times as V_A values. For the sake of simplicity, let us start with the reference value $V_A = 1$. As mentioned before, PE-MUGF returns the optimal cIMS setting (namely, the one exhibiting the maximum availability at the minimal cost) and a list of sub-optimal settings. Among the listed settings, we choose 6 of them $(S_1, \ldots, S_6)$, where S_1 represents the optimal one since it has the highest availability value at the minimum cost. Table II summarizes the composition of such 6 settings by specifying, in the second column, the number of redundant CFs per node. For instance, with respect to the optimal setting S_1, the P-CSCF node is made of 2 redundant CFs ($\text{CF}^{(P)} = 2$), the S-CSCF node is made of 2 redundant CFs ($\text{CF}^{(S)} = 2$), the I-CSCF node is made

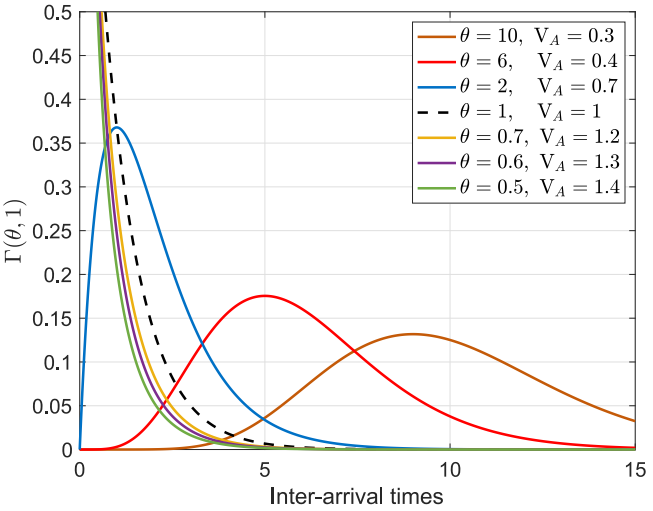


Fig. 7. Gamma-distributed inter-arrival times and corresponding coefficients of variation (V_A).

TABLE II
REDUNDANCY DEGREES FOR THE SIX EXEMPLARY SETTINGS (FOR $V_A = 1$)

Setting	Redundancy Degrees	$E^c(\boldsymbol{S})$	$A^c(\delta^*, \boldsymbol{S})$
$\boldsymbol{S_1}$	$[CF^{(P)} = 2, CF^{(S)} = 2, CF^{(I)} = 3, CF^{(H)} = 1]$	8	0.999992
$\boldsymbol{S_2}$	$[CF^{(P)} = 2, CF^{(S)} = 1, CF^{(I)} = 3, CF^{(H)} = 2]$	8	0.999985
$\boldsymbol{S_3}$	$[CF^{(P)} = 2, CF^{(S)} = 2, CF^{(I)} = 2, CF^{(H)} = 2]$	8	0.999957
$\boldsymbol{S_4}$	$[CF^{(P)} = 2, CF^{(S)} = 3, CF^{(I)} = 2, CF^{(H)} = 2]$	9	0.999957
$\boldsymbol{S_5}$	$[CF^{(P)} = 2, CF^{(S)} = 2, CF^{(I)} = 2, CF^{(H)} = 1]$	7	0.999949
$\boldsymbol{S_6}$	$[CF^{(P)} = 2, CF^{(S)} = 2, CF^{(I)} = 3, CF^{(H)} = 2]$	9	0.999999

of 3 redundant CFs ($\text{CF}^{(I)} = 3$), and the HSS is made of one CF ($\text{CF}^{(H)} = 1$). The third column reports the cost of each setting simply obtained as the sum of unitary costs of each CF per node. The fourth column reports the corresponding steady-state availability value. Now, by re-running PE-MUGF with a set of V_A values chosen among the most significant ones shown in Fig. 7, we re-evaluate the availability of the same six settings to make useful comparisons.

Such results are shown in Fig. 8 where, for the sake of comfort, y-axis reports (log scale) the unavailability values $(1 - A^c(\delta^*, \boldsymbol{S})$, lower is better) of the six settings.[4] We also draw three availability thresholds as horizontal black dashed bars at: 10^{-4} (four nines), 10^{-5} (five nines), and 10^{-6} (six nines). For example, when a bar lies above the 10^{-5} threshold, it means that the five nines steady-state availability requirement is violated.

For each setting we report 4 cases corresponding to different values of the coefficient of variation V_A. The first case includes a range of values obtained for $V_A \leq 0.7$ (blue bars). Focusing on this case, we see that S_1 (the optimal setting for the exponential case $V_A = 1$), S_2, and S_6 meet the five nines requirement (S_6 even satisfies the six nines requirement since the blue bar lies below the 10^{-6} line). Among the remaining settings, it is interesting to note that S_4 does not meet the five nines requirement even if its cost is higher than the S_1

[4]By construction, all the settings satisfy the performance constraint, namely, mean CSD $< \delta^*$.

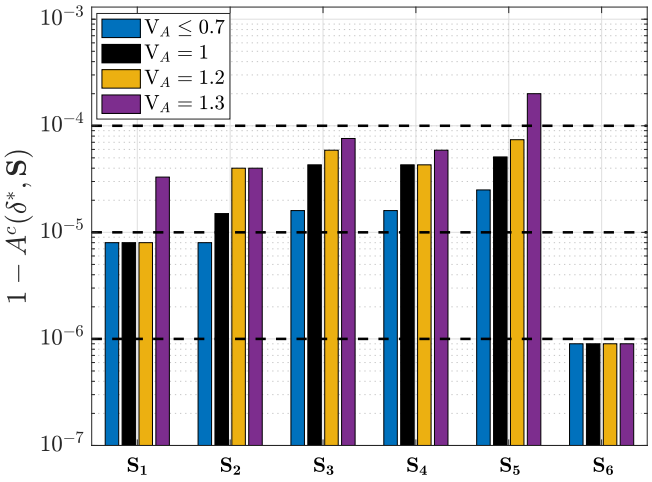


Fig. 8. Effect of the inter-arrival times variation on the steady-state availability of $\mathbf{S_1}, \ldots, \mathbf{S_6}$ settings.

cost ($E^c(\boldsymbol{S_1}) = 8$, and $E^c(\boldsymbol{S_4}) = 9$). As mentioned before, such an apparently counterintuitive behavior is due to a bad allocation strategy of redundant CFs for S_4. In particular, few redundant CFs have been assigned to the I-CSCF which, as can be seen from values in Table I, is slower than the other nodes to serve the IMS requests. This translates into higher values of mean sojourn times due to (5), with consequent impact on the overall availability according to (14).

By increasing V_A up to 1 (representing the benchmark case and whose availability values are reported numerically in Table II), we can notice that the availability of S_1 remains stable (0.999992), whereas the availability of S_2 trespasses the five nines threshold achieving 0.999985. We observe an availability decrease also for S_3 (from 999984 to 999957), S_4 (again from 999984 to 999957) and for S_5 (from 999975 to 999949), whereas S_6 continues to be stable. Similar considerations hold true for a V_A value of 1.2 (yellow bars) where in some cases the steady-state availability remains stable (S_1, S_4, S_6), whereas in the remaining cases it undergoes a deterioration. Finally, $V_A = 1.3$ (violet bars) seems to be a critical value since S_1 violates the five nines condition and S_5 even violates the four nines condition (by achieving the three nines), whereas, surprisingly, S_6 continues to fulfill the six nines condition. Thus, a network operator could decide to deploy S_6 (even if it not optimal due to its cost) since it appears to exhibit a great robustness to the variation of the inter-arrival times. At this point we can summarize some useful facts. First, we have seen how the availability is adversely affected when the inter-arrival times distributions show a greater variance (namely, V_A increases). To contrast such an effect we have two ways: the first one is to increment the redundancy paying the price of higher cost; the second one is to improve the service times (so as to reduce the impact of V_S in (6) up to a certain extent) but, also in this case, this translates into higher costs because more computation resources are needed.

The second fact is that the allocation strategy of CFs is crucial to obtain high availability values. In our case, in fact, S_1 achieves five nines even if no redundancy at all is provided for HSS. In contrast, S_3 which is obtained from S_1 by moving a CF replica from I-CSCF to HSS violates the five nines condition for all values of V_A. This is due to the fact that in S_1 we give more robustness to I-CSCF (with 3 CF replicas) which suffers from the slow service time.

The last fact is that, by paying a little more cost, we can obtain a very robust setting (S_6 in our case) with two advantages: first, it achieves the challenging six nines requirement (MAD of 32 seconds), and, then, it appears to be particularly insensitive to the variation of V_A which, as seen before, is detrimental for the steady-state availability.

C. Sensitivity Analysis

As the last analysis, we are interested in evaluating how the availability values are impacted when failure and repair parameters deviate from their nominal conditions (i.e., due to estimation errors or to non steady behaviours). Namely, we perform a sensitivity analysis wherein we fix the value V_A to 0.7 and compare the behavior of settings S_1 and S_2 since, for $V_A = 0.7$, both guarantee the high availability condition with the same value ($A^c(\delta^*, \boldsymbol{S_1}) = A^c(\delta^*, \boldsymbol{S_2}) = 0.999992$, for $V_A = 0.7$).

The three uppermost [lowermost] panels of Fig. 9 show the availability behavior in response to the variation of failure [repair] times for container, docker, and infrastructure layers. Each panel reports the horizontal blue dashed line as the five nines threshold. Moreover, the red circle includes the nominal value of the parameter as drawn from Table I. At first glance, we can notice that the different responses of the availability for S_1 and S_2 can be appreciated at the application layer, whereas the behaviors of the two considered settings tend to be the same at the docker and infrastructure layers. The reason is that failure and repair values for application layer span across a smaller range with respect to the remaining layers. More in detail, we can see that S_1 is more robust than S_2 to the variation of $1/\lambda_C$ (topmost-left panel). Precisely, for S_1 [S_2], $1/\lambda_C$ can be reduced of about 80% [65%] of its nominal value without violating the high availability condition. As concerns docker and infrastructure layers (topmost-middle and topmost-right panels), we note that parameters (both for S_1 and S_2) can be relaxed of about 16% and 50% of their nominal values, respectively. Likewise, the same robustness of S_1 w.r.t. S_2 is evident for $1/\mu_C$, whereas, no practical differences emerge in relaxing docker and infrastructure repair parameters for S_1 and S_2. Once again, it is useful to remark that, even if the number of CF replicas employed for S_1 and S_2 is the same (implying the same cost), the greater robustness of S_1 is explained through a better replicas allocation: the only node with no redundancy is the HSS which, in terms of mean service times, exhibits the best performance.

We finally note that such an analysis could be useful to decide between S_1 and S_2 that, according to PE-MUGF evaluated with $V_A = 0.7$, have the same availability and the same cost (0.999992 and 8), respectively. In fact, S_1 could be preferred since it turns out more robust w.r.t. variations of parameters λ_C and μ_C.

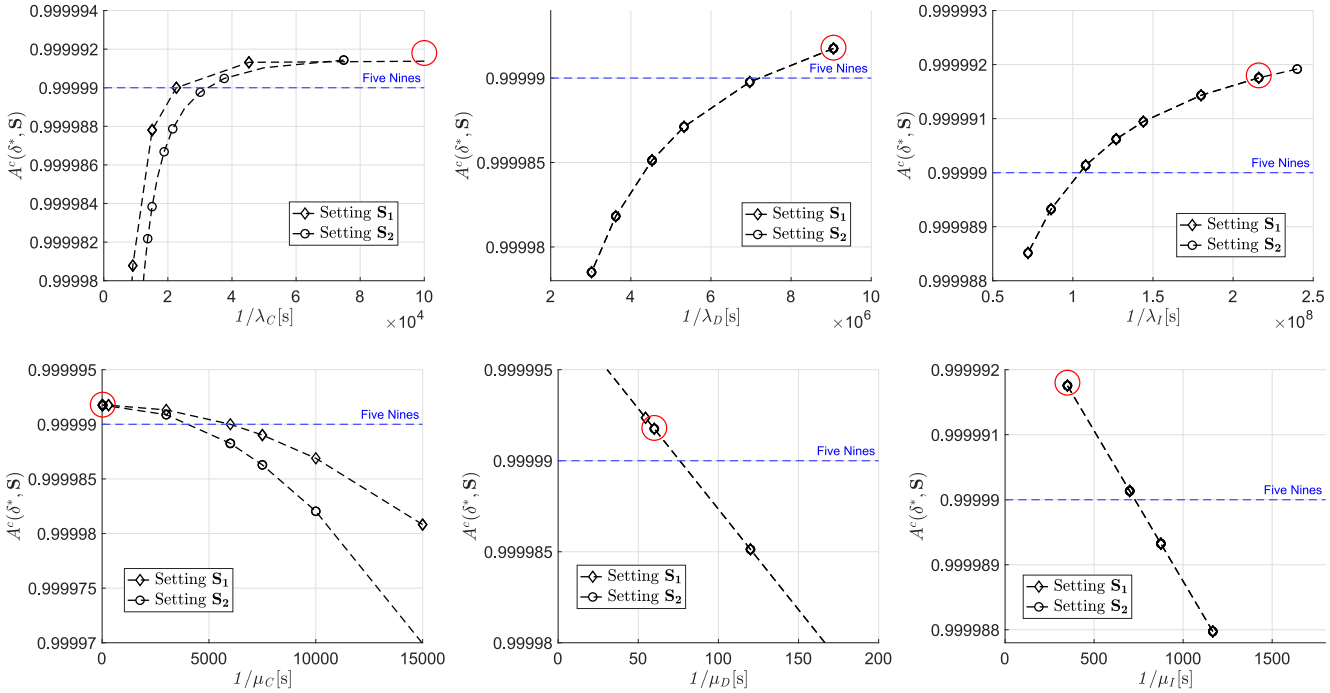


Fig. 9. Sensitivity analysis to evaluate the steady-state availability variation for two settings S_1 and S_2, when failure parameters (topmost panels) and repair parameters (lowermost panels) deviate from their normal behavior.

VIII. Conclusion

In this paper, we have examined in detail both performance and availability aspects of a container-based IMS (cIMS) infrastructure implementing the service function chain logic. In the first part, we have formalized an MSS model of the containerized function to cope with availability issues, and a *G/G/m* queueing model to deal with performance aspects. In the second part, supported by an *ad hoc* devised algorithm named PE-MUGF, we were able to derive the optimal-redundant cIMS setting where given performance (in terms of mean call setup delay) and availability (in terms of number of "nines") requirements are satisfied at the same time. The results allow to highlight that the allocation strategy of redundant cIMS elements is crucial to guarantee an availability value that depends as little as possible on the variations of the system parameters. Some hints for future developments may include: the possibility of further decomposing the MSS to take into account additional components (e.g., the hypervisor in case of virtual machine deployment); the possibility of embodying Quality-of-Service indicators to differentiate the cIMS requests according to some service classes (e.g., gold, bronze, silver); the possibility of examining the availability variations when the system is under particular stressed conditions (e.g., simulating busy hour requests).

References

[1] M. Gharbaoui et al., "An experimental study on latency-aware and self-adaptive service chaining orchestration in distributed NFV and SDN infrastructures," *Comput. Netw.*, vol. 208, pp. 1–15, May 2022.

[2] D. Borsatti, G. Davoli, W. Cerroni, C. Contoli, and F. Callegati, "Performance of service function chaining on the OpenStack cloud platform," in *Proc. IEEE CNSM*, 2018, pp. 432–437.

[3] Ericsson. "Cloud IMS." Accessed: May 28, 2023. [Online]. https://www.ericsson.com/en/portfolio/cloud-software--services/cloud-communication/consumer-communication/cloud-ims

[4] Nokia. "Voice over 5G core." Accessed: May 28, 2023. [Online]. https://www.nokia.com/networks/core-networks/voice-over-5g-vo5g-core/

[5] Huawei. "IMS." Accessed: May 28, 2023. [Online]. Available: https://carrier.huawei.com/en/products/core-network-v3/cs-ims/ims

[6] ETSI. "TS 123 251." Accessed: May 28, 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123200_123299/123251/14.00.00_60/ts_123251v140000p.pdf

[7] NEC. "Virtualized evolved packet core—vEPC." Accessed: May 28, 2023. [Online]. https://networkbuilders.intel.com/docs/vEPC_white_paper_w.cover_final.pdf

[8] Ericsson. "Virtualizing network services—The telecom cloud." Accessed: May 28, 2023. [Online]. https://www.ericsson.com/4ac61e/assets/local/reports-papers/ericsson-technology-review/docs/2014/er-telecom-cloud.pdf

[9] ETSI. "TS 101-563." Accessed: May 28, 2023. [Online]. https://www.etsi.org/deliver/etsi_ts/101500_101599/101563/01.03.01_60/ts_101563v010301p.pdf

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

[11] 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, Feb. 2017.

[12] ITU-T. "G.1028: End-to-end quality of service for voice over 4G mobile networks." Accessed: May 28, 2023. [Online]. Available: https://www.itu.int/rec/T-REC-G.1028/en

[13] Ericsson. "How can network operations make 5G systems resilient?" Accessed: May 28, 2023. [Online]. Available: https://www.ericsson.com/en/blog/2021/9/5g-resilient-system-network-operations

[14] L. De Simone, M. Di Mauro, M. Longo, R. Natella, and F. Postiglione, "Performability assessment of containerized multi-tenant IMS through multidimensional UGF," in *Proc. IEEE CNSM*, 2022, pp. 145–153.

[15] M. Niu et al., "HARS: A high-available and resource-saving service function chain placement approach in data center networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 829–847, Jun. 2022.

[16] Y. Zhang, F. He, and E. Oki, "Service chain provisioning with sub-chain-enabled coordinated protection to satisfy availability requirements," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 1629–1649, Jun. 2022.

[17] 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, Mar. 2021.

[18] J. Fan et al., "A framework for provisioning availability of NFV in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2246–2259, Oct. 2019.

[19] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF placement and resource allocation for the support of vertical services in 5G networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 433–446, Feb. 2019.

[20] Y. Yue, B. Cheng, M. Wang, B. Li, X. Liu, and J. Chen, "Throughput optimization and delay guarantee VNF placement for mapping SFC requests in NFV-enabled networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4247–4262, Sep. 2021.

[21] M. Nguyen, M. Dolati, and M. Ghaderi, "Deadline-aware SFC orchestration under demand uncertainty," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2275–2290, Dec. 2020.

[22] J. Zu, G. Hu, D. Peng, S. Xie, and W. Gao, "Fair scheduling and rate control for service function chain in NFV enabled data center," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 2975–2986, May 2021.

[23] A. Heideker and C. Kamienski, "Network queuing assessment: A method to detect bottlenecks in service function chaining," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4650–4661, Dec. 2022.

[24] P. K. Thiruvasagam, V. J. Kotagi, and C. S. R. Murthy, "The more the merrier: Enhancing reliability of 5G communication services with guaranteed delay," *IEEE Netw. Lett.*, vol. 1, no. 2, pp. 52–55, Jun. 2019.

[25] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *Proc. IEEE INFOCOM*, 2018, pp. 1943–1951.

[26] P. K. Thiruvasagam, V. J. Kotagi, and C. S. R. Murthy, "A reliability-aware, delay guaranteed, and resource efficient placement of service function chains in softwarized 5G networks," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1515–1531, Jul./Sep. 2022.

[27] R. Gouareb, V. Friderikos, and A. H. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, Oct. 2018.

[28] J. Bi, Z. Zhu, R. Tian, and Q. Wang, "Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center," in *Proc. IEEE CLOUD*, 2017, pp. 163–166.

[29] T. V. Phan, N. K. Bao, Y. Kim, H. J. Lee, and M. Park, "Optimizing resource allocation for elastic security VNFs in the SDNFV-enabled cloud computing," in *Proc. IEEE ICOIN*, 2010, pp. 370–377.

[30] H. Feng, Z. Shu, T. Taleb, Y. Wang, and Z. Liu, "An aggressive migration strategy for service function chaining in the core cloud," *IEEE Trans. Netw. Service Manag.*, early access, Dec. 23, 2022, doi: 10.1109/TNSM.2022.3231186.

[31] L. De Simone, M. Di Mauro, R. Natella, and F. Postiglione, "A latency-driven availability assessment for multi-tenant service chains," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 815–829, Mar./Apr. 2023, doi: 10.1109/TSC.2022.3183938.

[32] L. Rui, X. Chen, Z. Gao, W. Li, X. Qiu, and L. Meng, "Petri net-based reliability assessment and migration optimization strategy of SFC," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 167–181, Mar. 2021.

[33] G. L. Santos, P. T. Endo, T. Lynn, D. Sadok, and J. Kelner, "Automating the service function chain availability assessment," in *Proc. IEEE ISCC*, 2021, pp. 1–7.

[34] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, "Comparative performability assessment of SFCs: The case of containerized IP multimedia subsystem," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 258–272, Mar. 2021.

[35] M. Di Mauro, G. Galatro, F. Postiglione, and M. Tambasco, "Performability of network service chains: Stochastic modeling and assessment of softwarized IP multimedia subsystem," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 5, pp. 3071–3086, Sep./Oct. 2022.

[36] B. Tola, G. Nencioni, and 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, Dec. 2019.

[37] B. Tola, G. Nencioni, B. E. Helvik, and Y. Jiang, "Modeling and evaluating NFV-enabled network services under different availability modes," in *Proc. IEEE DRCN*, 2019, pp. 1–5.

[38] J. Zhu, N. Huang, J. Wang, and X. Qin, "Availability model for data center networks with dynamic migration and multiple traffic flows," *IEEE Trans. Netw. Service Manag.*, early access, Feb. 6, 2023, doi: 10.1109/TNSM.2023.3242321.

[39] K. Trivedi and R. Sahner, "Sharpe at the age of twenty two," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, pp. 52–57, May 2009.

[40] G. Ciardo, J. Muppala, and K. Trivedi, "SPNP: Stochastic Petri net package," in *Proc. PNPM*, 1989, pp. 142–151.

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

[42] F. Longo, M. Scarpa, and A. Puliafito, "WebSPN: A flexible tool for the analysis of non-Markovian stochastic Petri nets," in *Principles of Performance and Reliability Modeling and Evaluation* (Springer Series in Reliability Engineering). Cham, Switzerland: Springer, 2016, pp. 255–285.

[43] S. Yu, H. Chen, and Y. Xiang, *Maximal Service Profit in MAS-Based Cloud Computing Considering Service Security* (Lecture Notes in Electrical Engineering), vol. 355. Cham, Switzerland: Springer, 2015, pp. 861–867.

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

[45] "Clearwater project." 2018. Accessed: May 28, 2023. [Online]. Available: https://github.com/Metaswitch/clearwater-docker/tree/release-130

[46] R. E. Barlow and A. Wu, "Coherent systems with multi-state components," *Math. Oper. Res.*, vol. 3, no. 4, pp. 275–281, 2018.

[47] G. Levitin, *The Universal Generating Function in Reliability Analysis and Optimization*. London, U.K.: Springer-Verlag, 2005.

[48] 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, Dec. 2012.

[49] D. Tang, D. Kumar, S. Duvur, and O. Torbjornsen, "Availability measurement and modeling for an application server," in *Proc. IEEE DSN*, 2004, pp. 669–678.

[50] K. S. Trivedi and A. Bobbio, *Reliability and Availability Engineering*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[51] W. Whitt, *Stochastic-Process Limits: An Introduction to Stochastic-Process Limits and Their Application to Queues*. New York, NY, USA: Springer-Verlag, 2001.

[52] D. P. Bertsekas and R. G. Gallager, *Data Networks*. New York, NY, USA: Prentice-Hall, 1992.

[53] W. Krämer and M. Lagenbach-Belz, "Approximate formulae for general single systems with single and bulk arrivals," in *Proc. IEEE ITC*, 1976, pp. 235–243.

[54] T. Kimura, "A two-moment approximation for the mean waiting time in the GI/G/s queue," *Manag. Sci.*, vol. 32, no. 6, pp. 751–763, 1986.

[55] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains*, 2 ed. New York, NY, USA: Wiley, 2002.

[56] I. A. Ushakov, "A universal generating function," *Soviet J. Comput. Syst. Sci.*, vol. 24, no. 5, pp. 37–49, 1986.

[57] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Depend. Secure Comput.*. vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.

[58] 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, Dec. 2017.

[59] "Red hat OpenShift." Accessed: May 28, 2923. [Online]. Available: http://docs.openshift.com/container-platform/4.10/scalability_and_performance/recommended-host-practices.html

[60] "HashiCorp developer." Accessed: May 28, 2023. [Online]. Available: https://developer.hashicorp.com/hcp/docs/vault/high-avail-disaster-recover

[61] S. Sebastio, R. Ghosh, and T. Mukherjee, "An availability analysis approach for deployment configurations of containers," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 16–29, Jan./Feb. 2021.

[62] D. A. Patterson and J. L. Hennessy, *Computer Organization Design*, 4th ed. London, U.K.: Morgan Kaufmann, 2011.

[63] D. Ford et al., "Availability in globally distributed storage systems," in *Proc. USENIX*, 2010, pp. 61–74.

[64] A. N. Avramidis, A. Deslauriers, and P. L'Ecuyer, "Modeling daily arrivals to a telephone call center," *Manag. Sci.*, vol. 50, no. 7, pp. 896–908, 2004.

[65] B. N. Oreshkin, N. Reegnard, and P. L'Ecuyer, "Rate-based daily arrival process models with application to call centers," *Oper. Res.*, vol. 64, no. 2, pp. 510–527, 2016.

Luigi De Simone (Member, IEEE) is an Assistant Professor with the University of Naples Federico II, Italy. He contributed, as a author and a reviewer, to several top journals and conferences on dependable computing and software engineering. His research interests include dependability benchmarking, fault injection testing, virtualization reliability, and its application on safety- and secure- critical systems. He has been organizing multiple editions of the international workshop on software certification within the IEEE ISSRE Conference.

Mario Di Mauro (Senior Member, IEEE) is an Assistant Professor of Telecommunications with the University of Salerno, Italy. Before joining the academia, he was with Research Consortium on Telecommunications (formerly Ericsson Lab Italy) as a Team Leader in industrial research from 2007 to 2012. He authored more than 60 papers, mainly in the fields of network availability and security. His main fields of interest include network performance, network security and availability characterization, and data analysis for novel telecommunication infrastructures.

Maurizio Longo (Member, IEEE) is a Professor of Telecommunications, Emeritus, with the University of Salerno, Italy, where he served as the Dean of the Department of Information and Electrical Engineering and Applied Mathematics and as the Chairman of the Graduate School of Information Engineering. He has authored over 200 papers, mainly in the fields of statistical signal processing and telecommunication networks. His professional awards include a General Electric-Fulbright Fellowship in 1976, a Formez Fellowhip in 1986, a Lord Brabazon Prize from IERE-IEE in 1987, and a NATO-CNR Senior Fellowship in 1990.

Roberto Natella (Senior Member, IEEE) is an Associate Professor with the University of Naples Federico II, Italy. He authored more than 90 peer-review papers on software engineering and dependable computing in IEEE and ACM venues. His research interests include dependability benchmarking, software fault injection, software aging and rejuvenation, and their application in OS and virtualization technologies. In 2022, he received the DSN Rising Star in Dependability Award from the IEEE Technical Committee on Dependable Computing and Fault Tolerance and the IFIP Working Group 10.4 on Dependable Computing and Fault Tolerance.

Fabio Postiglione is an Associate Professor of Statistics with the University of Salerno, Italy. In the past, he was a Research Fellow with the University of Sannio and a Research Engineer with the Research and Development Department, Tin.it Company (Telecom Italia Group). He has coauthored about 130 papers, published in peer-reviewed international journals and conference proceedings, and one international patent. He is/has been involved in many European (FP7, H2020, EDF, HORIZON) and Italian research projects. His research interests include statistical characterization of degradation processes, reliability and availability modeling of complex systems, and Bayesian methods.

Open Access funding provided by 'Università degli Studi di Salerno' within the CRUI CARE Agreement