Performability Assessment of Containerized Multi-Tenant IMS through Multidimensional UGF

Luigi De Simone*, Mario Di Mauro†, Maurizio Longo†, Roberto Natella*, Fabio Postiglione†

*University of Napoli Federico II, Italy, {luigi.desimone,roberto.natella}@unina.it
†University of Salerno, Italy, {mdimauro,longo,fpostiglione}@unisa.it

***Abstract*—We advance a performability assessment of a multi-tenant containerized IP Multimedia Subsystem (cIMS), i.e.: one and the same infrastructure is shared among different providers (or tenants). Specifically, we: i) model each cIMS node (a.k.a. Containerized Network Function - CNF) through the Multi-State System (MSS) formalism to capture the dimensionality of the multi-tenant arrangement, and characterize each tenant through queueing theory attributes to catch latency-dependent performance aspects; ii) afford an availability analysis of cIMS by means of an extended version of the Universal Generating Function (UGF) technique, dubbed Multidimensional UGF (MUGF); iii) solve an optimization problem to retrieve the cIMS deployment minimizing costs while guaranteeing high availability requirements. The whole assessment is supported by an experiment based on the containerized IMS platform *Clearwater* which we deploy to derive some realistic system parameters by means of fault injection techniques.**

***Index Terms*—IP Multimedia Subsystem, Performability, Universal Generating Function, Redundancy Optimization.**

I. Introduction

Nowadays, network softwarization represents one of the most interesting paradigms allowing telco operators to improve flexibility and management efficiency [1]. Through the softwarization process, for example, it is possible to compose new services by means of service function chains (SFCs) [2], [3], where the interconnected elements are implemented in software by exploiting virtualization and containerization technologies. In particular, containers represent a lightweight solution to host network functionalities since they do not run any dedicated operating system. From a network management point of view, dealing with softwarized structures leads to carefully consider crucial metrics including: performance [4]–[7], namely the ability of guaranteeing a given service level, and availability [8]–[12], namely the probability of being operative at a given time). When combined, performance and availability give rise to the *performability* [13]. Accordingly, in this work we afford a performability assessment of a containerized IP Multimedia Subsystem (cIMS), that can be considered a particular kind of SFC and that is involved in multimedia contents management across mobile networks. Within the cIMS, each node is containerized and referred to as Containerized Network Function (CNF). Moreover, a CNF can be shared among different tenants (namely, different network providers) aimed at distributing management costs. Performances are captured through the Call Setup Delay (CSD), namely the average latency introduced across the whole cIMS infrastructure [14], whereas each tenant is modeled with an $M/G/k$ queueing system. Intuitively, the presence of redundant CNFs results in an overall CSD reduction since the load can be distributed among nodes. On the other hand, the CNF redundancy entails a steady-state availability analysis aimed at evaluating the probability that the cIMS is functioning in long runs. The availability target is typically expressed in terms of the number of *nines*: for example, five nines means that the cIMS has a steady-state availability of 0.99999, thus it can tolerate a maximum yearly downtime of 5 minutes and 5 seconds [15].

The following main findings stand out: i) We propose a Multi-State System (MSS) representation of a CNF shared among different tenants, where each state is described by a latency *performance vector*; ii) We employ the novel Multidimensional Universal Generating Function (MUGF) method which allows to find the MSS performance distribution of a multi-tenant cIMS by means of manageable algebraic procedures; iii) We sustain an availability assessment of a multi-tenant cIMS, by solving an optimization problem coupled with MUGF, to single out the optimal cIMS setting satisfying high availability requirements at a minimal cost; iv) We perform a large set of experiments (exploiting the *Clearwater* platform) to estimate values for model parameters, such as mean times to repair and mean service times.

A. Problem overview

Before delving into technical details, we offer an ampler overview of the addressed topic, by highlighting some key points pertaining both to technological and methodological aspects. On one hand, we move into the realm of network softwarization, where each network element can be decomposed into different layers resulting in a decoupling of hardware and software. Precisely, by adhering to the modern containerized paradigm, the software layer is further decomposed in containers (lightweight software facilities aimed at providing a specific service) and Docker layer (the software facilities manager). In accordance with such a model, we consider a

978-3-903176-51-5

containerized version of IP Multimedia Subsystem, a framework widely adopted within 4G and 5G to handle a number of multimedia services such as HD Voice/Video, presence information, multiparty gaming, content sharing and many others. The marriage between the container-based technology and the IMS framework opens new perspectives for providers (or tenants) which can share a common underlying service infrastructure (Hardware and Docker) to optimize costs, resulting in a multi-tenant infrastructure. On the other hand, the multi-tenant cIMS raises new intriguing challenges in terms of performability issues that we face by: i) adopting a Multi-State System representation useful to capture the complex dependencies among tenants states; ii) introducing the MUGF, a polynomial-like function designed to efficiently embody the relations among the said states, and that we exploit to solve the optimization problem aimed at finding the cIMS configuration which best satisfies the availability/costs trade off. Compared to classic Continuous-Time Markov Chains (CTMCs), the MUGF approach exhibits a reduced complexity, since it allows to express the overall cIMS performance distribution through a combination of single nodes performance distribution, in the spirit of a *divide et impera* principle. Moreover, by starting from a queueing model of IMS requests at each tenant, we achieve a good estimate of the mean Call Setup Delay (CSD), a key performance indicator in multimedia frameworks (such as IMS), that we use as performance metric in the MUGF.

II. Related Research

This work belongs to the broader field of joint performance and reliability analysis of computer systems (*performability*). Previous research addressed these aspects in many context, ranging from fault-tolerant safety-critical systems, to multi-tier web applications and IT networks [16]–[19].

With respect to affine literature dealing with availability aspects of softwarized networks, we can pinpoint more than one single element of difference and/or novelty.

For instance, we note that a part of literature focuses on availability problems where the models often account only for failures but not for repairs. It is the case of [20], where the authors face an availability issue concerning the optimal deployment of an SFC infrastructure. Precisely, they estimate the minimum number of backup VNFs satisfying a given availability level. Likewise, a heuristic algorithm to maximize the availability of an SFC through an optimal distribution of VNFs has been advanced in [21]. Authors in [22] tackle the virtual machine redundancy problem in a multi-tenant environment through optimal primary/backup logic. We highlight that, differently from such works, in our proposal we consider a complete failure/repair model of nodes under analysis.

Indeed, with respect to the model exploited to characterize the failure/repair life-cycle, another part of literature adopts compact formalisms having the benefit of avoiding errors during the model designing stage. It is the case of [23], where the Stochastic Reward Networks (SRNs) have been employed to face an availability assessment of some containerized infrastructures. SNRs have been adopted in [24] yet, to characterize some availability aspects of an Infrastructure-as-a-Service framework. Likewise, an availability analysis based on Stochastic Petri Networks (SPNs) has been carried out in [25], with applications in cloud-based arrangements. One drawback of such compact formalisms is the lack of a punctual characterization of system dynamics which makes it difficult to embed a performance metric. In contrast, the combination of the MUGF technique with the MSS formalism proposed in our work allows to jointly manage performance and availability metrics, having the complete access to each system state.

Remarkably, we also note that the UGF technique has been profitably adopted in the field of network management to characterize availability aspects. It is the case of [26], where the UGF has been adopted to calculate the expected service rewards in cloud environments; or [27], where a UGF-based technique has been employed to describe physical and virtual system failures. Such studies focus on single-tenant structures, with no need of a multidimensional form of UGF that we endorse in our work.

Finally, with respect to our previous work where MUGF [28] has been first conceived, here we present a restructured version thereof, useful to account for the novel latency-based metric and to quickly compute the performance distribution of the MSS. Moreover, in this new version, each tenant is characterized through the $M/G/k$ queueing formalism which allows to treat analytically the latency metric. Finally, an extensive experimental campaign (lacking in [28]) has been carried on a realistic cIMS deployment based on Clearwater platform, allowing us to estimate crucial quantities such as repair rates and mean service times.

III. Containerized IMS

In this section, we provide a quick description of a cIMS deployment realized through Clearwater [29], constituting the baseline architecture for our experimental analysis.

Each IMS node is developed as a container managed by a container engine (we use Docker in our deployment), which is installed on a physical machine. Figure 1 shows a sketch of Clearwater, where each containerized node is realized through a three-layer structure referred to as Containerized Network Function (CNF), whose details are presented in the next section. The following containerized nodes compose the cIMS infrastructure: *Bono*, which represents the P-CSCF (Proxy-Call Session Control Function) node and acts as contact point for users accessing IMS through Session Initiation Protocol (SIP); *Sprout*, which concurrently acts as S-CSCF (Serving-CSCF) for SIP registrations management, and as I-CSCF (Interrogating-CSCF) for users associations management; *Homestead* which represents the HSS (Home Subscriber Server) database for the users authentication procedures. The

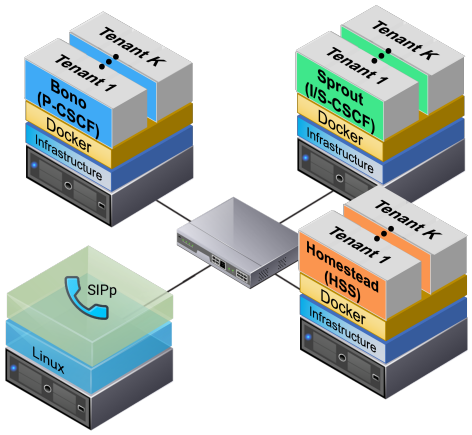


Fig. 1. Containerized IMS deployment through Clearwater.

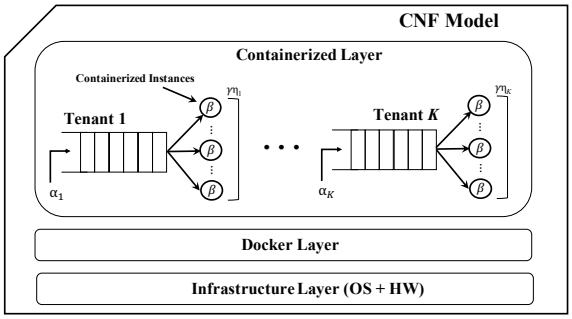


Fig. 2. A CNF which hosts K tenants. Each tenant is represented through a $M/G/\gamma\eta_i$ queueing model managing a set of containerized instances. Arrival and service rates are denoted by α_i and β, respectively.

architecture in Fig. 1 also includes an external node equipped with *SIPp*, a workload generator that we use to stress cIMS with SIP-based service requests. Through such service requests we are able to estimate the Call Setup Delay (CSD), namely the time interval between a *SIP Invite* message sent from the caller and the corresponding *Ringing* message received from the callee [30]. Remarkably, CSD has been elected as a critical Key Performance Indicator (KPI) [31]–[33] being strongly related to the end-user experience.

A. Containerized Network Functions (CNFs)

According to a decoupling logic offered by container technology [34]), a single CNF can be represented as a three-layer structure which includes: *Containerized Instances*, namely the upper layer which provides software instances (containers) representing specific IMS functionalities (e.g., Proxy, Interrogating, etc.); *Docker* layer, namely the intermediary between containers and physical layer; *Infrastructure*, namely the lower layer embodying both operating system (OS) and physical equipment (e.g., power supply, CPU, etc.).

As highlighted in Fig. 2, more tenants can share the same CNF. Precisely, tenant i ($i = 1, \ldots, K$) separately manages a maximum number of containerized instances (say n_i) within the containerized layer, but all tenants share the underlying layers (Docker and Infrastructure). In turn, a containerized instance is supposed to manage, at the same time, γ service requests which play the role of *serving capacity*. Such a quantity is crucial when designing the tenant queueing model presented in Sect. IV-A.

IV. Availability Model of CNF

From an availability perspective, each CNF layer admits a two-state model: active or failed. As regards the upper layer, for example, a containerized instance has a serving capacity amounting to γ when active, and to 0 when failed. Accordingly, the representation in Fig. 2 can be turned into the Multi-State System model depicted in Fig. 3 where:

- Each state is a K-dimensional vector $\boldsymbol{\eta} = (\eta_1, ..., \eta_K) \in \prod_{i=1}^{K}\{0, \ldots, n_i\}$, where $\eta_i \in \{0, 1, ..., n_i\}$ represents the number of working container instances belonging to tenant i. The initial state vector $(n_1, ..., n_K)$ pertains to a completely functioning system characterized by the maximum number of available working containerized instances per tenant. For instance, the vector $(n_1, ..., n_i - 1, ..., n_K)$ indicates a state where one of the containerized instances pertaining to tenant i is failed. Failure and repair processes of containerized instances are supposed to be Homogeneous Poisson Processes (HPPs), whose parameters (rates) are λ_{C_i} and μ_{C_i}, respectively.
- The state vector $(0, 0, ..., 0)$ pertains to a state with no working containerized instance holds, regardless of the Infrastructure layer and the Docker conditions.
- The state *Docker Layer Failure* (DLF) indicates the Docker failure condition which, in turn, causes the failure of all the containerized instances. Failure and repair processes of Docker are supposed to be HPPs whose rates are λ_D (dotted arrows from working states to DLF) and μ_D, respectively. Moreover, we assume that restoration of the Docker layer implies the restoration of all containerized instances, as remarked by the transition from DLF state to the initial 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. Likewise, failure and repair events of the Infrastructure layer are modeled as HPPs, where λ_I and μ_I are the corresponding rates. Similar to the previous case, the ILF brings down containerized instances and Docker layer (see dotted arrows towards ILF with the rate λ_I), whereas, a recovery of the Infrastructure layer is concluded with a complete system restoration (Docker and containerized instances layers) with the rate μ_I.

In case docker or infrastructure layers fail, no user request can be served, thus, the system is in practice unusable, a condi-

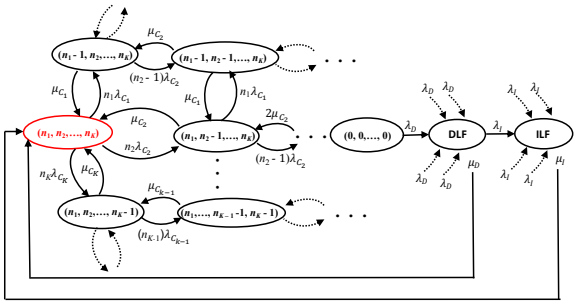


Fig. 3. Multi-State System (MSS) model of a CNF. In each state is reported the number of working containerized instances for all K tenants.

tion which is rarely met in modern cloud environments where infrastructure layers are extremely reliable. In contrast, when a containerized instance fails, only the part of user requests assigned to that instance will experience a malfunction.

Accordingly, we define the *capacity level* $g_{i,\boldsymbol{\eta}}$ associated to tenant i in state $\boldsymbol{\eta}$ the quantity:

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

Since CNF behavior varies over time (due to the presence of failures), the capacity level is time-varying itself and reflects the current condition.

The set which embeds all possible capacity levels $\boldsymbol{g_\eta} = (\boldsymbol{g}_{1,\boldsymbol{\eta}}, ..., \boldsymbol{g}_{K,\boldsymbol{\eta}})$ for each CNF is given by

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

where $(0, \ldots, 0)_D$ and $(0, \ldots, 0)_I$ are the capacity levels of the DLF and ILF states, respectively. Again, the total number of states pertaining to the MSS of Fig. 3 amounts to:

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

As a result, the CNF capacity level at time $t \geq 0$ can be expressed in terms of the vector stochastic process

$$\boldsymbol{G}(t) = (G_1(t), ..., G_K(t)) \in \mathcal{G}, \tag{4}$$

with (state) probability vector $\boldsymbol{p}(t)$ at time t gathering all the state probabilities $p_{\boldsymbol{\eta}}(t)$, being $p_{\boldsymbol{\eta}}(t) = \Pr\{\boldsymbol{G}(t) = \boldsymbol{g_\eta}\}$.

Given an initial probability vector at time $t = 0$, $\boldsymbol{p}(t)$ can be derived at $t \geq 0$ by solving the following system [35]:

$$\frac{\mathrm{d}\boldsymbol{p}(t)}{\mathrm{d}t} = \boldsymbol{p}(t)\mathbf{Q}, \tag{5}$$

along with the normalization condition $\sum_{\boldsymbol{\eta}} p_{\boldsymbol{\eta}}(t) = 1$. The symbol $\mathbf{Q}$ represents the infinitesimal generator matrix of the MSS shown in Fig. 3.

Remarkably, $\boldsymbol{p}$ is a vector which collects the steady-state probability $p_{\boldsymbol{\eta}}$ for each state $\boldsymbol{\eta}$, given by:

$$p_{\boldsymbol{\eta}} = \lim_{t \to \infty} p_{\boldsymbol{\eta}}(t) = \lim_{t \to \infty} \Pr\{\boldsymbol{G}(t) = \boldsymbol{g_\eta}\}. \tag{6}$$

Thus, the random vector $\boldsymbol{G} = (G_1, ..., G_K)$ represents the asymptotic behavior of $\boldsymbol{G}(t)$ $(t \to \infty)$ with values in the set (2) and with probabilities (6). In summary, the set of pairs $\{p_{\boldsymbol{\eta}}, \boldsymbol{g_\eta}\}$ determines the steady-state behavior of a CNF in terms of serving capacity.

A. CNF Queueing modeling

From a modeling point of view, the most crucial part of the whole CNF infrastructure is the containerized layer for two reasons. The first one is that, differently from docker and infrastructure layers, the containerized layer admits a non-trivial system state model where, as shown in Fig. 3, we consider each combination of states of containerized instances. The second reason is that, being involved in real-time session management, requests can be queued waiting to be served, implying the design of a queueing strategy. Across the technical literature, it is common the adoption of $M/M/1$ or $M/M/k$ systems to model the queueing aspects of multimedia (in particular, SIP-based) servers (see [36]–[38]). Unluckily, such models rely on an exponential service time assumption which is often unrealistic in the network field.

In contrast, we model each tenant as an $M/G/k$ queueing system, having the possibility to estimate generic (non-exponential) probability distributions of service times through our realistic testbed. More precisely, we propose an $M/G/G_i(t)$ model which accounts for the time-varying number of servers subject to failure/repair actions. At this aim, we highlight that, since the order of magnitude of arrival and service rates is much greater than failure and repair rates for all layers, it is possible to assume that the $M/G/G_i(t)$ model is equivalent to a $M/G/g_i = M/G/\gamma\eta_i$ for a certain state [39]. Pragmatically, given a state $\boldsymbol{\eta}$, tenant queues reach their steady-states very quickly compared to the fault occurrences.

We also remark that the mean CSD can be approximated by the average time that user requests spend in the containerized IMS, due to the processing time needed by each CNF to manage such requests. Thus, we evaluate the mean CSD per CNF (indicated by $\mathrm{CSD}^{(m)}$, with $m \in$ {P-CSCF, S-CSCF, I-CSCF, HSS}), where call setup requests are distributed according a Poisson process with parameter α_i. Before evaluating the mean CSD per CNF, we start to evaluate the average number of sessions $\nu_{i,\boldsymbol{\eta}}$ for an equivalent $M/M/\gamma\eta_i$ queueing model which describes tenant i in state $\boldsymbol{\eta}$ [35], namely

$$E[\nu_{i,\boldsymbol{\eta}}] = \gamma\eta_i \cdot \rho_{i,\boldsymbol{\eta}} + \rho_{i,\boldsymbol{\eta}} \frac{(\gamma\eta_i \cdot \rho_{i,\boldsymbol{\eta}})^{\gamma\eta_i}}{\gamma\eta_i!} \frac{\pi_{i,\boldsymbol{\eta}}}{(1 - \rho_{i,\boldsymbol{\eta}})^2}, \tag{7}$$

where $\rho_{i,\boldsymbol{\eta}} = \alpha_i / \gamma\eta_i$ is the *utilization factor*, and

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

Applying Little's law, the mean CSD$^{(m)}$ is:

$$E[d_{i,\boldsymbol{\eta}}] = \frac{E[\nu_{i,\boldsymbol{\eta}}]}{\alpha_i}. \tag{8}$$

Moreover, by exploiting the Kingman's law [40], we derive the mean CSD$^{(m)}$ for the considered $M/G/\gamma\eta_i$ model, representing the *performance level* of tenant i in state $\boldsymbol{\eta}$, viz.

$$\delta_{i,\boldsymbol{\eta}} \approx E[d_{i,\boldsymbol{\eta}}] \cdot \frac{1+CV_s^2}{2}, \tag{9}$$

where $CV_s = \sigma(S)/E[S]$ is the coefficient of variation of the empirical service time. Let now be

$$\boldsymbol{\Delta}(t) = (\Delta_1(t), ..., \Delta_K(t)), \tag{10}$$

the vector stochastic process gathering all tenants mean delays introduced by a single CNF. With a similar reasoning adopted before to derive the random vector $\boldsymbol{G}$, $\boldsymbol{\Delta}(t)$ can be expressed in terms of the random vector $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_K)$ as $t \to \infty$.

Interestingly, eq. (8) trivially suggests that mean CSD$^{(m)}$ is directly proportional to the average number of sessions. Thus, the increase of mean CSD$^{(m)}$ can be countered by: *i)* increasing the number of containerized instances $g_{i,\boldsymbol{\eta}}$ per each CNF, *ii)* introducing some redundant parallel CNFs, so that, sessions could be dispatched among a huger number of containerized instances belonging to those parallel CNFs (a.k.a. *flow dispersion* hypothesis). This latter option is, obviously, preferable when we have to cope with availability issues related to common layer failures, for instance, due to Docker and/or Infrastructure layers.

V. Availability of cIMS chain

Toward building the availability model for a cIMS chain we consider, first, that the entire cIMS is working when each CNF is working (*series* connection), and, second, that redundant CNFs have to be introduced to satisfy any availability requirement (*parallel* connection). The resulting availability model for a multi-tenant cIMS system is depicted in Fig. 4, where a series/parallel arrangement can be recognized. Specifically, CNF$^{(m,\ell)}$, which handles K tenants, represents the parallel CNF ℓ ($\ell = 1, \ldots, L_m$) associated to a specific m, $m \in$\{P, S, I, H\}, where P, S, I, H, indicate for brevity P-CSCF, S-CSCF, I-CSCF, HSS, respectively.

Let $\boldsymbol{\Delta}^{(m)}(t) = \left(\Delta_1^{(m)}(t), ..., \Delta_K^{(m)}(t)\right)$ and $\boldsymbol{\Delta}^c(t) = (\Delta_1^c(t), ..., \Delta_K^c(t))$ be the vector stochastic processes containing all mean CSDs$^{(m)}$ introduced, respectively, by node m and by the entire cIMS (namely, the series connection of nodes) at time t. Since the call flow has to traverse the whole cIMS chain, the overall delay is the sum of delays introduced by each single node and all the propagation delays. By neglecting the latters, for each tenant i, $\Delta_i^c(t) = \sum_m \Delta_i^{(m)}(t)$.

Accordingly, the set of pairs $\left\{p_{\boldsymbol{\eta}}^{(m)}, \boldsymbol{\delta}_{\boldsymbol{\eta}}^{(m)}\right\}$ represents the steady-state mean CSD distribution of CNF m, where: $\boldsymbol{\delta}_{\boldsymbol{\eta}}^{(m)} = \left(\delta_{1,\boldsymbol{\eta}}^{(m)}, ..., \delta_{K,\boldsymbol{\eta}}^{(m)}\right)$ is the mean delays vector of CNF 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. Similarly, $\left\{p_{\boldsymbol{\eta}}^c, \boldsymbol{\delta}_{\boldsymbol{\eta}}^c\right\}$ represents the steady-state mean CSD distribution of the entire cIMS.

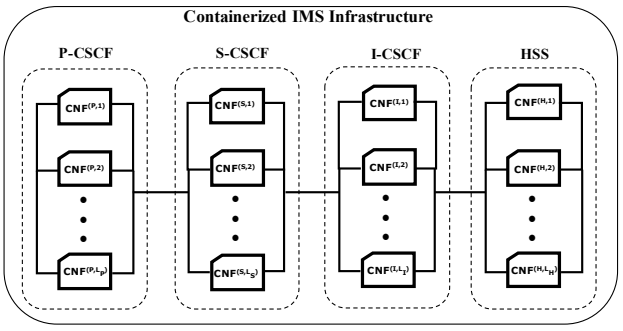


Fig. 4. Multi-tenant cIMS system where parallel CNFs are introduced for redundancy purposes. CNF$^{(m,\ell)}$ refers to parallel CNF ℓ ($\ell = 1, \ldots, L_m$) of node m, with $m \in$\{P-CSCF, S-CSCF, I-CSCF, HSS\}.

We consider the multi-tenant cIMS as *available* when every tenant is able to guarantee a mean CSD less than a (maximum) threshold value for its customers.

Thus, denoting with $\boldsymbol{W}^c(t) = (W_1^c(t), ..., W_K^c(t))$ the K-dimensional vector containing the maximum delay levels per tenant i at time t, the cIMS *instantaneous availability* $A^c[t, \boldsymbol{W}^c(t)]$ is defined (see [41], [42]) as the probability that mean CSD of cIMS for each tenant i (at $t > 0$) is not greater than $W_i^c(t)$, $i = 1, ..., K$, namely

$$A^c[t, \boldsymbol{W}^c(t)] = \Pr\{\Delta_i^c(t) - W_i^c(t) \le 0, \ \ \forall i = 1, ..., K\}. \tag{11}$$

Given constant maximum delay levels $\boldsymbol{W}^c(t) = \boldsymbol{w}^c = (w_1^c, ..., w_K^c)$, the *steady-state availability* $A^c(\boldsymbol{w}^c)$ can be derived from (11) for $t \to \infty$, as

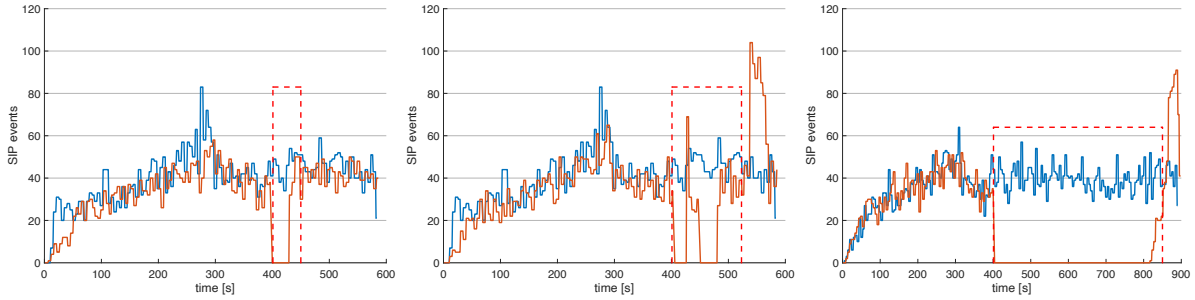
$$A^c(\boldsymbol{w}^c) = \sum_{\boldsymbol{\eta} \in J^c} p_{\boldsymbol{\eta}}^c \cdot \mathbf{1}\left(\delta_{i,\boldsymbol{\eta}}^c \le w_i^c, \forall i = 1, ..., K\right), \tag{12}$$

where $\mathbf{1}(\cdot)$ os 1 if condition holds true and 0 otherwise. The number of states J^c in (12) amounts to:

$$J^c = \prod_{m \in \{P,S,I,H\}} J^{(m)}, \tag{13}$$

where $J^{(m)} = \prod_{\ell=1}^{L_m} N^{(m,\ell)}$ represents the number of states of node m, and $N^{(m,\ell)}$ is given by (3) for each parallel CNF ℓ when $n_i^{(m,\ell)}$ containerized instances are taken into account for tenant i.

An efficient method to evaluate the steady-state availability in (12) is provided by MUGF, a hierarchical technique originally introduced in [28] as a multidimensional generalization of the UGF methodology [43]. Such a technique relies on a hierarchical approach which circumvents the problem of managing the huge state-space model of the whole cIMS, and allows to compute $\left\{p_{\boldsymbol{\eta}}^c, \boldsymbol{\delta}_{\boldsymbol{\eta}}^c\right\}$, by easily combining the steady-state distributions $\left\{p_{\boldsymbol{\eta}}^{(m)}, \boldsymbol{\delta}_{\boldsymbol{\eta}}^{(m)}\right\}$ of single CNFs.



(a) Containerized layer fault. (b) Docker layer fault. (c) Infrastructure layer fault.

Fig. 5. Sprout I-CSCF under fault injection. Blue: Fault Free execution. Orange: Faulty execution. Dashed Orange: Recovery window.

A. Insights of MUGF method

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. Precisely, by considering our cIMS setup in the multi-tenant case, the MUGF of the steady-state mean CSD$^{(m)}$ distribution pertaining to CNF m is

$$u^{(m)}(\boldsymbol{z}) = \sum_{\boldsymbol{\eta} \in J^c} p_{\boldsymbol{\eta}}^{(m)} \prod_{i=1}^{K} z_i^{\delta_{i,\boldsymbol{\eta}}^{(m)}}, \tag{14}$$

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

According to the generating functions theory, the sum of multivariate random variables has a generating function represented by the product of the generating functions of single variables. Thus, since the mean CSD of cIMS is the sum of mean CSDs$^{(m)}$, MUGF $u^c(\boldsymbol{z})$ of cIMS is the product of MUGFs of single CNFs computed by (14), namely.

$$u^c(\boldsymbol{z}) = \prod_{m} \left[\sum_{\boldsymbol{\eta} \in J^c} p_{\boldsymbol{\eta}}^{(m)} \prod_{i=1}^{K} z_i^{\delta_{i,\boldsymbol{\eta}}^{(m)}} \right], \tag{15}$$

being $u^c(\boldsymbol{z})$ a polynomial-shape function in $z_1, \ldots, z_K$ indeterminates. Finally, (12) provides the steady-state availability $A^c(\boldsymbol{w^c})$ of the multi-tenant cIMS architecture.

B. A redundancy optimization problem

A problem of practical interest is to identify those cIMS settings which minimize the number of redundant elements for each cIMS functionality.

We define by vector $\boldsymbol{S}$ a particular setting of multi-tenant cIMS, where each CNF m has its own redundant element ℓ (ℓ=1,...,L_m). Moreover, we denote by $C^{(m,\ell)}$ the cost of parallel CNF ℓ belonging to CNF m. Thus, the cost of the setting $\boldsymbol{S}$ of the multi-tenant cIMS is

$$C^c(\boldsymbol{S}) = \sum_{m} \sum_{\ell=1}^{L_m} C^{(m,\ell)}. \tag{16}$$

Let us denoting by d_{max} the maximum tolerated delay introduced by the whole cIMS (typically provided by international standards) for each tenant, namely $\boldsymbol{w}^c = (d_{max}, \ldots, d_{max})$. Accordingly, the steady-state availability of the configuration $\boldsymbol{S}$ in terms of mean CSD is derived by (12), namely

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

If we consider an availability constraint A^* (for istance, $A^* = 0.99999$ commonly known as "five nines" availability), it is possible to define the set of feasible cIMS settings which satisfy such a constraint as $\mathcal{L}^c = \{\boldsymbol{S} : A^c(d_{max}, \boldsymbol{S}) \geq A^*\}$.

Consequently, the cIMS settings with minimum cost which satisfy the availability constraint A^* can be derived as solution of the following optimization problem:

$$\boldsymbol{S}^* = \arg\min_{\boldsymbol{S} \in \mathcal{L}^c} C^c(\boldsymbol{S}). \tag{18}$$

VI. Experimental Results

The testbed we have deployed to perform experiments relies on Clearwater (release 130), where each node (namely, each CNF) as appears in Fig. 1 is equipped with: an operating system based on Linux kernel 4.4.0, a Docker engine (ver. 19.03.5) running on 16-Core 1.80GHz Intel Xeon CPU, and with 64GB of RAM. It is useful to separate the experimental part into two sub-parts: in the first part we estimate repair times (namely, μ_C, μ_D, μ_I); in the second part we derive a number of feasible cIMS settings (including the optimal one) by exploiting the MUGF method.

A. Parameters Estimate

We estimate the repair times μ_C, μ_D, μ_I, by applying *fault injection* techniques [44], [45]. Such techniques consist in deliberately introducing faults/errors at different levels of the target software systems to trigger failures. This approach aims at accelerating failure occurrence to assess the existing fault-tolerance mechanisms and estimate the failure recovery effectiveness of the system under test.

Precisely, we inject three fault types: i) *Containerized layer faults*, including abrupt terminations of containers and other software faults [47] such as resource exhaustion, I/O exceptions, race condition bugs; ii) *Docker layer faults* consisting in forcing the stop of the *dockerd* process which, in turn, provokes the termination of all managed containers; iii) *Infrastructure layer faults* consisting in causing an intentional crash of the physical machine, which translates in Docker and containers crash.

The fault injection experiments have been automated through custom routines developed to perform specific actions (start/stop fault injection, managing containers recovery, collecting SNMP metrics). Through such routines we perform 30 fault injection experiments per CNF and per fault type, resulting in 360 experiments.

We have measured about 10 minutes in trials involving Containerized and Docker layers, and about 15 minutes in those involving the Infrastructure layer. Moreover, before injecting faults, we wait for an extra-time (about 400 seconds) to let the underlying software processes reach a regime and to fully restore steady-state performance due to caching effects and queuing delays.

This fault injection experiment represents a substantial advancement w.r.t. most previous model-based availability studies that make simplistic assumptions about the time-to-recovery for container-based scenarios by only considering the time to perform a restart action for a container (in the order of tens of seconds). In contrast, our experiment allows us to estimate more precisely the container recovery times, which can be in the order of minutes.

For the sake of simplicity, we just report the condition of Sprout I-CSCF under fault injection in Fig. 5 for: Containerized layer fault (panel 5a), Docker layer fault (panel 5b), and Infrastructure layer fault (panel 5c).

The repair time coincides with the unavailability period of a CNF, beginning at a given fault injection time and ending when the metric raises up to its regime value after repair 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 [46]. The variability of experiments is responsible of the difference among curves in Fig. 5. The estimated repair times, along with other parameters (some of which derived from technical literature) are shown in Table I.

TABLE I
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 s^{-1}
α_2	IMS request arrival rate at tenant 2‡	200 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
d_{max}	Maximum tolerated CSD	50 ms

† From scientific literature
‡ From experiments

B. Deriving feasible cIMS settings through MUGF

In this second experimental part, we perform an availability assessment aimed at pinpointing the optimal cIMS setting $\boldsymbol{S}^*$ which satisfies (18). Along such an optimal setting, we also derive additional settings to make useful comparisons. Pragmatically, we consider a cIMS configuration with 2 tenants ($K = 2$), where the first tenant manages $n_1 = 2$ containerized instances and the second tenant manages $n_2 = 3$ containerized instances. The resulting MSS model is depicted in Fig. 6, where the number of states amounts to $N = (n_1+1)(n_2+1)+2 = 14$ according to (3). A routine written in Mathematica® allows to: i) compute the MUGF; ii) evaluate the steady-state availability in (12); iii) solve the optimization problem (18) with $A^* = 1 - 10^{-5}$ (five nines). The routine produces in output all the feasible cIMS configurations, including the optimal one.

From such an output we extract five exemplary settings whose results are reported in Fig. 7. For a better readability, on the y-axis we report the unavailability $1 - A^c(\boldsymbol{w^c})$. Two dashed lines at 10^{-5} and 10^{-6} represent the unavailability thresholds corresponding to five nines (high availability) and six nines, respectively. For instance, $\boldsymbol{S_1}$ achieves five nines (the corresponding bar lies below the 10^{-5} line) but not six nines (the corresponding bar lies above the 10^{-6} line). Within the legend in Fig. 7 we report the composition of each setting by using the following formalism: $\text{CNF}^{(m)} = k$ means that the m-th CNF admits k parallel CNFs of the same type (or replicas). For example, $\text{CNF}^{(P)} = 3$ means that the Proxy CNF is made of 3 P-CSCF replicas. By assuming that each CNF has a unitary cost, within each bar we report the cost for each setting. Thus, for the setting $\boldsymbol{S_1}$ the cost amounts to $3 + 3 + 3 + 1 = 10$.

Some interesting considerations can be drawn by analyzing the considered settings. First, it is easy to recognize setting $\boldsymbol{S^*}$ as the one with the best availability/cost trade-off (five nines with a cost of 8). Interestingly, we can see that $\boldsymbol{S_2}$ exhibits the same cost of $\boldsymbol{S^*}$ but it is not able to achieve the five nines. In practice, $\boldsymbol{S^*}$ and $\boldsymbol{S_2}$ have the same number of replicas but differently allocated. In $\boldsymbol{S_2}$ we have a perfect balancing of replicas across all nodes (2 replicas per node), but this strategy seems to be not effective in achieving the desired availability target. In contrast, an unbalanced CNF replicas allocation

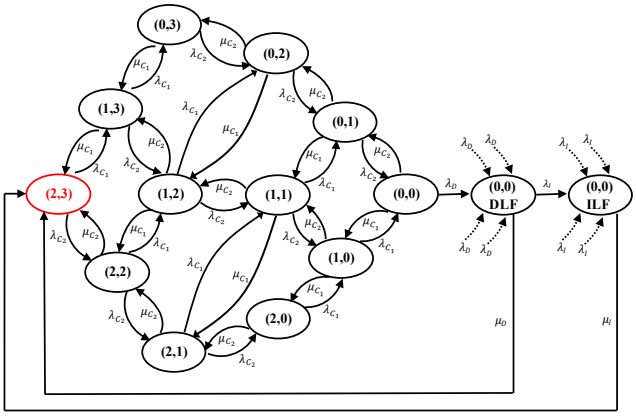


Fig. 6. Transition-state diagram of a CNF with two tenants (14 states).

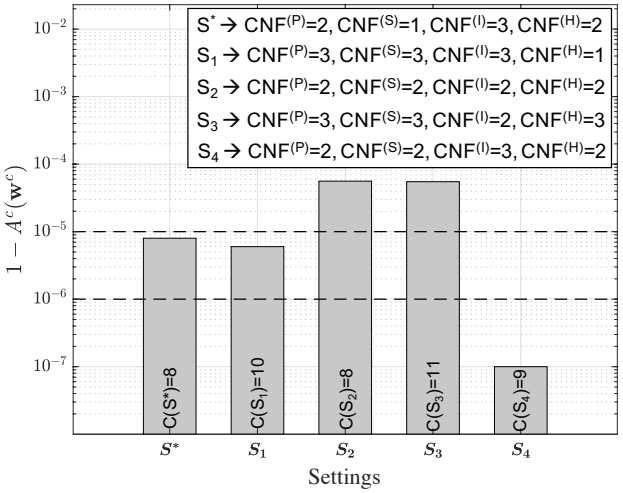


Fig. 7. Five exemplary settings (including the optimal one S^*) derived via MUGF. On the y-axis is reported the unavailability $1-A^c(\boldsymbol{w}^c)$. Within each bar is reported the cost associated to each setting.

strategy has been adopted for $\boldsymbol{S_1}$, where no redundancy at all for HSS has been envisaged. This notwithstanding, $\boldsymbol{S_1}$ achieves five nines availability but at a higher cost (amounting to 10) with respect to $\boldsymbol{S^*}$. We also note that, unexpectedly, $\boldsymbol{S_3}$ is not able to reach the high availability condition despite its high cost which amounts to 11. Such an apparently weird behavior is due to the fact that I-CSCF is the most critical node since its service time is particularly high (see Table I) if compared with service times of remaining nodes (P-CSCF, S-CSCF, HSS). Such a criticality directly impacts the availability result which embeds the delay performance metric that, in turn, is related to service time via (9). Finally, it is interesting to note that, with a relatively low cost (amounting to 9), we are able to obtain a setting which meets the challenging six nines requirement, namely $\boldsymbol{S_4}$. Pragmatically, such a setting is obtained from $\boldsymbol{S^*}$ by adding a CNF replica to S-CSCF.

In summary, through the MUGF method we are able to derive availability results of feasible cIMS settings, by easily embodying the performance metric (namely, the mean CSD). Then, the solution of the optimization problem (18) allows to derive the optimal setting $\boldsymbol{S^*}$ but also to save some additional settings which meet different criteria such as: perfect balance of CNF replica allocation or satisfying higher availability requirements (e.g., six nines). We want to remark that the originality of obtained results is largely due to the presence of containerized layer which admits a challenging characterization: first, the containerized instances queueing model allows to obtain a realistic estimate of the CSD, and, then, the application of MUGF allows to elegantly formulate a steady-state availability problem.

Finally, from a complexity point of view, we want to highlight that the application of MUGF is beneficial with respect to some monolithic approaches such as the Continuous-Time Markov Chains (CTMCs). At this aim, we highlight that, by exploiting CTMCs, the cIMS state space amounts to $J^c = 14^{\sum_{m\in\{P,S,I,H\}} L_m}$ (by virtue of (13)). In our numerical setting, this is tantamount to solve a system with $14^{2+1+3+2} = 14^8$ equations to obtain the optimal configuration $\boldsymbol{S^*} = (2,1,3,2)$. In contrast, through the MUGF approach we compute the steady-state distribution of the mean CSD per CNF, where a system of 14 equations has to be solved (being 14 the number of states of the MSS model per CNF - see Fig. 6). Then, exploiting the MUGF operators, we are able to combine the CNF distributions to get the mean CSD probability distribution of the whole cIMS, along with the corresponding steady-state availability obtained by applying (12). The MUGF computation to retrieve the cIMS optimal configuration requires about 360 s (PC with Intel Quad-Core Xeon E5 CPU@3.7GHz.)

VII. Concluding Remarks

We have employed a multidimensional version of UGF (MUGF) to perform a joint analysis (performance and availability, a.k.a. performability) of a cIMS infrastructure shared by different tenants and represented by a series/parallel arrangement. In particular, the performance metric (the mean Call Setup Delay) has been derived through the application of a $M/G/k$ queueing model of a cIMS node (or CNF). Then, such a performance metric has been embedded into the MUGF structure to evaluate the cIMS steady-state availability with the mean CSD as a constraint. From an experimental point of view, we have deployed a testbed based on *Clearwater*, an opensource implementation of cIMS, allowing us to estimate some critical parameters (e.g., the mean repair times) via fault injection techniques. A natural evolution of such a work would be to extend the proposed method to assess the performability of other service chains, where the series/parallel arrangements may assume more complex and tangled forms.

REFERENCES

[1] W. Cerroni, A. Galis, K. Shiomoto, and M. F. Zhani, "Telecom Software, Network Virtualization, and Software Defined Networks," *IEEE Communications Magazine*, vol. 58, no. 7, pp. 42–43, 2020.

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

[3] D. Borsatti, G. Davoli, W. Cerroni, and F. Callegati, "Service function chaining leveraging segment routing for 5G network slicing," in *Proc. IEEE CNSM*, pp. 1–6, 2019.

[4] 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*, pp. 432–437, 2018.

[5] T. Wen, H. Yu, and X. Du, "Performance guarantee aware orchestration for service function chains with elastic demands," in *Proc. IEEE NFV-SDN*, pp. 1–4, 2017.

[6] L. Cao, P. Sharma, S. Fahmi, and V. Saxena, "NFV-VITAL: A framework for characterizing the performance of virtual network functions," in *Proc. IEEE NFV-SDN*, pp. 93–99, 2015.

[7] M. G. Khan et al., "A Performance modelling approach for SLA-aware resource recommendation in cloud native network functions," in *Proc. IEEE NetSoft*, pp. 292–300, 2020.

[8] 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, 2019.

[9] 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.

[10] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, "HASFC: A MANO-compliant framework for availability management of service chains," *IEEE Commun. Mag.*, vol. 59, no. 6, pp. 52–58, 2021.

[11] P. C. Rangarajan, F. Khendek, and M. Toeroe, "Managing the availability of VNFs with the availability management framework," in *Proc. IEEE CNSM*, pp. 1–4, 2017.

[12] R. Nakamura, and N. Kamiyama, "Analysis of content availability at network failure in information-centric networking," in *Proc. IEEE CNSM*, pp. 1–7, 2020.

[13] B.R. Haverkort, R. Marie, G. Rubino, and K.S. Trivedi, *Performability modelling techniques and tools*. Chichester(UK), John Wiley and Sons, Ltd., 2001.

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

[15] M. Di Mauro, M. Longo, F. Postiglione, G. Carullo, and M. Tambasco, "Service function chaining deployed in an NFV environment: An availability modeling," in *Proc. IEEE CSCN*, pp. 42–47, 2017.

[16] J.F. Meyer, "On evaluating the performability of degradable computing systems,", in *IEEE Trans. Computers*, vol. 8, pp. 720–731, 1980.

[17] W.H. Sanders, J.F. Meyer, "A unified approach for specifying measures of performance, dependability and performability," in *Proc. Dependable Comp. Crit. App.*, Springer, Vienna, 1991.

[18] R.A. Sahner, K. Trivedi, A. Puliafito, "Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package," Springer Science & Business Media, 2012.

[19] A. Bondavalli, S. Chiaradonna, D. Cotroneo, L. Romano, "Effective fault treatment for improving the dependability of COTS and legacy-based applications," *IEEE Trans. Dep. Sec. Comp.*, vol. 1, no. 4, pp. 223-237, 2004

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

[21] 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*, pp. 1–6, 2017.

[22] 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.

[23] 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, 2021.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] Clearwater Project, 2018 [Online]. http://www.projectclearwater.org/.

[30] K. Al-Begain, and A. Ali, *Multimedia services and applications in mission critical communication systems*. Hershey (PA), IGI Global, 2017.

[31] 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.

[32] J. E. Vargas 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, 2013.

[33] 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*, pp. 7-11, 2014.

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

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

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

[37] S. V. Subramanian, R. Dutta, "Comparative study of M/M/1 and M/D/1 models of a SIP proxy server ," in *Proc. IEEE ATNAC*, 2008, pp. 397-402.

[38] S. V. Subramanian, R. Dutta, "Performance and scalability of M/M/c based queuing model of the SIP Proxy Server - a practical approach", in *Proc. IEEE ATNAC*, 2009, pp. 1-6.

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

[40] T. Kimura, "Approximations for multi-server queues: System interpolations," in *Queueing Systems*, vol. 17, no. 3, pp. 347–382, 1984.

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

[42] G. Levitin, "A Universal Generating Function in the analysis of multi-state systems," in: A Universal Generating Function in the Analysis of Multi-state Systems, M. K. Misra Eds. Springer London, 2008, pp. 447–464, ISBN: 978-1-84800-131-2.

[43] I. A. Ushakov, "A Universal Generating Function," *Soviet Journal of Computer Systems Science*, vol. 24, no. 5, pp. 37–49, 1986.

[44] 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.

[45] D. Cotroneo, A.K. Iannillo, R. Natella, and S. Rosiello, "Dependability assessment of the Android OS through fault injection," *IEEE Trans. Rel.*, vol. 70, no. 1, pp. 346–361, 2021.

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

[47] D. Cotroneo, L. De Simone, R. Natella, "Run-Time detection of protocol bugs in storage I/O device drivers," *IEEE Trans. Rel.*, vol. 67, no. 3, pp. 1–16, 2019.