Performance and Availability Challenges in Designing Resilient 5G Architectures

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

***Abstract*—This work proposes a stochastic characterization of resilient 5G architectures, where attributes such as performance and availability play a crucial role. As regards performance, we focus on the delay associated with the Packet Data Unit session establishment, a 5G procedure recognized as critical for its impact on the Quality of Service and Experience of end-users. To formally characterize this aspect, we employ the *non-product-form queueing networks* framework where: *i*) main nodes of a 5G architecture have been realistically modeled as $G/G/m$ queues which do not admit analytical solutions; *ii*) the *decomposition method* useful to catch subtle quantities involved in the chain of 5G interconnected nodes has been conveniently customized. The results of performance characterization constitute the input of the availability modeling, where we design a hierarchical scheme to characterize the probabilistic failure/repair behavior of 5G nodes combining two formalisms: *i*) the Reliability Block Diagrams, useful to capture the high-level interconnections between nodes; *ii*) the Stochastic Reward Networks to model the internal structure of each node. The final result is an optimal resilient 5G setting that fulfills both a performance constraint (e.g., a temporal threshold) and an availability constraint (e.g., the so-called *five nines*) at the minimum cost, namely, with the smallest number of redundant elements. The theoretical part is complemented by an empirical assessment carried out through *Open5GS*, a 5G testbed that we have deployed to realistically estimate main performance and availability metrics.**

***Index Terms*—5G networks, Performance and Availability analyses of 5G networks, Resilience of 5G networks.**

I. Introduction and Contribution

Before the advent of virtualization, designing resilient network infrastructures was tantamount to replicate the entire physical architecture to guarantee service continuity when anomalous events (including network attacks, natural disasters, etc.) occurred. In recent years, the decoupling between hardware and software has made it more flexible to implement resilience strategies: a given functionality offered by a node, in fact, can be easily embodied into a software module (commonly known as a Virtual Network Function - VNF) and replicated across the network to guarantee a certain resilience. Accordingly, we propose a resilience assessment of a virtualized 5G infrastructure by taking into account two interconnected aspects. The first one is performance, pertaining to the ability of a system to provide a service by not exceeding a predefined time threshold. Such an aspect becomes even more critical when dealing with systems of interconnected nodes realizing a so-called Service Function Chain (SFC) [1], [2], due to the propagation of delays across the whole chain. The second aspect is the availability, referring to the ability of a system to be *up and running* when called for use. Typically, a given availability target (e.g., the so-called *five nines*) is achieved by increasing the redundancy degree of a network node [3], [4]. Remarkably, different and flexible availability strategies involving the hardware part, the virtualization part, or the software part of a node can be implemented to fulfill the desired availability target [5]. Obviously, having a high number of nodes results in growing costs. Thus, implementing an effective resilience strategy means designing an optimal 5G virtualized architecture able to satisfy given performance and availability requirements at the minimum cost [6].

This is the main focus of our proposal, where the following contributions emerge:

- We devise a performance model relying on the *non-product-form queueing networks* framework, able to well capture the "chained" connection among the 5G nodes, and where each node is realistically modeled as a $G/G/m$ queue. The objective is to analyze the latency impact of the whole chain onto the *Packet Data Unit (PDU) session establishment*, a crucial procedure in 5G networks in charge of providing a data path between User Equipments (UEs) and the 5G core network. Remarkably, such a procedure is recognized to be critical in terms of the QoS/QoE impact on the end users [7].
- We devise a *hierarchical* availability model able to: *i*) characterize high-level interconnections among 5G nodes through Reliability Block Diagrams (RBDs); *ii*) characterize probabilistically the failure/repair behavior of the 5G nodes by employing the Stochastic Reward Network (SRN) methodology, where we differentiate between proactive recovery (i.e., controlled reboots, the so-called *rejuvenation*) and reactive recovery (reboots due to failure events).
- Relying on a testbed based on Open5GS, we have designed specific software routines aimed at: *i*) estimating the service times distributions of the crucial Open5GS nodes that we use as the input parameters for the performance model; *ii*) evaluating the repair times of Open5GS

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

nodes through *fault injection* techniques, that we use as the input parameters for the availability model.

In summary, the final purpose is to evaluate a set of (optimal) resilient 5G settings able to satisfy at the same time: *i*) the performance requirements, by providing mechanisms to model and control the latency introduced by various nodes, and *ii*) the availability requirements, by introducing a controlled level of redundancy to fulfill a given availability constraint, namely the *five nines*, corresponding to a maximum tolerated downtime of 5.26 minutes per year.

We organize the remainder of the paper as follows: Section II proposes a collection of affine works, where the main novelties of our proposal are highlighted. In Sect. III we describe the architecture of our testbed based on Open5GS, useful to gather/estimate performance and availability data. Section IV is focused on the queueing networks modeling that we exploit to characterize performance features of Open5GS. Similarly, in Sect. V we propose an availability modeling of Open5GS nodes by employing the Stochastic Reward Networks. In Sect. VI we detail the experimental results carried out on the open5GS infrastructure, and Sect. VII concludes the paper by summing up the main findings.

II. Related Work

Performance and availability are two crucial aspects to take into account when designing resilient modern virtualized network infrastructures. A resilient network, in fact, must satisfy strict performance requirements to support low latency and fast data transmission, but also specific high availability needs to guarantee uninterrupted access to the provided services. As concerns performance aspects, there are many works focused on latency-related issues of 5G-based infrastructures, where the queueing theory is widely adopted. It is the case of [8], where the authors propose a heuristic solution called TO-DG to optimize the throughput with latency guarantees when instantiating VNFs in a virtualized network environment, and where the adopted queueing model involves exponentially distributed quantities. Authors in [9] focus on the resource allocation in 5G networks and formulate an optimization problem to minimize the maximum ratio between actual and maximum end-to-end latency, where each VNF follows the well-known $M/M/1$ queueing model. A method to estimate 5G network service resilience performance is proposed in [10], where exponentially distributed arrival and service times have been considered. In particular, the authors identify the traffic changes as primary threats to service resilience, and put in the field an orchestration system along with network slices as potential solutions. Authors in [11] propose a method for enhancing the reliability of an SFC in 5G communication services, where the involved VNFs have been modeled both as $M/M/1$ and $M/M/m$ queueing systems. An $M/M/1$ queueing model has been adopted also in [12], where a method to detect bottlenecks in SFCs based on network queue occupation has been devised, and in [13] to model the SFC scheduling process in data centers. An $M/G/1$ queueing model has been adopted to characterize the latency of ultra-reliable low latency communications (URLLC) in 5G environments in [14]. Once quantified such a latency, the authors devise a resource allocation scheme achieving a good balance between URLLC latency and enhanced mobile broadband (eMBB) throughput. A resource allocation procedure for 5G new radio-based vehicle-to-vehicle has been advanced in [15], where an $M/G/1$ queueing scheme has been employed to model the end-to-end latency. In [16], the authors address the problem of deploying network services in 5G cloud systems based on latency requirements. Both $M/M/1$ and $M/M/m$ models have been exploited to characterize the presence of one processing unit and two or more processing units per VNF, respectively.

A common characteristic of all the aforementioned works is the usage of queueing models ($M/M/1$, $M/M/m$, $M/G/1$) that admit an exact solution through the employment of closed formulas. A drawback is that, in many cases, such models rely on assumptions (e.g., exponentially distributed arrival/service times) that could not represent with sufficient precision the behavior of real-time systems. Accordingly, we try to fill this gap by adopting the most possible general queueing scheme $G/G/m$ (G stands for *general* distribution) to characterize the latency introduced by a node with m containerized instances on top. Each $G/G/m$ node model is then embedded into a queueing network model to estimate the latency associated with the whole chain by exploiting the so-called *decomposition method*, a framework useful to deal with *non-product-form* queueing networks (namely, queueing networks where the exponentially assumptions are violated). The output of our performance model is the optimal number of containerized instances (say m^*) to deploy onto each node aimed at guaranteeing that the overall latency (that, in turn, negatively affects the PDU session establishment procedure) is kept under a given threshold.

As regards the availability aspects, the technical literature proposes different solutions to characterize failure and repair actions affecting a system, where redundancy strategies must be set to face possible breakdowns due to disasters, attacks, etc. Since failures and repairs can be modeled as particular states of a system, one of the most natural choices is to employ state-space formalisms such as the well-known Continuous-Time Markov Chains (CTMCs). In line with this consideration, the authors in [17] characterize, from an availability/reliability perspective, a set of related VNFs that form the so-called network service. For the modeling stage, a CTMC has been adopted to encode failure and repair states. The final aim is to compare the availability of different solutions (i.e., VNFs placed in a single host node rather than in multiple host nodes). In [18] the authors perform an availability evaluation of a cloud system by combining hierarchically two formalisms: Reliability Block Diagrams (RBDs) to catch high-level interconnections between nodes constituting a cloud infrastructure, and CTMCs to model failure/repair events of components inside the nodes. CTMC modeling has been adopted also to evaluate the availability of some radio-based elements of 5G networks (see [19]), where the authors have modeled the failure/repair behavior of the base stations. Yet, a dependability assessment of the 5G-AKA authentication service in the presence of failures by employing CTMCs has been proposed in [20]. Finally, authors in [21]–[23] use CTMC as the basis

to build a multi-state system model of virtualized nodes of IP Multimedia Subsystem (IMS), a framework adopted to manage multimedia content in 4G/5G infrastructures. Even if the CTMC modeling has the advantage of allowing full control of single failure/repair states of a system, it has the drawback of suffering from the so-called state-space explosion. It means that, when modeling the failure/repair behavior of real-world systems, the corresponding CTMC easily exhibits a huge number of states, which makes it difficult to deal with. Accordingly, especially when evaluating the availability of complex systems, it is preferable to employ more compact formalisms such as the stochastic networks and their variants. Across the technical literature, we find a number of works where such a formalism has been adopted to characterize failure and repairs of 5G-related network infrastructures. It is the case of [24], where the authors examine the usage of the Stochastic Petri Net (SPN) framework to automate the evaluation of SFCs availability along with specific redundancy strategies. SPNs have been also adopted in [25], where an algorithm for resource sharing in a virtualized scenario has been designed. In this work, the authors assume that the process of resource virtualization is itself resource-consuming, thus failures may occur. As a solution, some backup-sharing strategies are implemented with the support of an SPN model of virtualized blocks. Timed Stochastic Colored Petri Nets have been employed in [26] to analyze the resiliency of a 5G infrastructure by dividing it into the main principal components (the physical infrastructure, the virtual infrastructure and the network services). Again, Generalized Stochastic Petri Nets have been employed in [27] to face availability problems in data centers in charge of managing SFCs.

Another variant of SPN named Stochastic Reward Net (SRN) is adopted in [28], where a comparative availability analysis among different containerized IMS deployments (homogeneous, heterogeneous, co-located) allows pinpointing the optimal IMS redundant configuration in terms of availability/cost trade-off. SRNs have been profitably adopted also in [29], where the probabilistic behavior of some cloud components has been modeled. Again, Stochastic Activity Networks (SANs) have been adopted in [30] and [31] aimed at a quantitative assessment of network factors affecting the availability of services provided by NFV architectures.

We employ RBDs to model the high-level connections among the "chained" 5G nodes, and the SRN formalism to capture the probabilistic failure/repair behavior of each 5G node that can be represented as a 3-layered structure. Remarkably, two enhancements can be highlighted. First, we create a *pipelined* system where the optimal number of containerized instances (m^*) able to guarantee a constrained overall latency is used as input to build the so-called *reward function* useful to evaluate the SRN model. Then, through the employment of *coverage factors* into the SRN model of a 5G node, we analyze different availability behaviors due to: *i*) controlled reboots (a.k.a. rejuvenation [32], [33]), meaning that a node can be periodically rebooted at a convenient time (e.g., when there is little or no load in the system) to mitigate the software aging phenomenon; *ii*) necessary reboots, due to unavoidable failures (e.g., caused by bugs) that make the node to be recovered.

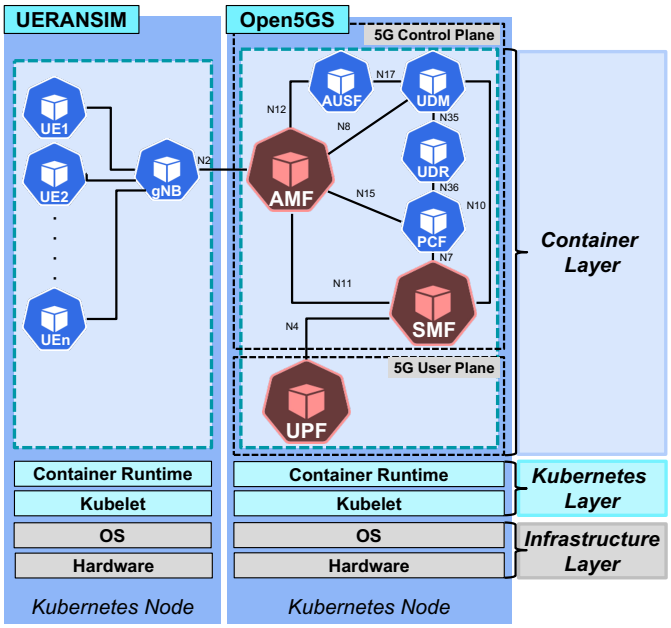


Fig. 1: Kubernetes-based Open5GS testbed implementing a complete 5G network infrastructure with the radio part (on the left) implemented with UERANSIM (UE + RAN part), and the core part (on the right) implemented with Open5GS.

III. Testbed based on the Open5GS architecture

Open5GS is one of the most complete (open-source) projects implementing the core part of a 5G network infrastructure compliant with the Release 17 of 3GPP [34], and able to interact with emulated/simulated network radio environments. Open5GS is divided into two segments: the control plane and the user plane. The former is in charge of managing sessions, handling mobility and paging procedures, and configuring bearers. The latter manages data packets transferred between user equipment and external WANs. Figure 1 shows a simplified representation of our testbed where the core part (on the right) is implemented with Open5GS, and the radio part (on the left) is implemented with UERANSIM [35]. This latter is an open-source simulator of user equipment (UE) and Radio Access Network (RAN). In particular, UERANSIM provides a simulation of the *gNodeB*, a node that can be physically deployed as a *tower* or virtualized through software-defined radio, and in charge of supervising operations such as access control, handovers, dynamic resource allocation, etc.

For the core part, it is possible to pinpoint three main nodes (highlighted in red in Fig. 1):

- *Access & mobility Management Function (AMF)*: it acts as an interface between the radio part and the core part by controlling which UEs can access the core network and exchange data traffic. Among the others, main AMF functionalities include registration management (permitting/denying UEs to register/de-register with a 5G network), connection management (establishing signaling connections with UEs), and mobility management (tracking the UE's location and governing handovers).

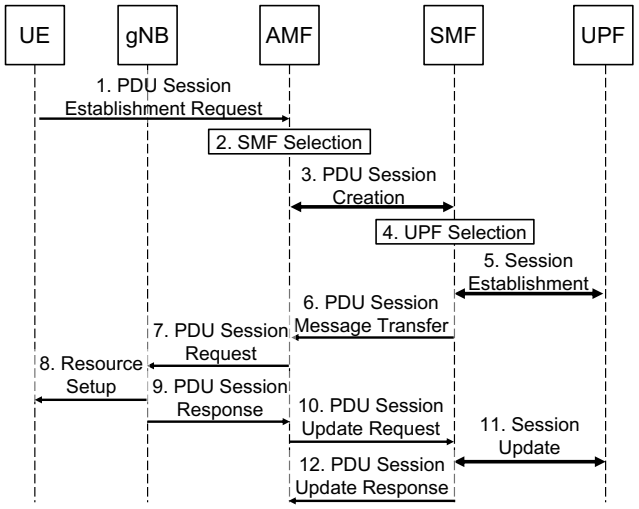


Fig. 2: Simplified sequence diagram of a PDU session establishment (PSE) procedure.

- *Session Management Function (SMF)*: it acts as an interface between control and user planes, and manages the whole lifecycle of a PDU session, being a logical connection between the UE and the data network (e.g., an internet connection). The most appropriate SMF node to manage the PDU session is selected by the AMF.
- *User Plane Function (UPF)*: it acts as an anchor point for PDU sessions, and offers crucial functionalities such as: packet routing and forwarding, traffic filtering, QoS handling, rate limiting, traffic usage reporting, etc.

Other nodes present in the Open5GS architecture are: the *Authentication Server Function (AUSF)* responsible for managing UE authentication procedures, the *Unified Data Management (UDM)* managing user's subscription data contained in the *Unified Data Repository (UDR)*, and the *Policy Control Function (PCF)* in charge of providing policy rules to SMF and AMF based on the data stored into the UDR.

In our testbed, all the nodes of Open5GS have been deployed onto Kubernetes, one of the most popular container orchestration systems [36]. Accordingly, the functionalities of 5G nodes have been virtualized through the container technology as schematically depicted in Fig. 1, where three layers can be recognized:

- *Infrastructure layer*, which embodies all the physical equipment (CPU, RAM, etc.) and the operating system (i.e., Linux);
- *Kubernetes layer*, which includes the *container runtime*, i.e., the software responsible for running containers (e.g., *containerd*), and the *kubelet*. This latter is an agent responsible for managing the state of containers including allotted resources, healthy conditions, etc.
- *Container layer*, which includes the containerized 5G nodes. Remarkably, each containerized node can be made of one or more containerized instances (AMF instances, SMF instances, etc.) working in parallel to reduce the time of processing, implying a performance improvement.

Our experimental setup encompasses 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). These machines host a complete Kubernetes cluster with 1 control plane node (first machine) and 2 worker nodes (second and third machines). The control plane node includes the fundamental services for Kubernetes orchestration (e.g., *kube-controller-manager*, *kube-scheduler*, *apiserver*, network manager, etc. [46]). The two worker nodes host UERANSIM (radio part) and Open5GS (core part). Each server machine runs Ubuntu v22.04.3 LTS, and Linux kernel v5.15. Moreover, we use Kubernetes v1.28.2 and *containerd* v1.6.24 (as container runtime).

Aimed at providing more details about the interoperability of the three main 5G nodes (i.e., AMF, SMF, and UPF), we depict in Fig. 2 the simplified sequence diagram of a PDU session establishment (PSE), representing the most common operation performed by a UE when requesting a 5G service. By assuming that a UE has correctly accessed the radio access network through the gNB, the first message is sent by UE to the AMF as a request to initiate a PDU session. Then, the AMF selects the most appropriate SMF to initiate the session, which, in turn, contacts the UPF for the session establishment. At this point, the PDU session message is back-propagated to the UE for the resource setup and the session can effectively start. As mentioned before, the performance of the PSE procedure has a direct impact on the QoS of the network and the QoE of end users [7], being connected to the overall delay introduced by the traversed nodes. In particular, the delay introduced by each node can affect negatively such a crucial procedure which must be repeated in case a time threshold is exceeded. Such a threshold is known as the $T3580$ timer [37].

Accordingly, we provide a method to estimate the overall delay associated with the PSE procedure that we elect as our performance metric, as it will be clear in the next section.

IV. Performance of Open5GS: a queueing networks perspective

Common operations in telecommunication systems (e.g., session establishment, session management, routing calls, etc.) are executed sequentially by a series of nodes often virtualized. The result is a logic chain (today realized through Service Function Chaining) where a request processed by a given node represents the input of the next node for further processing. This implies that a problem on a given node (i.e., an anomalous delay in processing a request) unavoidably affects the performance of the whole chain of nodes. At this aim, we employ the queueing networks theory due to its ability to model the performance of complex systems involving a flow of entities.

More precisely, we will cope with the *non-product-form* queueing networks, which are particularly suited to model real-world systems. For this class of queueing networks, the classic assumptions (e.g., exponentially distributed service times) are typically violated, since each node is better modeled as a $G/G/m$ queue, where we recall that the first G denotes a generic distribution for inter-arrival times, the second G denotes a generic distribution for service times, and m represents

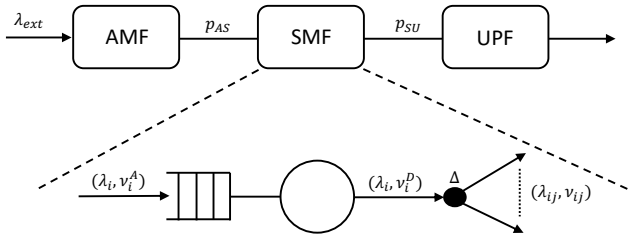


Fig. 3: Schematic representation of 5G core network nodes, where each node is analyzed via decomposition method (non-product-form queueing networks).

the number of containerized instances working in parallel on a given node. Among the available numerical solutions to solve non-product-form queueing networks, we adopt the *decomposition method* [38]–[40] which offers a flexible way for the approximate performance analysis, where the queueing network is broken up into subsystems which are then analyzed. It comprises two phases: in the first one, an iterative procedure is employed to calculate the squared coefficients of variation ($\sigma(\cdot)/E(\cdot)$) of inter-arrival times at each node; in the second phase, we employ an approximation formula (the Allen-Cunneen formula) to derive the mean sojourn time, a well-known performance indicator useful to estimate the mean total time that a request spends at a node.

A. Decomposition Method Applied to a 5G Queueing Network

Figure 3 shows a schematic representation of the 5G core network where the 3 crucial nodes (AMF, SMF, and UPF) are interconnected to form a chain that will be modeled with the queueing network formalism. The routing probabilities p_{ij} to route a flow from node i to node j are indicated by p_{AS} (routing probability from AMF to SMF) and by p_{SU} (routing probability from SMF to UPF). The i-th node is modeled as a queueing system with the following set of parameters: λ_i is the rate of the arrival process at the node i, $\nu_{A,i}$ and $\nu_{D,i}$ are the squared coefficients of variation of the inter-arrival and of the inter-departure processes at node i, respectively, λ_{ij} and ν_{ij} are the arrival rate and the squared coefficient of variation of the subflow process between node i and node j. The external arrival rate at the AMF (which represents the first contact point between the radio and the core part of 5G networks) is denoted by λ_{ext}. Moreover, the coefficient of variation of the inter-arrival times at the AMF depends on the ingress traffic distribution and, in case of Poisson, is set to $\nu_{A,1} = 1$.

The calculation of squared coefficients of variation of inter-arrival times at remaining nodes (first phase of the decomposition method) is performed iteratively (see Algorithm 1) through the following steps:

1) Merge several arrival processes to a given node into a single arrival process and evaluate the pertinent intermediate squared coefficient of variation of inter-arrival times, viz.

$$\nu_{A,j} = \frac{1}{\lambda_j} \sum_{i=1}^{N} \nu_{ij} \lambda_i p_{ij} \quad (j = 2, 3), \tag{1}$$

Algorithm 1: Open5GS network queue decomposition

Input: λ_i, μ_i, m_i, p_{ji},$\nu_{A,1}$, $\nu_{S,i}$, $\epsilon > 0$
ν_{ij}=1 % initialization
for i=1 ... N **do**
  evaluate $\nu_{A,j}$ from (1)
  evaluate $\nu_{D,i}$ from (2)
  evaluate ν_{ij} from (3)
  update $\nu_{A,i}$
  if $\nu_{A,i}(new) - \nu_{A,i}(old) < \epsilon$ **then**
    exit loop;
  end
end
% **Output**: $\nu^*_{A,i}$

where N is the number of nodes (in our case $N = 3$ due to the presence of 3 nodes: AMF, SMF, and UPF). Moreover, as the literature suggests [42], we set $\nu_{ij} = 1$.

2) Calculate $\nu_{D,i}$ that depends on $\nu_{A,i}$ and on $\nu_{S,i}$, being this latter the squared coefficient of variation for the service time at the node i. For this step, several authors propose different approximation formulas. We adopt the formula of Pujolle [43]:

$$\nu_{D,i} = \rho_i^2(1 + \nu_{S,i}) + (1 - \rho_i)\nu_{A,i} + \rho_i(1 - 2\rho_i), \tag{2}$$

being $\rho_i = \lambda_i/(m\mu_i)$ the utilization factor at the node i, with μ_i the mean service rate.

3) Evaluate ν_{ij} at the "decomposition point" Δ (see Fig. 3) according to the following formula:

$$\nu_{ij} = 1 + p_{ij}(\nu_{D,i} - 1). \tag{3}$$

The ν_{ij} values are used as input to evaluate new values of $\nu_{A,i}$ iteratively. Iterations will stop when no appreciable variation of coefficients $\nu_{A,i}$ are observed, and the value $\nu^*_{A,i}$ is the output. Being each single Open5GS node modeled as a $G/G/m$ queue, we have now to calculate the elected performance measure, namely the mean time that a request (in our case a PDU session request) spends at the node i, viz.

$$T_i = W_i + \frac{1}{\mu_i}, \tag{4}$$

being W_i the mean time that a request spends in the queue, and $1/\mu_i$ the mean time that a request spends to be served, at the node i. It is useful to remark that, for $G/G/m$ queues, no analytical solution exists, thus we have to employ an approximated formula. We exploit the Allen-Cunneen formula [41] for $G/G/m$ queues which allows deriving a good approximation for W_i:

$$W_i \approx \frac{P_{m,i}}{(m_i\mu_i - \lambda_i)} \frac{(\nu^*_{A,i} + \nu_{S,i})}{2}, \tag{5}$$

where $P_{m,i}$ is the steady-state probability of a node i modeled as a classic $M/M/m$ queueing system. If we put (5) in (4), we completely characterize the total mean time spent at the node i. Now, the mean time spent across the whole chain of nodes can be reasonably approximated with the mean time required

to complete a PSE procedure. Such a quantity can be derived by applying the Little's theorem as follows [42]:

$$T_{PSE} = \frac{1}{\lambda_{ext}} \sum_{i=1}^{N} \lambda_i T_i, \tag{6}$$

that, in view of (4) and (5) can be rewritten as:

$$T_{PSE} \approx \frac{1}{\lambda_{ext}} \sum_{i=1}^{N} \left(\frac{\lambda_i P_{m,i}}{(m_i \mu_i - \lambda_i)} \frac{(\nu^*_{A,i} + \nu_{S,i})}{2} + \frac{\lambda_i}{\mu_i} \right). \tag{7}$$

B. A Performance-related Optimization Problem

We can note that (7) is basically influenced by the mean time W_i. In particular, in view of (5), a decrease in time W_i is associated to a performance improvement that can be achieved by: reducing the arrival rates λ_i (difficult to control since they depend on the users needs), increasing the services rate μ_i (e.g., by assigning additional computational resources), or increasing the number of containerized instances m_i which is typically implemented in scalable container-based architectures where more instances can be added according to specific needs. Thus, an optimization problem to find the optimal number of containerized instances to be deployed onto the whole $5G$ chain aimed at keeping T_{PSE} under a given threshold can be set as follows:

$$\text{minimize} \sum_{i=1}^{N} m_i \quad \text{subject to} \begin{cases} m_i \geq m_0, \\ \\ T_{PSE} \leq T^*, \end{cases} \tag{8}$$

where the first constraint is introduced to impose the queueing stability ($\rho_i < 1$), being $m_0 = \lfloor \lambda_i / \mu_i \rfloor + 1$ ($\lfloor \cdot \rfloor$ is the integer round-down operator). The second constraint is the threshold T^* that cannot be exceeded by the overall end-to-end mean time. To solve (8) we employ an exhaustive procedure inspired by server allocation problems in manufacturing networks [44]. Intuitively, we start with the smallest possible allocation of containerized instances per node. Then, at each iteration, we add one instance per node and we evaluate T_{PSE}. Pragmatically, the output of the optimization problem (8) will be the minimum number of instances per node (namely, m^*_i) which guarantees that $T_{PSE} \leq T^*$. Such a value will be used as input for the SRN reward function of the availability model, as it will be clear in the next section.

V. Availability of Open5GS: a Stochastic Reward Networks approach

Stochastic Reward Networks (SRNs) provide a powerful formalism to model, from a probabilistic viewpoint, the failure and repair events of a system. In practice, SRNs allow to compact the classic CTMC representation by avoiding the problem of the state space explosion affecting CTMCs especially when modeling real-world systems characterized by a huge number of states.

A. SRN Basic Notation

It is now useful to briefly introduce the symbology adopted by the SRN. Formally speaking, an SRN is a bipartite directed graph with the following entities: a *place* (represented by a circle) which denotes a specific condition such as a working or failed node. A place can host one or more tokens representing a holding condition. On the other hand, a *transition* (represented by a rectangle) denotes a specific action. In the SRN formalism, two types of transitions are admitted. The first type is the *timed* transition (represented by a thick and blank rectangle and indicated by "T") and encodes a temporal action ruled by an exponential random variable (a common assumption in the availability assessment [45]). Within an SRN, two timed transitions are introduced: one denoting a failure action (with failure rate α), and another one denoting a repair action (with repair rate β). The second type of transition is the *immediate* transition (represented by a thin and black rectangle and indicated by "t") and encodes an instantaneous action. For example, if the *Infrastructure* layer fails (see Fig. 1), the two upper layers (*Kubernetes* and *Container* layers) *immediately* fail as a consequence. In the SRN terminology, a transition (both timed or immediate) is *fired* when it causes the tokens to be moved from one place (e.g., a place denoting a working condition) to another one (e.g., a place denoting a failure condition). Another symbol encountered in the SRN formalism is the inhibitory arc (represented by a circle-headed curve from a place to a transition) whose presence indicates that the corresponding transition can fire if and only if the connecting place does not contain tokens. Once the SRN model of a system is designed, evaluating the corresponding instantaneous availability is straightforward. We have to preliminarily define the so-called *reward function* $R(t)$ which amounts to 1 when the system is in a working state, and to 0 when the system is in a failed state. Thus, it is possible to show [45] that the instantaneous availability is

$$A(t) = Pr\{R(t) = 1\} = E[Y(t)] = \sum_{s \in \mathcal{S}} r_s \cdot \pi_s(t), \tag{9}$$

where $\mathcal{S}$ is the set of feasible tokens distributions (a.k.a. *markings*), r_s (the reward rate) is the value of $R(t)$ in marking s, and $\pi_s(t)$ is the corresponding probability. The set of markings can be split into a subset of working states ($r_s = 1$) and a subset of not working states ($r_s = 0$). Obviously, to evaluate the availability, we are interested in the subset of working states.

B. SRN Availability Model of an Open5GS Node

Figure 4 shows the SRN model of the i-th three-layered Open5GS node. Let us start to analyze the evolution of such a model by considering that the node is fully working. It is now useful to focus on the *Container* layer (the left part of the model in Fig. 4). Place P_{cnt} (namely, the place representing a working condition of the *Container* layer) contains a number of tokens representing the minimum number of containers (m^*_i, derived from performance analysis in the previous section) needed to guarantee a given performance target. As one container fails, transition T_0 is fired and one token is

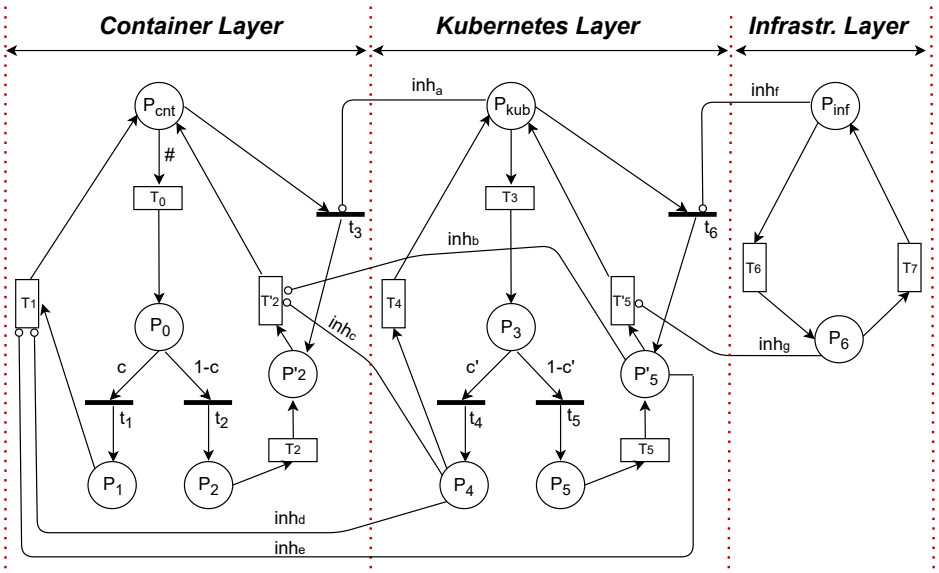


Fig. 4: SRN of a generic Open5GS node made of three layers: *i*) *Container* layer (providing the service logic), *ii*) *Kubernetes* layer (providing a software interface), *iii*) *Infrastructure* layer (providing hardware modeling).

moved from P_{cnt} to the place P_0. We note in passing that, T_0 is *marking-dependent* (indicated with the symbol # nearby) meaning that its rate is multiplied by the number of tokens present in P_{cnt}. We also note that P_0 is a transient place, that, once reached, forces the token to follow two possible paths: *i*) with probability (or *coverage*) c, the token reaches the place P_1 once the immediate transition t_1 is fired, or *ii*) with probability $(1-c)$, the token reaches the place P_2 once the immediate transition t_2 is fired. The former path indicates that a container is ready to be rejuvenated (via transition T_1), namely, even if not strictly necessary, the container is rebooted to mitigate the software aging effects. The latter path indicates that the container continues to work for an extra time[1] ruled by transition T_2. In other words, in this second case, the node continues to work until a real failure occurs with a consequent moving of the token into the place P'_2. Once here, the transition T'_2 is fired to perform the *Container* layer reboot.

Let us now focus on the middle part of the SRN in fig. 4 representing the *Kubernetes* layer. After a quick inspection, it is possible to notice many commonalities with the *Container* layer. For instance, when the *Kubernetes* layer fails, the token in P_{kub} (namely, the place representing a working condition of the *Kubernetes* layer) is moved into P_3 once T_3 is fired. Similarly to the *Container* layer case, the token can arrive to P_4 through the immediate transition t_4 with coverage c' (and, then, ready to be rejuvenated via T_4), or to P_5 through the immediate transition t_5 with coverage $(1-c')$. In the latter case, the *Kubernetes* layer is still working until the token reaches P'_5. Once in this phase, Kubernetes is ready to be rebooted (due to occurred failures) once the transition T'_5 is fired.

Some additional considerations are needed. First, as one token is moved from P_{kub} (namely, the *Kubernetes* layer fails), the inhibitory arc inh_a activates the immediate transition t_3 with the consequence that the *Container* layer is forced to fail (namely, token(s) in P_{cnt} are immediately passed to P'_2). Moreover, when *Kubernetes* layer is down (namely, the corresponding token is either in P_4 or in P'_5), the *Container* layer cannot work until the *Kubernetes* layer is restored. Such a condition is specified through 4 inhibitory arcs (inh_b, inh_c, inh_d, and inh_e) aimed at blocking transitions T'_2 and T_1. Similar reasoning occurs when analyzing the rightmost part of SRN in fig. 4 which depicts the *Infrastructure* layer. In case of a physical failure, in fact, the token in P_{inf} is transferred into P_6 via transition T_6 and can come back into working place after a hardware repair ruled by transition T_7. Likewise, when the *Infrastructure* layer is down, *Kubernetes* layer is forced to fail (since inh_f activates t_6) and cannot be repaired (being T'_5 blocked by inh_g) until the infrastructure repair. Now, we have to evaluate the availability of a single 5G node by specializing (9). In particular, we have to build the reward rate per node i, by pinpointing the markings corresponding to a working state of the node. By inspecting Fig. 4, we can notice two facts. First, a node works when the *Container* layer (which provides the service logic of the node) works; if the *Container* layer fails but *Kubernetes* and *Infrastructure* layers continue to work, the node remains not able to provide its service. Second, if we focus on the *Container* layer, we can safely say that such a layer is working when we have tokens or in P_{cnt} or in P_2. It means that P_{cnt} must contain at least m_i^* tokens (being m_i^* the minimum number of containerized instances guaranteeing the performance requirement), and P_2 must have at least one token. We note in passing that, including P_0 into the reward function is irrelevant, being P_0 a transient place acting as a *selector* ruled by t_1 and t_2. Now, taking the limit for $t \to \infty$ to obtain the steady-state availability (namely, the availability under regime condition) we can rewrite (9) for

[1]We reasonably set to 200 hours the extra times ruled by both transitions T_2 and T_5.

node i as follows:

$$A_i = \sum_{s \in \mathcal{S}} r_{s,i} \cdot \pi_{s,i}, \tag{10}$$

with

$$r_{s,i} = \begin{cases} 1 & \text{if} \quad \left(\sum_{i=1}^{N} \odot P_{cnt} \geq m_i^*\right) OR \left(\odot P_2 \geq 1\right), \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Expression (11) represents the reward rate associated to the i-th node with the pertinent steady-state probability $\pi_{s,i}$, while $\odot P_k$ indicates the number of tokens in place k.

C. RBD Formalism for Series/Parallel Structures

In the previous subsection, we have shown how to employ the SRN methodology to evaluate the availability of a single Open5GS node through (10). At this stage, we need to model: i) the high-level interconnections with other nodes as in the scheme depicted in Fig. 3 where a *series* structure can be recognized, and ii) the presence of *parallel* (redundant) replicas of nodes aimed at guaranteeing a certain resilience degree. To perform this task, we employ the RBD formalism that allows us to evaluate the availability of a series/parallel structure through straightforward combination rules, viz.

$$A^{(\sigma)} = \prod_{i=1}^{N} A_i \ , \qquad A_i^{(\pi)} = 1 - \prod_{r=1}^{R} (1 - A_r), \tag{12}$$

where $A^{(\sigma)}$ is the availability of a series of N interconnected nodes, whereas $A_i^{(\pi)}$ is the availability of a single node replicated R times.

VI. Experimental Results

This section contains two parts. In the first part, we provide details about the estimation of metrics related to performance (e.g., service times) and availability (e.g., rejuvenation and reboot times). Most of the metrics have been directly derived from experiments executed on our testbed, whereas other metrics have been derived from technical literature. In the second part, we assess the optimal set of 5G resilient settings through: i) a performance evaluation useful to derive the optimal vector of containerized instances per node (m^*), ii) a steady-state availability evaluation useful to derive a set of 5G resilient settings which fulfill a given availability constraint (five nines) with the minimum number of redundant elements.

A. Estimation of Performance and Availability Metrics

We recall that, as the principal performance metric, we have elected the PSE interval time which, in turn, is directly related to the $T3580$ timer, the threshold that the PSE procedure should not exceed to guarantee a satisfactory service.

To estimate the mean service time per node $1/\mu_i$ ($i = 1, 2, 3$), we have performed 100 trials with UEs simulating 30 concurrent PSE procedures per trial. Then, we have analyzed: i) the log files provided by the AMF (the most crucial node involved in the PDU session establishment procedure) and, ii) the corresponding Wireshark captures of the Next-Generation Application Protocol (NGAP) able to track the messages exchanged between radio and core part of a 5G network. By averaging such information, we have reasonably estimated $1/\mu_i$ around 90 ms with a standard deviation of around 50 ms.

On the other hand, availability metrics include: failure rates α_{cnt}, α_{kub}, and α_{inf} for the *Container*, *Kubernetes*, and *Infrastructure* layers, respectively. Such failure rates have been derived from the technical literature [47]. As regards repair rates, we have to distinguish repairs due to rejuvenation (*rej*) and repairs due to necessary reboots (*reb*) as a consequence of a failure. Accordingly, we have two repair rates for the *Container* layer ($\beta_{cnt}^{(rej)}$ and $\beta_{cnt}^{(reb)}$), two repair rates for the *Kubernetes* layer ($\beta_{kub}^{(rej)}$ and $\beta_{kub}^{(reb)}$), and one repair rate for the *Infrastructure* layer (β_{inf}).

Notably, repair rates have been estimated through the employment of the so-called *fault injection* techniques consisting in artificially provoking a fault of a layer, and then, measuring the corresponding repair time. We simulate software aging conditions caused by memory leaks (i.e., software bugs where unused memory is not deallocated, thus accumulating over time) and memory hogs (i.e., processes that cause over-consumption of memory due to incorrect capacity planning or configuration). Previous work showed that memory issues are the predominant cause of software aging [48]. Thus, to simulate software aging, we inject an additional workload that stresses the memory subsystem [49], by allocating large amounts of memory and performing a large volume of memory accesses. Large memory allocations put stress on the memory subsystem, since memory becomes fragmented (which slows down subsequent memory allocations), and forces the OS to move memory pages from and to the swap area on the disk. Moreover, memory accesses saturate the memory bandwidth, as in the case of memory hogs that force resource competition on the memory.

From our fault injection experiments, we found that rejuvenating a container can be significantly quicker than recovering a container from a software aging failure. Typically, we can perform rejuvenation at a convenient time (e.g., nightly, when the user workload is null or very low), when the container is not yet in a software aging state (i.e., a *proactive* approach). Thus, in this case, the container restarts quickly. However, if we neglect rejuvenation and perform the restart only when an effective failure occurs (i.e., a *reactive* approach), the restart of a container takes significantly longer time, since the system is overloaded both by software aging (i.e., memory leaks and hogs), and by the user workload (since the failure can occur at arbitrary time).

To the time for recovering a container from a software aging failure, we added an amount of time that is needed to detect the software aging failure. Differently from a "crash" failure (which is notified by the OS within a short amount of time), a software aging failure cannot be detected immediately: in a software aging failure, the system enters into a *degraded state*, in which the system still appears as available (e.g., the process is still running), but its performance (e.g., its response time) is significantly worse due to software aging (e.g., memory leaks and hogs). It takes a non-negligible

TABLE I: Performance and availability metric values

Metric	Description	Value
$1/\mu_i$	mean service time for node i	90 ms
$\nu_{S,i}$	coeff. of variation of service time of node i	0.309
T^*	threshold timer	0.5 s
$1/\alpha_{cnt}$	mean time for container failure	1258 hours
$1/\alpha_{kub}$	mean time for Kubernetes failure	2516 hours
$1/\alpha_{inf}$	mean time for infrastructure failure	60000 hours
$1/\beta_{cnt}^{(rej)}$	mean time for container rejuvenation	1 s
$1/\beta_{cnt}^{(reb)}$	mean time for container reboot	30 s
$1/\beta_{kub}^{(rej)}$	mean time for Kubernetes rejuvenation	2 s
$1/\beta_{kub}^{(reb)}$	mean time for Kubernetes reboot	60 s
$1/\beta_{inf}$	mean time for infrastructure repair	5 min
c, c'	coverage values	0.1 - 0.9

amount of time to detect such a degraded state, since the system administrator (either manually through a dashboard, or automated monitoring systems) needs to observe the system over a long period to confidently raise an alarm that a failure occurred. In [50], authors gathered evidence from industry practitioners indicating that automated detection takes a few tens of seconds. Consequently, we incorporate this additional time into the reboot time of a container (set at 30 seconds as indicated in Table I). Similarly, proactively restarting Kubernetes for rejuvenation takes less time than reactively restarting it for recovery, as both the containers and the Kubernetes processes take longer to restart due to the stress. Values of estimated performance and availability metrics are reported in Table I.

At this point it is also useful to highlight some limitations pertaining to the parameters estimation. As regards performance metrics (mean service times per nodes), it is useful to remark that we operate into a simulated environment (no real network operators typically provide performance data) that unavoidably introduces simplifications (e.g., limited number of trials, simplified nodes implementation). A simulated environment has also an impact onto the estimation of availability metrics (in particular, onto repair rates), since reboot operations may be affected by exogenous factors (e.g., nodes overload, temporary congestions) which characterize real environments and that we cannot easily deal with. This notwithstanding, the validity of our framework remains unaltered and independent from the particular estimation of numerical parameters.

B. Performance Evaluation

In this part of the experiment, we demonstrate how to determine the optimal composition of resilient 5G architectures fulfilling performance and availability requirements at the same time. Since performance and availability assessments are designed to be applied in *pipeline* (the output of the performance assessment represents the input of the availability assessment), we find it useful to start from the performance analysis which involves service times.

The estimated mean service times per node $1/\mu_i$ are used to calculate W_i from (5) in the following way. We start by evaluating the queueing stability per node (the first constraint in (8)) with the smallest allocation of containerized instances per node ($m_i = 1$). In the hypothesis that such a constraint is not violated ($m_i \geq m_0$), we launch the Algorithm 1 with

TABLE II: Performance results

$m = (m_1, m_2, m_3)$	$m_i \geq m_0$ (see (8))	$\nu_A^* = \left(\nu_{A,1}^*, \nu_{A,2}^*, \nu_{A,3}^*\right)$	$T_{PSE} \leq T^*$ (see (8))
$(1, 1, 1)$	NO	–	–
$(2, 1, 1)$	NO	–	–
$(2, 2, 1)$	NO	–	–
...	NO	–	–
$(3, 3, 3)$	OK	$(1, 0.2672, 0.1939)$	NO
$(4, 3, 3)$	OK	$(1, 0.5878, 0.2259)$	OK

$m_i = 1$ as input. After a few iterations, the algorithm returns stabilized inter-arrival time values $\nu_{A,i}^*$ that, along with m_i and $1/\mu_i$, allow evaluating T_i from (4), and then T_{PSE} from (6). In case the second constraint of (8) is not violated ($T_{PSE} \leq T^*$) we obtain the desired T_{PSE} value. Otherwise, we start to increase m_i node by node ($m_i \rightarrow m_i + 1$) and we re-launch the Algorithm 1.

We note that the estimated value for T_{PSE} (in the order of hundreds of milliseconds) does not take into account propagation delays, traffic congestion, and other possible phenomena occurring in real 5G networks. As regards the threshold T^* (defined in Sect. IV-B), we recall that such a value should play the role of the timer $T3580$ whose value, from 3GPP technical specifications [51], is in the order of ten seconds. This notwithstanding, in line with the aforementioned considerations on the absence of propagation delays and congestion in our testbed, we reasonably set T^* to 0.5 s.

The first set of results related to the performance assessment is reported in Table II, where we have considered $\lambda_{ext} = 30$ req/s and a unitary squared coefficient of variation of the traffic inter-arrival times at the ingress of the chain. The first column of the table contains the vector of allocated instances per node where m_1, m_2, and m_3 denote the allocated instances for the AMF, the SMF, and the UPF, respectively. For example, the vector $(2, 1, 1)$ means that the AMF has 2 containerized instances working in parallel, whereas both SMF and UPF have only 1 containerized instance working. The second column contains the binary value (NO / OK) pertaining to the satisfaction of the first constraint in (8) related to the qeueuing stability. In case queueing stability is not achieved, we increase the number of allocated instances per node until such a requirement is satisfied and, then, we launch Algorithm 1. The third column contains the vector of coefficients of variation of the inter-arrival times per node in output from the Algorithm 1. The fourth column contains the binary value (NO / OK) pertaining to the satisfaction of the second constraint in (8) related to the time threshold, thus, the resulting T_{PSE} value represents the estimated time to perform the whole PSE procedure. Notably, the penultimate row in Table II displays the configuration $m = (3, 3, 3)$ which satisfies the queueing stability constraint but not the second constraint of (8), since the obtained value T_{PSE} exceeds T^*.

Conversely, the optimal configuration (see the last row of Table II) of allocated instances satisfying both first and second constraints of (8) is represented by $m^* = (4, 3, 3)$. The corresponding vector containing the coefficients of variation of inter-arrival times $\nu_A^* = (1, 0.5878, 0.2259)$ is obtained after 3 iterations of the Algorithm 1 (at the fourth iteration we obtain the same value, thus we stopped at the third). With this configuration, we obtain $T_{PSE} = 0.462$ s. We note that

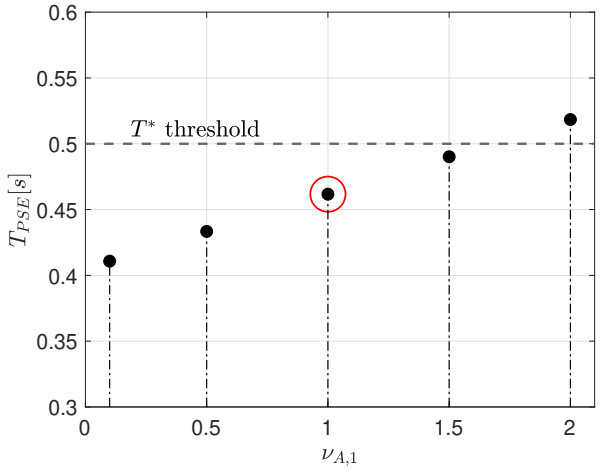


Fig. 5: Effect of the coefficient of variation of inter-arrival ingress traffic (at the AMF node) onto T_{PSE}.

such a value has been calculated by considering a Poisson distributed traffic at the ingress of the AMF node, where the coefficient of variation $\nu_{A,1}$ is unitary by definition. Since the adopted approximations for qeueuing models allow us to deal with generic inter-arrivals, we find it useful to analyze the behavior of T_{PSE} as $\nu_{A,1}$ varies. Obviously, the analysis requires running again the Algorithm 1 with different values of $\nu_{A,1}$ associated to non-Poisson distributed arrival traffic. Figure 5 shows the results of such analysis, where the case $\nu_{A,1} = 1$ has been elected as a benchmark (the point with the red circle). We note that for $\nu_{A,1} > 1$ performance degrades, and T_{PSE} moves close to the critical threshold T^* (for $\nu_{A,1} = 2$ we have $T_{PSE} > T^*$, thus the second constraint of (8) is violated). This behavior is due to the greater level of dispersion around the mean value, since the coefficient of variation is expressed as the ratio between the standard deviation and the mean value. Obviously, the problem does not arise for those inter-arrival distributions with $\nu_{A,1} < 1$, including, for example, the subset of Gamma or Weibull distributions with the scale parameter equal to 1 and the shape parameter greater than 1.

C. Optimal Resilient 5G Settings

We are now ready to calculate the steady-state availability associated with each single node by evaluating the SRN of Fig. 4 where: the number of tokens in the P_{cnt} amounts to 4 for the SRN model of the AMF node, and amounts to 3 for the SRN models of both SMF and UPF nodes, in accordance to the optimal vector of allocated instances derived in the performance assessment above ($m^* = (4, 3, 3)$ and $\nu_{A,1} = 1$). As a tool for the SRN evaluation, we use the well-tested Sharpe software [52]. Before proceeding with the numerical evaluation of the steady-state availabilities, we recall that both *Container* and *Kubernetes* layers in the SRN model of Fig. 4 can be "repaired" according to two different ways: they can be rejuvenated to prevent/mitigate the software aging, or rebooted as a consequence of faults. Pragmatically, the possibility of rejuvenating a layer is left to the system manager where the following trade-off should be taken into account: rejuvenating too often is favorable to mitigate software aging, but it is unfavorable in terms of the service interruptions increase. We recall that coverage factors c and c' regulate the rejuvenation/necessary reboots for the container and *Kubernetes* layers, respectively. For example, for the SRN model of Fig. 4, $c = 0.9$ means that the *Container* layer will be rejuvenated in 90% of cases and, for the remaining 10% of cases, it continues to work until a failure will occur.

Accordingly, we find it meaningful to analyze 4 cases:

- *Case I*: this case corresponds to a *strong* rejuvenation with $c = c' = 0.9$, implying that both *Container* and *Kubernetes* layers are rejuvenated very often;
- *Case II*: this is an intermediate case with $c = 0.1$ (infrequent container rejuvenation) and $c' = 0.9$ (frequent Kubernetes rejuvenation);
- *Case III*: this is the opposite intermediate case with $c = 0.9$ (frequent container rejuvenation) and $c' = 0.1$ (infrequent Kubernetes rejuvenation);
- *Case IV*: this case corresponds to a *weak* rejuvenation with $c = c' = 0.1$, implying that both *Container* and *Kubernetes* layers are rarely rejuvenated.

Due to the impossibility of analyzing all possible combinations with intermediate values of c and c' (from 0.1 to 0.9), we have chosen such *extreme* 4 cases to highlight some interesting facts as it will be clear in the following. We note in passing that, the SRN model for each node must be evaluated 4 times with c and c' chosen in accordance with the 4 cases above.

We start to investigate the behavior of the instantaneous availability associated with a single node, by recalling that the steady-state availability (namely, the regime value of the availability) is obtained by evaluating the instantaneous availability at $t \to \infty$. For instance, Fig. 6 shows the instantaneous availability of the AMF node in accordance to (9), where each colored curve corresponds to a given case (black, red, blue, and violet curves for Cases I, II, III, and IV, respectively). We see that all curves start at the maximum value of availability ($A_{AMF}(t = 0) = 1$) since, initially, the node is completely working (namely, all the tokens lie into the place P_{cnt}). As time elapses, the steady-state value of the availability associated with the AMF (in accordance to (10)) is achieved after about 16 mins and amounts to: 0.9921 (Case I), 0.9917 (Case II), 0.9378 (Case III), 0.9334 (Case IV). Similar analyses are carried out for the two remaining nodes (SMF and UPF). Two interesting facts emerge: first, we note a "jump" between the first couple of cases and the second couple of cases. This is basically due to the stronger impact of the rejuvenation onto the *Kubernetes* layer with respect to the *Container* layer. Second, we note that the steady-state availability value of a single node is far from respecting the five nines availability threshold (0.99999). Remarkably, the steady-state availability value associated to the chain of nodes will be even lower in view of the application of the RBD series rule $A_i^{(\sigma)}$ (see (12)) which involves some products of quantities less than 1.

Consequently, to achieve the desired availability target, we need to add a controlled redundancy per node and evaluate a new availability value for the whole chain by exploiting the

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3404560

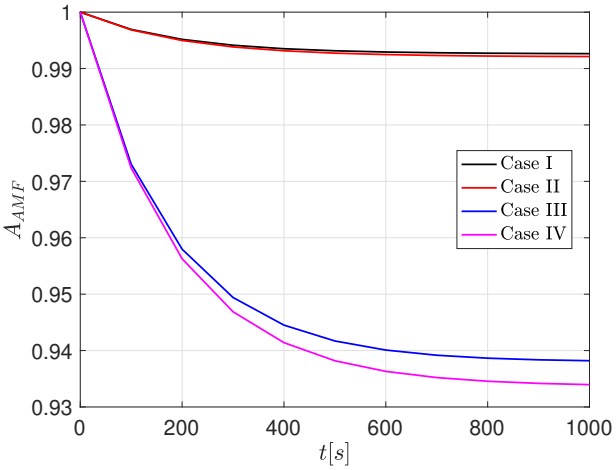


Fig. 6: Instantaneous availability associated to the AMF node (see (9)).

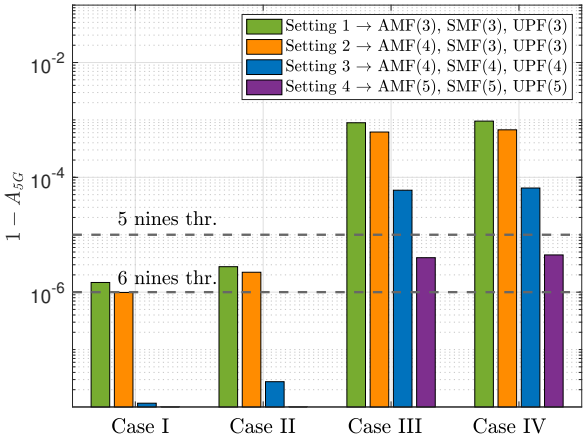


Fig. 7: Steady-state unavailability for the whole 5G chain for different cases and different redundancy settings.

RBD parallel rule $A_i^{(\pi)}$ (see (12)). Also, the redundancy control mechanism per node is implemented through a heuristic procedure by adding a new *replica* (namely, a replica of the whole SRN model in Fig. 4) per node, and evaluating the corresponding steady-state availability denoted by A_{5G}. The result is a set of possible 5G settings with each node having a given degree of redundancy.

Aimed at a valuable comparison, from this set, we extract four exemplary settings. The first setting is made of 3 replicas per node (AMF(3), SMF(3), UPF(3)). In the second setting, AMF is replicated 4 times and both SMF and UPF are replicated 3 times (AMF(4), SMF(3), UPF(3)). In the third setting, all the nodes are replicated 4 times (AMF(4), SMF(4), UPF(4)), and in the fourth setting, all the nodes are replicated 5 times (AMF(5), SMF(5), UPF(5)). The steady-state availability values of such settings are shown in the graph bar of Fig. 7 where on the x-axis we denote the four cases discussed above, and on the y-axis, for visualization comfort, we report the unavailability values associated with the four settings (green, orange, blue, and violet for settings 1, 2, 3, and 4, respectively). We also see two horizontal dashed lines at 10^{-5} and 10^{-6}, representing the five nines and six nines thresholds, respectively.

With this representation, it is very easy to pinpoint settings respecting a given threshold: bars (namely, settings) below the 5 nines line respect the five nines threshold and, even the six nine threshold in case bars are below the 6 nines line, as well. Some meaningful observations can be derived from such results. We start by analyzing Case *I*. We recall that this case implements an aggressive rejuvenation policy of both *Container* and *Kubernetes* layers, to extremely reduce the phenomenon of software aging, and to ensure correct functioning for the longest possible period. This directly translates into have a strong resiliency for all settings: setting 1 fulfills the five nines requirement ($A_{5G} = 0.9999985$), setting 2 barely fulfills the six nines requirement ($A_{5G} = 0.99999901$), and settings 3 and 4 exhibit very extremely high availability values amounting to $A_{5G} = 0.99999998$ (7 nines) and to $A_{5G} = 0.999999999908$ (10 nines), respectively. In practice, Case *I* guarantees very robust settings but is also high the risk of frequent service interruptions (aggressive rejuvenation strategy). A milder situation is encountered with Case *II*, where we can see that a weak rejuvenation policy for the *Container* layer does not have dramatic effects on the availability of settings. Interestingly, we observe an overturning for Cases *III* and *IV*, namely, the cases where the weak rejuvenation policy is applied for the *Kubernetes* layer. Such behavior reflects what we have already seen in Fig. 6, and confirms that the rejuvenation of the *Kubernetes* layer has a great impact on the whole structure. This occurs because a fault of Kubernetes directly implies a fault of all containers running on top. It means that for Cases *III* and *IV* we need settings with at least a 5-degree of redundancy per node to fulfill the five nines threshold. This could be unacceptable due to the high costs resulting from replicating all the layers many times. Remarkably, the impact of redundancy and rejuvenation cases can also be easily interpreted by considering the classic formula of steady-state availability: $A = MTTF/(MTTF + MTTR)$. Here, the numerical values of MTTF (mean-time-to-failure) and MTTR (mean-time-to-repair) for various parameters are derived from Table I. In practice, on one extreme (Case *I*), there is a high probability of rejuvenation, namely a great probability of having low MTTRs values for containers and Kubernetes, thereby pushing the steady-state availability towards 1. Consequently, even when redundancy is not so extensive (as in Setting 2, Case *I*), we are able to meet the challenging constraint of six nines. On the other extreme (Case *IV*), there is a high probability of no-rejuvenation, namely a high probability of having high MTTRs values for containers and Kubernetes. This results in the steady-state availability moving away from 1. In this latter case, to achieve the desired availability constraint (e.g., the five nines) we need to further increase the redundancy levels and associated costs.

From a system manager perspective, we can safely state that the best trade-off between rejuvenation and resiliency is offered by setting 1 in the Case *II* where we have: the minimum redundancy degree (3 replicas per node), a medium-aggressive rejuvenation policy, and the satisfaction of the five nines availability requirement.

VII. Conclusions and Future Prospects

We have presented a solution that combines performance and availability methodologies to characterize and evaluate a set of resilient virtualized 5G settings. In particular, performance aspects (related to delay issues) have been addressed through *non-product-form queueing networks* which allow for an approximate performance assessment to solve $G/G/m$ queue models adopted to characterize the 5G nodes (AMF, SMF, and UPF). Then, availability aspects (related to optimal redundancy issues) have been addressed through a hierarchical approach involving: *i*) Reliability Block Diagrams to model the high-level interconnections among 5G nodes, and *ii*) Stochastic Reward Networks to characterize probabilistically the failure/repair behavior of a single node, where we take into account rejuvenation procedures (to mitigate software aging) and necessary reboots (due to unpredictable failures). Most of the parameter values used in our performance and availability assessments have been directly estimated through a testbed based on the Open5GS framework. The resulting 5G settings can help to answer some interesting designing-related questions, such as: *Which parameters must be tuned to improve the performance (e.g., to reduce delay)*? *Which is the impact of the redundancy onto the availability constraint? Which is the consequence of an aggressive rejuvenation strategy?*

As future prospects, we may consider several directions to expand the results presented in this work. A first improvement pertains to a complete characterization of the radio part (eventually not emulated) to take into account even more realistic values of delays introduced by physical components (e.g., antennas). Then, for the three-layered structure of virtualized nodes, one can investigate different charge values to be assigned per layer to numerically quantify the cost associated with a replicated node. Finally, a more detailed characterization of the 5G core network could include additional nodes (e.g., Authentication Server Function, Policy and Charging Function, etc.) that have been not considered in this study.

References

[1] 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.

[2] M. Gharbaoui, C. Contoli, G. Davoli, D. Borsatti, G. Cuffaro, F. Paganelli, W. Cerroni, P. Cappanera, and B. Martini, "An experimental study on latency-aware and self-adaptive service chaining orchestration in distributed NFV and SDN infrastructures," *Computer Networks*, vol. 208, pp. 1–15, 2022.

[3] M. Niu, Q. Han, B. Cheng, M. Wang, Z. Xu, W. Gu, S. Zhang, and J. Chen, "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, 2022.

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

[5] ETSI GS NFV-REL 006. Available online: https://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/006/03.01.01_60/gs_nfv-rel006v030101p.pdf. (Accessed: Nov. 10, 2023).

[6] Ericsson, "How can network operations make 5G systems resilient?," [Online]. https://www.ericsson.com/en/blog/2021/9/5g-resilient-system-network-operations. (Accessed: Nov. 10, 2023).

[7] ETSI, "TS 101-563," [Online]. https://www.etsi.org/deliver/etsi_ts/128500_128599/128552/15.01.00_60/ts_128552v150100p.pdf. (Accessed: Nov. 10, 2023).

[8] 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, 2021.

[9] 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, 2019.

[10] R. Li, B. Decocq, A. Barros, Y.-P. Fang, and Z. Zeng, "Estimating 5G network service resilience against short timescale traffic variation," *IEEE Trans. Netw. Service Manag.*, 2023, doi: 10.1109/TNSM.2023.3269673.

[11] P. Kaliyammal 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, 2019.

[12] 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, 2022.

[13] 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, 2021.

[14] B. Shi, F.-C. Zheng, C. She, J. Luo, and A. G. Burr, "Risk-resistant resource allocation for eMBB and URLLC coexistence under M/G/1 queueing model," in *IEEE Trans. on Vehic. Tech.*, vol. 71, no. 6, pp. 6279-6290, 2022.

[15] W. Yoo, J. Yun, J. Jung, G. Hwang, and J.-M. Chung, "Optimized resource utilization scheme for real-time V2V sidelink unicast communication in 5G networks," in IEEE Wireless Communications Letters, 2023, doi: 10.1109/LWC.2023.3290048.

[16] 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, 2018.

[17] P. Mandal, "Comparison of placement variants of Virtual Network Functions from availability and reliability perspective," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 860-874, June 2022.

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

[19] H. Farooq, M. S. Parwez, and A. Imran, "Continuous Time Markov Chain based reliability analysis for future cellular networks," in *Proc. IEEE GLOBECOM*, 2015, pp. 1–6.

[20] L. Jiang, X. Chang, J. Bai, J. Misic, V. Misic, and Z. Chen, "Dependability analysis of 5G-AKA authentication service from server and user perspectives," *IEEE Access*, vol. 8, pp. 89562-89574, 2020.

[21] 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, 2023.

[22] L. De Simone, M. Di Mauro, M. Longo, R. Natella, and F. Postiglione, "Multi-Provider IMS infrastructure with controlled redundancy: a performability evaluation," *IEEE Trans. Netw. Service Manag.*, 2023, doi: 10.1109/TNSM.2023.3282745.

[23] 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.

[24] 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.

[25] 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, 2021.

[26] R. Li, B. Decocq, A. Barros, Y.-P. Fang, and Z. Zeng, "Petri Net-based model for 5G and beyond networks resilience evaluation," in *Proc. IEEE ICIN*, 2022, pp. 131–135.

[27] 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.*, doi: 10.1109/TNSM.2023.3242321.

[28] 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, 2021.

[29] E. Ataie, R. Entezari-Maleki, L. Rashidi, K. S. Trivedi, D. Ardagna, and A. Movaghar, "Hierarchical Stochastic Models for performance, availability, and power consumption analysis of IaaS clouds," *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 1039-1056, 2019.

[30] 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.

[31] 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.

[32] J. Bai, X. Chang, F. Machida, L. Jiang, Z. Han, and K. S. Trivedi, "Impact of Service Function aging on the dependability for MEC Service Function Chain," in *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 4, pp. 2811-2824, 2023,

[33] Y. Li, L. Li, J. Bai, X. Chang, Y. Yao, and P. Liu, "Availability and reliability of Service Function Chain: a quantitative evaluation view," in *Int. J. Comput. Intell. Syst.*, vol. 16, no. 52, 2023.

[34] 3GPP, "TR 21.917," [Online]. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3937. (Accessed: Nov. 10, 2023).

[35] UERANSIM, [Online]. https://github.com/aligungr/UERANSIM. (Accessed: Nov. 10, 2023).

[36] Kubernetes, [Online]. www.kubernetes.io/. (Accessed: Nov. 10, 2023).

[37] ETSI, "TS 124-501," [Online]. https://www.etsi.org/deliver/etsi_ts/124500_124599/124501/15.00.00_60/ts_124501v150000p.pdf. (Accessed: Nov. 10, 2023).

[38] P. J. Kühn, "Approximate analysis of general queuing networks by decomposition," in *IEEE Trans. Commun.*, vol. 27, no. 1, pp. 113-126, 1979.

[39] R. L. Disney, W. P. Cherry, "Some topics in queueing network theory ," in *Lecture Notes in Economics and Mathematical Systems*, vol. 98, pp. 23-44, 1974.

[40] P. J. Courtois, "Decomposability, instabilities, and saturation in multiprogramming systems ," in *Commun. of the ACM*, vol. 18, no. 7, pp. 371-377, 1975.

[41] O. Allen, *Probability, Statistics and Queueing Theory with Com- puter Science Applications.* . New York: Academic Press, 2 ed., 1990.

[42] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains*. New York: John Wiley & Sons, Inc., 2 ed., 2002.

[43] G. Pujolle and W. Ai, "A Solution for multiserver And multiclass open queueing networks," *INFOR: Information Systems and Operational Research*, vol. 24, no. 3, pp. 221–230, 1986.

[44] O.J. Boxma, A. Rinnooy Kan, and M. Van Vliet, "Machine allocation problems in manufacturing networks," *European Journal of Operational Research*, vol. 45, pp. 47–54, 1990.

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

[46] M. Barletta, M. Cinque, L. De Simone, and R. Della Corte, "Criticality-aware monitoring and orchestration for containerized industry 4.0 environments," *ACM Transactions on Embedded Computing Systems.* 2023

[47] 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.

[48] L. Vinícius, L. Rodrigues, M. Torquato, and F. A. Silva, "Docker platform aging: a systematic performance evaluation and prediction of resource consumption," *The Journal of Supercomputing,* vol. 78, no. 10, pp. 12898–12928.

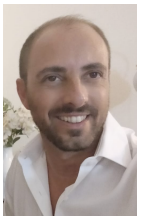
[49] C. King, "Stress-ng upstream project Git repository," [Online]. https://github.com/ColinIanKing/stress-ng. (Accessed: Nov. 10, 2023).

[50] M. Grottke, D.S. Kim, R. Mansharamani, M. Nambiar, R. Natella, and K.S. Trivedi, "Recovery from software failures caused by Mandelbugs," *IEEE Trans. Reliability*, vol. 65, No. 1, pp. 70-87, 2016.

[51] 3GPP, "Spec. 24.501," [Online]. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3370. (Accessed: Nov. 10, 2023).

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

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



Mario Di Mauro (Ph.D., SMIEEE'21) is an assistant professor of Telecommunications at the University of Salerno, Italy. Before joining the academia, from 2007 to 2012 he was with Research Consortium on Telecommunications (formerly Ericsson Lab Italy) as a Team Leader in industrial research. His main fields of interest include: network performance, network security and availability characterization, data analysis for novel telecommunication infrastructures. He authored more than 60 papers, mainly in the fields of network availability and security.

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

Fabio Postiglione (Ph.D.) is an associate professor of statistics at the University of Salerno, Italy. He received his Laurea degree (summa cum laude) in Electronic Engineering and his Ph.D. degree in Information Engineering from University of Salerno in 1999 and 2005, respectively. His research interests include statistical characterization of degradation processes, reliability and availability modeling of complex systems, and Bayesian methods. He has co-authored about 130 papers, mainly published in international journals.