IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 4, 2020, accepted July 5, 2020, date of publication August 3, 2020, date of current version August 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013597

Scalability and Performance Analysis in 5G Core Network Slicing

CARLOS HERNAN TOBAR ARTEAGA[1], (Member, IEEE), ARMANDO ORDOÑEZ[1,2], AND OSCAR MAURICIO CAICEDO RENDON[1], (Senior Member, IEEE)

[1]Grupo de Ingeniería Telemática, Universidad del Cauca, Popayán 190001, Colombia
[2]Ingeniería de Sistemas, Fundación Universitaria de Popayán, Popayán 190001, Colombia

Corresponding author: Carlos Hernan Tobar Arteaga (carlost@unicauca.edu.co)

This work was supported in part by the University of Cauca, in part by the Ministry of Science, Technology, and Innovation (Minciencias)—Colombia through the Ph.D. Scholarship under Grant 527-2015, and in part by the Postdoctoral Fellowship under Grant 80740-468-2019.

ABSTRACT Slicing the 5G core network involves specifying network services according to the functional and quality requirements of typical 5G use cases. In doing so, assessing the scalability and performance of network services plays a leading role, since it allows dimensioning their capacity. Since years ago, diverse approaches have proposed formal models to analyze network services' scalability, but they have not focused on the 5G core. Also, recent approaches have focused on analyzing network services' performance in the 5G core, but they have not investigated the scalability issue. In this paper, we propose a method, based on the Performance Evaluation Process Algebra (PEPA), aiming at enabling the systematic analysis of the performance and scalability of network services in the 5G core. We introduce new composite structures based on PEPA and intended to model and evaluate 5G network core procedures. We illustrate how to use our method by presenting the modeling and assessment of two services regarding scalability and performance metrics. The former corresponds to the session establishment in a 5G network slice. The latter corresponds to the user registration process in a network slice for Vehicle-to-Everything. Results show the usefulness of our method to model 5G core network services and dimension the capacity of slices that implement them. Furthermore, the validation results corroborate, in terms of accuracy, that our PEPA-based method measures performance metrics (throughput, average response time, and processor utilization) with negligible difference regarding a traditional approach like the Layered Queuing Network model.

INDEX TERMS 5G network slicing, network performance, PEPA, scalability.

I. INTRODUCTION

Network slicing is a central concept in the Fifth Generation (5G) communication systems, which aims at running multiple end-to-end logical networks (*i.e.,* encompassing core and access) as independent business operations on shared infrastructure [1]. 5G Network Slicing (5GNSL) envisions to support different use cases, such as enhanced Mobile BroadBand (eMBB), massive Internet of Things (mIoT), and Ultra-Reliable Low-Latency Communication (URLLC) [2]–[4]. Slicing the 5G core network involves specifying network services according to the functional and quality requirements of use cases above mentioned [5]. To specify 5G core network services before deployment tasks, the Network Service Providers (NSPs) need to analyze mandatorily under varying workloads the performance and scalability of 5GNSLs. This analysis is pivotal for NSPs dimension their capacity (*e.g.,* size of physical/virtual infrastructure).

Prados-Garzon *et al.* [6] present an approach to evaluate the performance and scalability of virtualized Mobility Management Entity (vMMEs) by using Queuing Networks and measuring the average response time. As this approach is 4G-oriented, it does not model the 5G core network that defines new Network Functions (NFs), such as Access and Mobility Management Function (AMF), Session Management Function (SMF), and Authentication Server Function (AUSF) [7]. Also, this approach does not measure some essential performance indices, such as processor utilization and throughput. In the 5GNSL literature, the approaches

The associate editor coordinating the review of this manuscript and approving it for publication was Wenjie Feng.

proposed by Trivisonno *et al.* [8], Campolo *et al.* [9], and Schneider *et al.* [10] use different techniques, such as simulations, emulations, and Queuing Petri Nets, to evaluate performance. However, these approaches do not analyze the scalability of 5GNSLs. The approaches afore-cited highlight the necessity of investigating the scalability of 5GNSLs located at the core network deeply. Furthermore, NSPs should adopt modeling and evaluation formalisms to perform such an in-depth scalability and performance investigation.

In this paper, we propose a method based on the Performance Evaluation Process Algebra (PEPA) for modeling, evaluating, and analyzing the scalability and performance of 5GNSLs in the core network systematically. PEPA is a modeling formalism useful to define distributed and composite systems in a systematic way. In this sense, we follow PEPA [11] to introduce new composite structures intended to model and evaluate 5G network core procedures. We illustrate how to use our method by two case studies: the session establishment 5GNSL and the user registration in the Vehicle-to-Everything (V2X) 5GNSL. The case studies focus on dimensioning the capacity of slices regarding users to attend and QoS requirements. We validate our method's accuracy with the Layered Queuing Network (LQN) modeling formalism. Results show that our method is useful to model 5GNSLs core procedures and dimension the capacity of slices. The validation results corroborate that our PEPA-based method measures performance, in terms of throughput, average response time, and processor utilization, with negligible difference regarding a traditional approach like LQN.

The contributions of this paper are:

- A method that introduces new composite structures based on PEPA and intends to model and evaluate 5G network core procedures.
- The scalability analysis in 5GNSLs considering concurrent users and the multiplicity of NFs.
- Two use cases that illustrate how to use our method for modeling, evaluating, and analyzing the performance and scalability in 5GNSLs.
- The dimensioning of 5GNSLs regarding concurrent users and QoS.

The remainder of this paper is organized as follows. Section II presents the related work. Section III details our PEPA-based method. Section IV and Section V evaluate the scalability and performance of the session establishment 5GNSL and the user registration V2X 5GNSL, respectively. Section VI provides some conclusions and implications for future work.

II. RELATED WORK

A. SCALABILITY AND PERFORMANCE ANALYSIS IN NETWORK SLICING

Trivisonno *et al.* [8] proposed an approach that aims at serving a massive number of Internet of Thing (IoT) connections efficiently. This approach allows modeling and evaluating an end-to-end IoT network slice for the device registration and core network bearer setup, which requires to connect diverse NFs, such as AMF, SMF, and AUSF. The authors by simulations measured the control plane signaling and the user plane traffic generated by devices accessing randomly, but they did not analyze the scalability of the IoT 5GNSL modeled.

Campolo *et al.* [9] presented a network slicing architecture to support V2X applications. This architecture introduces a service layer in which the core network comprises several NFs, such as AMF, AUSF, and Unified Data Modeling (UDM). The core network includes multiple AMF instances to manage the high signaling load generated by devices-mobility and avoid the increase in latency during slice registration procedures. The authors evaluated the user plane of a V2X slice by using the Mininet emulator [12] in which they measured latency, throughput, and packet drops. In contrast to this slicing architecture, we do not focus on the scaling and performance evaluation of the user plane, but in the control plane. In particular, here, we measure the throughput, processor utilization, and average response time in the registration procedure.

B. PERFORMANCE EVALUATION WITH MODELING FORMALISMS

Schneider *et al.* [10] introduced an approach that uses Queuing Petri Nets (QPNs) for specifying and analyzing virtual network services. In particular, the authors analyze the end-to-end delay, throughput, and queue lengths of a video streaming service. Network services following the proposed specification technique can be interpreted as scalable templates, where the behavior of each NF instance is formally specified using QPNs. The authors state that scaling-out new instances, their approach can specify how traffic is balanced between multiple instances of the same NF or whether incoming traffic from multiple instances is synchronized. However, this approach does not show a scalability analysis of network services. Unlike this study, we perform an in-depth scalability analysis for 5GNSL using the PEPA formalism.

Prados-Garzon *et al.* [6] proposed a solution that uses queuing networks to modeling and evaluating the performance of a Long Term Evolution (LTE) vMME hosted in a data center. In particular, this solution allows computing the number of vMME processing instances to provide a target system delay given the number of users in the system. Although the proposed solution allows evaluating the scalability of vMME, the current 5G architecture defines other NFs, such as AUSF and NF Repository Function (NRF), which should be involved in the analysis [7]. Moreover, this solution omits essential performance indices, such as throughput and processor utilization. In this paper, we focus on the scalability and performance of 5GNSLs by using PEPA.

C. PEPA-BASED SOLUTIONS

PEPA has been used to model and evaluate distributed systems. Hillston *et al.* [13] introduced an approach that uses

TABLE 1. Related Work - Summary.

Work	Modeling Formalism	Application Scenario	Scalability Analysis
Hillston *et al.* [13]	PEPA	Web application	Multiplicity of threads
Tribastone [14]	PEPA	Distributed application	-
Trivisonno *et al.* [8]	-	5G network slicing	-
Campolo *et al.* [9]	-	5G network slicing	-
Schneider *et al.* [10]	Queuing Petri Nets	Video streaming	-
Prados-Garzon *et al.* [6]	Queuing Networks	LTE-EPC	Multiplicity of vMME
Our proposal	PEPA	5G network slicing	Multiplicity of 5G NF instances

continuous PEPA models to represent large systems with multiple replications in components, such as clients, servers, and devices. From this approach, our method leverages replication to represent NF instances and processors that host them, the continuous approximation for efficient analysis of large systems, and the modeling patterns for processor computation and synchronous communication.

Tribastone [14] presented an approach that maps the Layered Queuing Network (LQN) model to PEPA; LQN is a traditional model for describing (software and hardware) systems with layers and resource contention. This approach uses PEPA to model the semantics of layered multi-class servers, resource contention, the multiplicity of threads, and processors. The authors validated the mapping approach through simulations regarding the accuracy in the translation of LQN to PEPA. The validation allowed concluding that PEPA-based methods are useful in the LQN context to build up efficient models with large component replications. From this approach, our method adopts two PEPA structures, namely fork/join synchronization, and the accuracy-based validation.

Table 1 summarizes the most relevant related work to our method, revealing several facts. First, 5GNSL lacks a method to analyze the scalability of slices in the core network regarding different configurations of NFs. Second, in 5G, scalability and performance assessment must use modeling formalisms to achieve efficient analysis and compositionality. In this sense, PEPA allows stochastic simulation and approximation techniques to analyze models efficiently with a large number of instances, threads, and processors related to NFs. Also, PEPA provides the composition principle that facilitates the modeling of systems with many orchestration alternatives as the 5G core network offers. Third, a scalability analysis needs to consider the multiplicity, threads, and processors of NF instances.

III. SCALABILITY AND PERFORMANCE ANALYSIS METHOD

This section describes our PEPA-based method that allows modeling, evaluating, and analyzing the performance and scalability of 5GNSLs regarding response time, throughput, and processor utilization. Thus, our method facilitates the dimensioning of 5GNSLs in the core network, which leads to avoiding the wasting of resources.

A. MODELING FUNDAMENTALS

5GNSL intends to slice the 5G system (access and core) into several logical networks to support different use cases with non-uniform Quality of Service (QoS) requirements. The 5G core network is a service-based architecture [7], in which NFs cooperate to perform signaling procedures, such as connection, registration, mobility management, and session management [15]. The connection management establishes and releases the signaling connection in the control plane. The registration management registers a user with the 5G system, and creates its user context, allowing the use of data services. The mobility management keeps track of the current location of User Equipment (UE), enabling the handover aware to radio conditions, load balancing, or QoS requirements. The session management controls the user plane functionality, such as packet routing and forwarding, packet inspection, and traffic steering. Session establishments enable that subscribers can use applications, such as browsing the web and advanced assisted driving.

Slicing the 5G core network involves composing NFs to perform the signaling procedures afore-described. Let us consider a session establishment 5GNSL. Fig. 1 presents the service chain of this slice, including NFs, service interfaces, and roles (*e.g.,* service consumer or service producer) taken by UE, AMF, SMF, and NRF. UE allows the user to connect to the 5G system and use its applications. AMF provides the communication service using the interface *Namf_Communication*. This service enables an NF to communicate with UE through Non-Access-Stratum (NAS) messages or the access network. *Namf_Communication* defines several operations, including *UEContextTransfer*, which offers the general registration procedure [16]. SMF supports, through its interface *Nsmf_PDUSession*, the establishment, modification, and release of data sessions between the User Plane Function (UPF) and the access network. Also, SMF configures traffic steering policies at UPF, allocates IP address to UE, and applies charging rules. NRF registers and discovers NFs. The discovery functionality maintains uniform resource locators (URLs), profiles, and services of NF instances available for composing service chains. Discovery and registration are offered by interfaces *Nnrf_NFDiscovery* and *Nnrf_NFManagement*, respectively [15].

Fig. 1 also shows the NFV Infrastructure (NFVI) of session establishment 5GNSL. In our example, NFVI indicates that each NF (AMF, SMF, and NRF) occupies a Virtual Machine (VM). However, other deployment alternatives are feasible; for instance, each VM can host several NF instances. The number of active NF instances can increase (scaling out) or decrease (scaling in) to face a growing or declining workload, respectively, to improve resource utilization. Moreover, the number of CPU cores can be dynamically allocated to

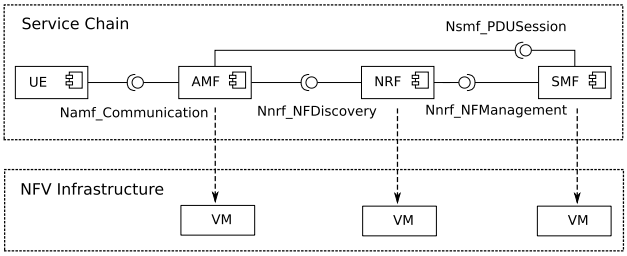


FIGURE 1. Service Chain and NFVI of session establishment 5GNSL.

the VMs to handle workload variations, leading to the scaling up (increasing CPU cores) or the scaling down (decreasing CPU cores). Scaling out/in (horizontal scaling) and scaling up/down (vertical scaling) allow adjusting the slice capacity to the workload dynamics.

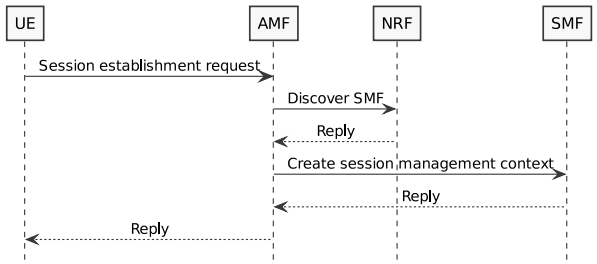


FIGURE 2. Session establishment in 5GNSL.

Fig. 2 describes the behavior of the session establishment 5GNSL by a sequence diagram. The session establishment includes an AMF, which serves as the single-entry point for a UE for all its communication. Once the user decides to use one application, for example, to browse the web, AMF needs to assign an SMF to manage the user session context. As NFs can be instantiated and deleted at any time, AMF needs to discover an available and suitable SMF via the NF Discovery operation performed between AMF and NRF. To accomplish successful discovery, SMF must register beforehand with NRF. It is noteworthy that the pair of messages (request-reply) in the communication between NFs indicates synchronous communication.

We perform some assumptions for modeling 5GNSLs close to reality. These assumptions are related to the 5G core network features, such as the communication pattern and the high number of concurrent users.

- For reducing complexity, we model 5G control plane procedures in a high-abstraction level without losing representativeness in their overall behavior.
- We represent UE and NFs by sequential PEPA-components because 5G control plane procedures operate sequentially and distributively.
- We model the communication between NFs by the client-server pattern [17] with synchronous and asynchronous [13] mode due to NFs can act as server and client. For example, UE sends requests to AMF (server). In turn, AMF (client) sends requests to SMF.
- We assume that each NF instance has a pool of threads to handle in parallel the myriad of concurrent incoming requests in the 5G control plane.

B. PERFORMANCE EVALUATION PROCESS ALGEBRA

PEPA is a language for the analysis of concurrent systems [18], such as the control plane of the 5G core network. PEPA offers formality, abstraction, and compositionality. Formality gives a precise meaning to all terms in the language. Abstraction allows building up complex models from components but disregarding their internal behavior; in this paper, we model NFs as PEPA components. Compositionality allows modeling the interaction between components by cooperation processes. A PEPA model is a composition of entities or components intended to perform actions sequentially. Actions can involve one (independent) or several components (composite) [19]. PEPA defines the following operators.

Prefix $(\alpha, r).P$ is the basic form to construct the behavior of components, such as AMF and SMF. $(\alpha, r).P$ carries out activity (α, r) that has a type α and a duration exponentially distributed with mean $1/r$ time units. $(\alpha, r).P$ subsequently behaves as P. A prefix allows capturing the behavior that involves precedence between two distinct activities. For example, SMF creates a session context to manage the use of applications by subscribers. Subsequently, SMF sends back a reply. The complete model of this sequential component is $(createSession, r_{cs}).Reply_{smf}$.

Choice $P + Q$ represents a system which may behave either as P or Q. Let us consider 5G multimedia services that support emergency sessions. To provide such services, AMF may contain information of an SMF configured statically, being unnecessary to request the discovery operation to NRF. In this way, AMF can be modeled as $(call_{nrf}, p \cdot r_{call}).Disc + (call_{smf}, (1-p) \cdot r_{call}).SC$. The choice operator enables actions $call_{nrf}$ and $call_{smf}$, which are executed with probabilities p and $(1-p)$, respectively. Once UE sends a session creation request, AMF may perform the discovery of an available SMF by using NRF or the session creation directly using an SMF for an emergency.

Constant AdefP models the cyclic behavior of an NF. Once an NF completes the operations requested by a client, the NF returns to the initial state to serve a new client. Let us model the behavior of the processing of the VM hosting SMF (SMFP) by using a constant. SMFP gets access to the processor by the action get_{smfp}, performs the action $createSession$, and returns to the initial state. As these actions are sequential and cyclic, SMFP can be modeled by $SMFPdef(get_{smfp}, r_p).(createSession, r_{cs}).SMFP$.

Cooperation $P \underset{L}{\bowtie} Q$ models the interactions between components by the synchronization of P and Q over the action types in the set L. All the other actions are performed independently. Let us consider that AMF cooperates with SMF for creating a user session context. This cooperation can be modeled by $(cs, r_{cs1}).(update, r_u).AMF \underset{\{cs\}}{\bowtie} (cs, r_{cs2}).(rep, r_r).$

SMF. In this cooperation, the two components perform the shared action *cs* (create session), subsequently behaving as $(update, r_u).AMF \underset{\{cs\}}{\bowtie} (rep, r_r).SMF$. Then, actions *update* (update state) and *rep* (generate reply) are carried out independently. The cooperation operator defines the overall system model and, so, it is also known as the system equation.

The cooperation operator is fundamental to analyze scalability, since it allows to specify a pool of threads working in parallel in a processor, as well as multiple NF instances for each NF type. In our approach, we note the number of threads per processor and the number of NF instances as N_t and N_{nf}, respectively. For scalability analysis, let us consider P and Q as the components that model two multi-thread, multi-instance *NFs*: NF_1 and NF_2, respectively. The cooperation between P and Q is modeled as $P[N_{nf_1} \cdot N_{nfp} \cdot N_t] \underset{L}{\bowtie} Q[N_{nf_2} \cdot N_{nfp} \cdot N_t]$. The product $N_{nf} \cdot N_{nfp} \cdot N_t$ is the total number of processors allocated to an NF instance. On the other hand, the product $N_{nf} \cdot N_{nfp}$ is the total number of processors allocated to an NF. Let us consider the outlined example again, the cooperation between components *AMF* and *SMF* may be defined as $AMF[N_{amf} \cdot N_{nfp} \cdot N_t] \underset{L}{\bowtie} SMF[N_{smf} \cdot N_{nfp} \cdot N_t]$, where N_{amf} and N_{smf} are the number of instances of AMF and SMF, respectively; N_{nfp} is the number of processor allocated to each NF instance of AMF and SMF, and N_t is the number of threads that each processor can handle.

C. MODELING PATTERNS

Communication. Our method supports the modeling of synchronous and asynchronous communication. The synchronous mode is represented by two sequential actions: $(req_{service}, v).(rep_{service}, v)$. The first one defines the service request. The second action represents the response. Here, $1/v$ is the expected duration of request and response actions in time units. Considering the running example (see Fig. 2), AMF requests the discovery service of SMF from NRF. The communication between AMF and NRF is modeled as $(req_{discovery}, v).(rep_{discovery}, v)$. Since the communication between NFs is almost instantaneous, we must set v to an enough high value like 100, 000. The asynchronous mode is modeled by a request action $(req_{service}, v)$. In contrast to the synchronous mode, the NF acting as the client is not blocked to receive or send other operations.

Processing on a VM. A VM hosting an NF uses processing to process a request. For modeling such processing, we use the pattern defined in [13], [20], in which a two-state sequential component models a single processing unit. The first state enables an action to obtain exclusive access to the resource, whereas the second state performs all the actions deployed on the processor. For example, the processing in NRF can be modeled as in Listing 1. $Nrfp_1$ (first state) gets access to the processor using the action (get_{nrfp}, r_p). $Nrfp_2$ (second state) performs the action $(discover, r_d)$. This action performs the discovery service by NRF.

Listing 1. NRFP.

$$Nrfp_1 \overset{def}{=} (get_{nrfp}, r_p).Nrfp_2$$
$$Nrfp_2 \overset{def}{=} (discover, r_d).Nrfp_1$$

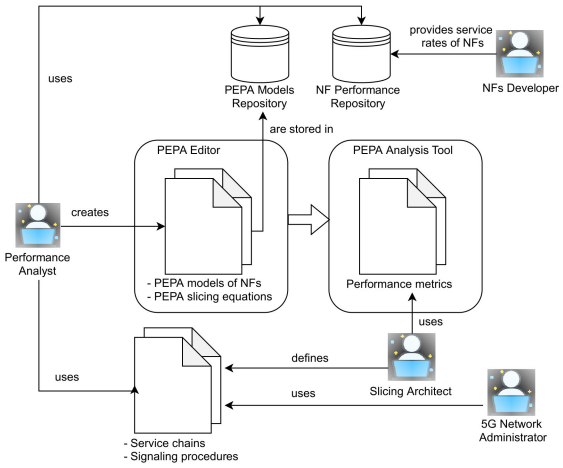


FIGURE 3. Method functioning - actors and tasks.

D. FUNCTIONING

Fig. 3 depicts the functioning, actors, and tasks of our method. The Slicing Architect defines the service function chains and their signaling procedures (*e.g.,* session establishment). Furthermore, the Architect uses the performance analysis results to select the appropriate scaling configuration (*i.e.,* type and the number of NF instances) to handle the expected workload.

The Performance Analyst creates PEPA models to represent NFs, specifying in the PEPA editor the equations that define the cooperation between NFs. It is noteworthy that the Analyst can reuse existing PEPA models and slicing equations available in the *PEPA Models Repository*. Furthermore, since NF models include service rates, the Analyst retrieves them from the *NF Performance Repository*. The Analyst also performs steady-state analysis of slices for measuring performance indices, such as average response time, throughput, and processor utilization. The PEPA Analysis Tool enables steady-state analysis. The 5G Network Administrator uses the service chain specifications to deploy them.

The NF Developer implements NFs and runs performance tests to determine the average service rates of operations, such as registering or discovering an NF in and from a repository. These rates are useful to model the expected duration of the operations. For example, a Developer can measure the service rates by generating N calls to the service by calculating the average time taken to generate responses. This Developer also records pairs of values that relate operations of NFs with service rates in the *NF Performance Repository*.

The state-space underlying PEPA models can exponentially grow when the number of components increases (state-space explosion problem), which brings a high

computational cost. Note that this problem is present in other analysis techniques, such as Petri nets, where the quantitative results depend on the numerical solution of the underlying Continuous-Time Markov Chain (CTMC) [21]. Conversely, traditional queuing networks do not suffer from this problem due to their low computational cost that adapts well to the size of the system under study [14]. However, queuing networks consider the exclusive possession of a resource (*e.g.,* network node) and do not model nested service requests that characterize client-server architectures. To address the state explosion problem, in this paper, we use the fluid approximation proposed in [22] that results in a set of Ordinary Differential Equations (ODEs), reducing the computational cost for solving PEPA models.

IV. CASE STUDY: SESSION ESTABLISHMENT 5GNSL

This case study aims three-folds. First, presenting the use of our method by modeling the session establishment 5GNSL. Second, dimensioning this 5GNSL to satisfy QoS requirements and avoid bottlenecks by its evaluation and analysis in terms of average response time, throughput, processor utilization, and scalability when NF instances and concurrent users vary. Third, validating our method with LQN.

We used the PEPA Eclipse plugin [23] to develop our method and performing a steady-state analysis of the session establishment 5GNSL. We deployed this plugin in an Intel Core i5 PC (1.7 GHz) with 4 GB of RAM.

A. MODELING

The session establishment is a primary task of the 5G core network that allows end-users to use session-based 5G applications. A session is the user activity carried out between the instant the user launches and closes a network application. Within a session, the application sends or receives all necessary data from performing tasks, such as download a web page, streaming a video, or make a call. The Inter-Arrival Time (IAT) is the time interval between the start of two consecutive sessions [6].

Let us recall the service chain (Fig. 1) and the message sequence (Fig. 2) of the session establishment 5GNSL. Modeling this slice comprises *1)* Defining service rates for UE, AMF, SMF, and NRF; these rates model the duration of actions in PEPA components. *2)* Composing new PEPA structures to model UE, AMF, SMF, and NRF; these structures use the PEPA operators and the modeling patterns to create sequential components. *3)* Modeling the interaction between PEPA components by the cooperation operator; this operator is also useful to define the overall system model (*i.e.,* the model of the session establishment 5GNSL).

Service rates. Table 2 presents the service rates (r) in UE, AMF, SMF, and NRF, as well as in processors of VMs that host these NFs. $1/r$ defines, in time units, the average execution demands for actions in NFs or processing on VMs.

PEPA components. We model UE, AMF, SMF, NRF, and the processing of VMs that host them as PEPA components.

TABLE 2. Service rates for modeling the session establishment 5GNSL.

Service rate	Description
r_{iat}	$1/r_{iat}$ (inter-arrival time) is the time that UE interposes between successive requests of session establishment
r_{pr}	$1/r_{pr}$ is the average time demand for preparing the service workflow
r_r	$1/r_r$ is the average time demand for preparing a response message by AMF
r_d	$1/r_d$ is the average time that takes the discovery service performed by NRF
r_{sc}	$1/r_{sc}$ is the average time that takes the session creation service performed by SMF
r_p	$1/r_p$ is the average execution demand on VM's processors
v	v is a high service rate that models almost instantaneous service calls and replies between NFs

We note the states with the name of corresponding NF (first letter in uppercase) and a sequential number (*e.g., Amf_1*). We noted the action types in lowercase (*e.g.,* discovery). We note action types by subscripts to add details. For example, the notation to access to the processor of AMF, call to an operation served by NRF, and service request and response for discovery is get_{amfp}, $call_{nrf}$, req_d, and rep_d, respectively.

Component *UE* models users requesting the session establishment to the network slice (see Listing 2). The behavior of UE is cyclic and interposes an IAT between successive requests. UE has two states. The first one (Ue_1) gets access to the processor using the action (get_{uep}, r_p), and then, performs the thinking action ($think, r_{iat}$). The second state (Ue_2) models the synchronous actions request and response that UE and AMF interchange to establish a session. These actions are (req_{se}, v) and (rep_{se}, v).

Listing 2. UE.

$$Ue_1 \stackrel{def}{=} (get_{uep}, r_p).(think, r_{iat}).Ue_2$$
$$Ue_2 \stackrel{def}{=} (req_{se}, v).(rep_{se}, v).Ue_1$$

Component *AMF* is the entry point for session establishment requests from UE. The action sequence is defined for a set of states Amf_i, $i = \{1, \ldots, 12\}$ as follows. Amf_1 models the service request. Amf_2 is a local action that prepare the service workflow. Amf_3 uses the choice operator and allows for the fork of the action sequence depending on regular or emergency requests. An occurrence probability of 0.95 modifies the rate for regular requests, whereas a probability of 0.05 does for emergency requests. Amf_4 to Amf_8 are for regular requests, Amf_4 starts a call to NRF for the discovery service. Amf_5 models the synchronous communication between AMF and NRF. Amf_6 starts a call to SMF for the creation of a session management context. Amf_7 represents the synchronous communication between AMF and SMF. Amf_8 prepares a message response to UE. Amf_9 to Amf_{11} are specific for emergency requests. In this case, AMF uses a statically configured SMF and does not require the discovery service by NRF. Amf_9 starts a call to SMF. Amf_{10} models the synchronous communication between AMF and SMF. Amf_{11} prepares a message response to UE.

Finally, the fork joins in Amf_{12}, which models the session establishment response. Listing 3 presents the PEPA-based model for AMF.

Listing 3. AMF.

$$Amf_1 \stackrel{def}{=} (req_{se}, v).Amf_2$$
$$Amf_2 \stackrel{def}{=} (get_{amfp}, r_p).(prepare, r_{pr}).Amf_3$$
$$Amf_3 \stackrel{def}{=} (choose, 0.95 \times v).Amf_4 + (choose, 0.05 \times v).Amf_9$$
$$Amf_4 \stackrel{def}{=} (get_{amfp}, r_p).(call_{nrf}, v).Amf_5$$
$$Amf_5 \stackrel{def}{=} (req_d, v).(rep_d, v).Amf_6$$
$$Amf_6 \stackrel{def}{=} (get_{amfp}, r_p).(call_{1smf}, v).Amf_7$$
$$Amf_7 \stackrel{def}{=} (req_{1sc}, v).(rep_{1sc}, v).Amf_8$$
$$Amf_8 \stackrel{def}{=} (get_{amfp}, r_p).(respond_1, r_r).Amf_{12}$$
$$Amf_9 \stackrel{def}{=} (get_{amfp}, r_p).(call_{2smf}, v).Amf_{10}$$
$$Amf_{10} \stackrel{def}{=} (req_{2sc}, v).(rep_{2sc}, v).Amf_{11}$$
$$Amf_{11} \stackrel{def}{=} (get_{amfp}, r_p).(respond_2, r_r).Amf_{12}$$
$$Amf_{12} \stackrel{def}{=} (rep_{se}, v).Amf_1$$

Component *NRF* is modeled using actions that represent discovery requests and replies, which are performed in cooperation with AMF. These actions are (req_d, v) and (rep_d, v), for request and reply, respectively. In addition, a state (Nrf_2) defines the access to the processor and the local operation that performs the discovery service. Nrf_2 is modeled using the sequential actions (get_{nrfp}, r_p) and $(discovery, r_d)$, for access to the processor and the discovery service, respectively. Listing 4 presents the PEPA-based model for NRF.

Listing 4. NRF.

$$Nrf_1 \stackrel{def}{=} (req_d, v).Nrf_2$$
$$Nrf_2 \stackrel{def}{=} (get_{nrfp}, r_p).(discover, r_d).Nrf_3$$
$$Nrf_3 \stackrel{def}{=} (rep_d, v).Nrf_1$$

Component *SMF* models the service for creating a session management context that allows UE to use 5G applications. SMF is modeled similarly to NRF. Four additional actions are needed, two actions are now necessary to model the request and two ones for the response. This addition is due to the fork in AMF that handles regular and emergency service. The state (Smf_2) defines the access to the processor and the local operation that performs the session creation service. Smf_2 is modeled using the sequential actions (get_{smfp}, r_p) and $(createSession, r_{cs})$, for access to the processor and the session creation service, respectively. Listing 5 presents the PEPA-based model for SMF.

Listing 5. SMF.

$$Smf_1 \stackrel{def}{=} (req_{1sc}, v).Smf_2 + (req_{2sc}, v).Smf_2$$
$$Smf_2 \stackrel{def}{=} (get_{smfp}, r_p).(createSession, r_{cs}).Smf_3$$
$$Smf_3 \stackrel{def}{=} (rep_{1sc}, v).Smf_1 + (rep_{2sc}, v).Smf_1$$

Components *UEP*, *AMFP*, *NRFP*, and ***SMFP*** model the processing entities on which UE, AMF, NRF, and SMF execute, respectively. These processing entities are modeled following the pattern of processing given in the Subsection III-C, in which each component hast two states. The first one gets access to the processor. The second state performs the actions deployed on the processor. As the model for *NRFP* was already presented in Listing 1, Lists 6, 7, and 8 present the PEPA-based models for *UEP*, *AMFP*, and *SMFP*, respectively.

Listing 6. UEP.

$$Uep_1 \stackrel{def}{=} (get_{uep}, r_p).Uep_2$$
$$Uep_2 \stackrel{def}{=} (think, r_{iat}).Uep_1$$

Listing 7. AMFP.

$$Amfp_1 \stackrel{def}{=} (get_{amfp}, r_p).Amfp_2$$
$$Amfp_2 \stackrel{def}{=} (prepare, r_{pr}).Amfp_1 + (call_{nrf}, v).Amfp_1 + (call_{1smf}, v).Amfp_1 + (respond_1, r_r).Amf_1 + (call_{2smf}, v).Amf_1 + (respond_2, r_r).Amf_1$$

Listing 8. SMFP.

$$Smfp_1 \stackrel{def}{=} (get_{smfp}, r_p).Smfp_2$$
$$Smfp_2 \stackrel{def}{=} (createSession, r_{cs}).Smfp_1$$

Overall system model. We use the operator of cooperation (Subsection III-B) to model the session establishment 5GNSL. List 9 presents the system model by using our new 5G-oriented PEPA constructions.

Listing 9. Overall system model.

$$((((Ue_1[N_{ue}] \underset{L_1}{\bowtie} Amf_1[N_{amf} \cdot N_{amfp} \cdot N_t]) \underset{L_2}{\bowtie} Nrf_1[N_{nrf} \cdot N_{nrfp} \cdot N_t]) \underset{L_3}{\bowtie} Smf_1[N_{smf} \cdot N_{smfp} \cdot N_t]) \underset{L_4}{\bowtie} (((Uep_1[N_{uep}] \underset{\emptyset}{\bowtie} Amfp_1[N_{amf} \cdot N_{amfp}]) \underset{\emptyset}{\bowtie} Nrfp_1[N_{nrf} \cdot N_{nrfp}]) \underset{\emptyset}{\bowtie} Smfp_1[N_{smf} \cdot N_{smfp}]))$$

$$L_1 = \{req_{se}, rep_{se}\}$$
$$L_2 = \{req_d, rep_d\}$$
$$L_3 = \{req_{1sc}, rep_{1sc}, req_{2sc}, rep_{2sc}\}$$
$$L_4 = \{get_{uep}, think, get_{amfp}, prepare, call_{nrf}, call_{1smf}, call_{2smf}, respond_1, respond_2, get_{nrfp}, discover, get_{smfp}, createSession\}$$
$$\emptyset = \{\}$$

Where N_{ue} is the number of UE. N_{amf}, N_{nrf}, and N_{smf} are the number of available instances of AMF, SMF, and NRF, respectively. N_{amfp}, N_{smfp}, and N_{nrfp} are the number of processors that are allocated to each instance of AMF, SMF, and NRF, respectively. It is noteworthy that each processor can handle a set of concurrent threads, which is noted by N_t. Thus, the product $N_{nf} \cdot N_{nfp} \cdot N_t$ represents the total number of threads of an NF. Moreover, the product $N_{nf} \cdot N_{nfp}$ is the total number of processors allocated to an NF. Also note that if a UE has a single processor, N_{ue} and N_{uep} are equal.

B. PERFORMANCE METRICS

To measure the performance and scalability of 5GNSLs, we use the metrics processor utilization, throughput, average response time, and scalability.

Utilization. This metric measures the total utilization of the processor when it is performing actions at the steady-state [20]. For example, the utilization of NRFP (Listing 1) is its population level when NRFP performs the action $(discover, r_d)$.

Throughput. This metric measures the frequency at which an action is performed at the steady-state [20]. For instance, the throughput for the session establishment 5GNSL is measured in the action (rep_{se}, v) of AMF (Listing 3) which represents the session establishment responses per time unit.

Average response time. This metric measures the average time that a component spends passing throughout a particular set of states [20]. For example, in the session establishment 5GNSL, the average response time corresponds to the time that AMF spends to discover SMF, create the session context, and prepare the response to UE.

Scalability Metric. This metric measures the scalability (Equation 1) as the ratio between the productivity of a system at two different scale factors k_2 and k_1 [24]. If $\psi(k_1, k_2) < 1$, the productivity of the system at scale k_2 is less than at scale k_1. When $\psi(k_1, k_2) = 1$, the productivity of system at scales k_1 and k_2 is equal. If $\psi(k_1, k_2) > 1$, the productivity of system at scale k_2 is higher than at scale k_1.

$$\psi(k_1, k_2) = \frac{F(k_2)}{F(k_1)} \tag{1}$$

where, $F(k)$ is the productivity of a system at the scale k, given by Equation 2.

$$F(k) = \frac{\lambda(k) \cdot f(k)}{C(k)} \tag{2}$$

where $\lambda(k)$ is the average throughput achieved at scale k, $C(k)$ is the running cost per second of the system at scale k, and $f(k)$ (Equation 3) is a value function determined by evaluating the performance of the scaled system. We consider the value function given in [24] to compare the average response time at scale k, $\overline{T}(k)$, with a target value $\widehat{T}$:

$$f(k) = \frac{1}{1 + \overline{T}(k)/\widehat{T}} \tag{3}$$

It is noteworthy that $f(k)$ rewards the productivity. This reward tends to one if $\overline{T}(k)$ is relatively much smaller than $\widehat{T}$. It is near to 0.5 if $\overline{T}(k)$ is close to $\widehat{T}$. It tends to zero if $\overline{T}(k)$ is relatively much higher than $\widehat{T}$.

C. RESULTS AND ANALYSIS

Table 3 presents the service rates used in the experimental evaluation. In particular, for $1/r_{iat}$, we used 600 seconds as IAT per subscriber, which is a typical value found in [25] from an analysis of the usage pattern of mobile data traffic. We set the other service rates by performing measurements in our 5G core network prototype available in [26]. In the experiments, we set $N_t = 36$ as the number of threads per processor, which is in the range supported by modern multi-task processors, and varied the number of concurrent users from 1 to 100, 000 to analyze configurations extensively.

Fig. 4 presents the throughput results for the basic configuration of the session establishment 5GNSL, $(N_{nrf}, N_{smf}, N_{amf}) = (1, 1, 1)$. These results show how the throughput saturation point depends on the number of processors that the NF instance uses. In particular, configuration (1,1,1)

TABLE 3. Service rates used for numerical experimentation.

Service Rate	Value	Description
r_{iat}	$1/600$	Rate between subsequent requests
r_{pr}	$1/0.001$	Rate of initial preparing by AMF
r_r	$1/0.02$	Rate of reply by AMF
r_d	$1/0.03$	Rate of discovery by NRF
r_{sc}	$1/0.025$	Rate of session creation by SMF
r_p	$1/0.00001$	Rate of processing on VMs
v	$1/0.00001$	Rate of almost real-time communication

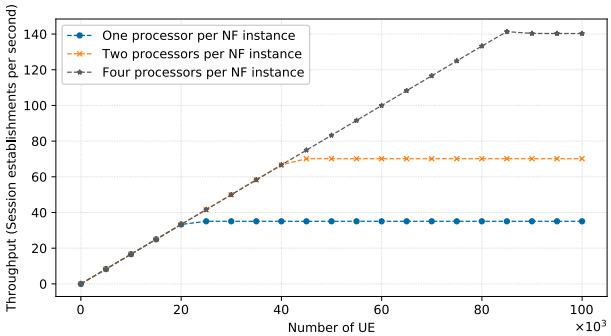


FIGURE 4. Throughput of session establishment 5GNSL for the basic configuration $(N_{nrf}, N_{smf}, N_{amf}) = (1, 1, 1)$.

can handle up to 35, 70, and 140 session establishment requests per second with one, two, and four processors per NF instance, respectively. Once the basic configuration achieves these saturation points, the session establishment 5GNSL starts to drop incoming requests. Summarizing, the results mentioned above confirm the correlation between throughput and number of processors; few processors lead to support low throughput and, in turn, many processors support high throughput.

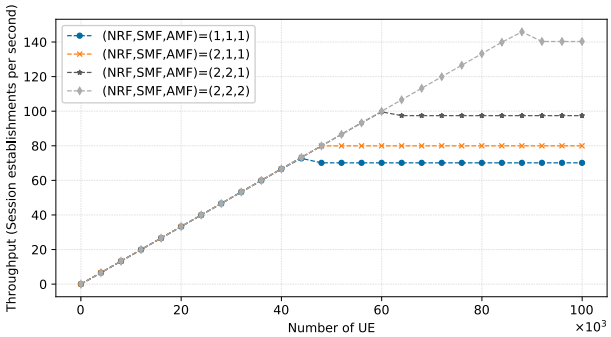


FIGURE 5. Throughput of session establishment 5GNSL for different configurations $(N_{nrf}, N_{smf}, N_{amf})$.

Fig. 5 presents the throughput for configurations (1,1,1), (2,1,1), (2,2,1), and (2,2,2). In this evaluation, we use two processors in each VM that hosts a VNF. As expected, in every configuration, the throughput increases with the number of users until it reaches its maximum. In this set of configurations, the increase in the number of NF instances produces an improvement in the throughput. Configurations (2,1,1), (2,2,1), and (2,2,2) achieved an improvement in the maximum throughput around 14%, 39%, and 101%,

respectively, regarding the maximum throughput offered by configuration (1,1,1). To sum up, the throughput supported increases with the number of NF instances.

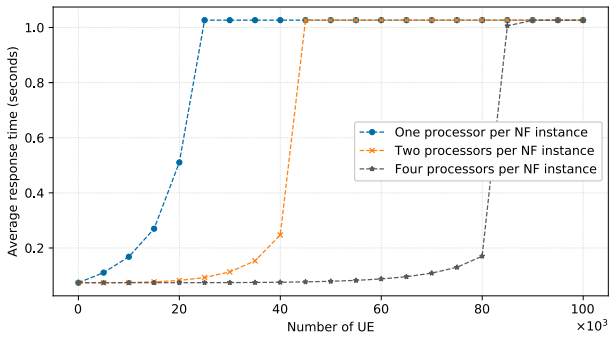


FIGURE 6. Average response time of session establishment 5GNSL for configuration ($N_{nrf}, N_{smf}, N_{amf}$) = (1, 1, 1), which is measured when AMF goes through states Amf_2 to Amf_{12} (see AMF component).

Fig. 6 presents the average response time results for the basic configuration of the session establishment 5GNSL. These results show that this time increases with the number of concurrent users up to a saturation point. This saturation point depends on the number of processors; in particular, configuration (1,1,1) can handle up to 23,000, 42,000, and 85,000 concurrent users with one, two, and four processors before the session establishment 5GNSL starts to drop incoming requests. If a Slice Architect sets, for instance, the target average response time to 200 *ms*, the session establishment 5GNSL can handle nearly 80,000 concurrent users using four processors per NF instance. Summarizing, the results mentioned above confirm the correlation between the average response time and the number of processors; many processors lead to support a high number of concurrent users in a short time.

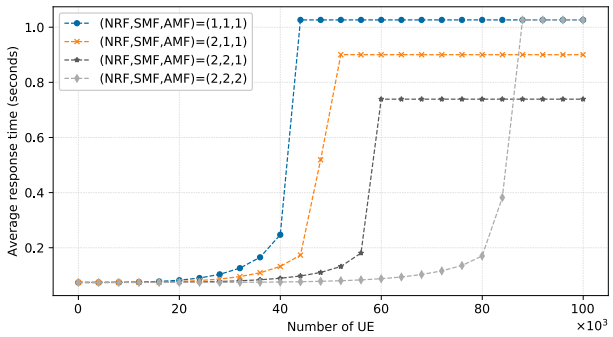


FIGURE 7. Average response time of session establishment 5GNSL for configurations (1,1,1), (2,1,1), (2,2,1), and (2,2,2).

Fig. 7 presents the average response time results for configurations (1,1,1), (2,1,1), (2,2,1), and (2,2,2) when VMs that host the VNF instances use two processors. According to these results, the average response time improves as NFs scale-out. Specifically, this time is lower in configurations (2,1,1) and (2,2,1) than in basic configuration around 13% and 28%, respectively. However, the maximum average response time in configuration (2,2,2) is as high as in configuration (1,1,1) due to the entry point of the slice is AMF. When AMF is scaled-out, it sends a high number of requests to the other NFs. Thus, SMF and NRF must also be scaled-out to avoid bottlenecks.

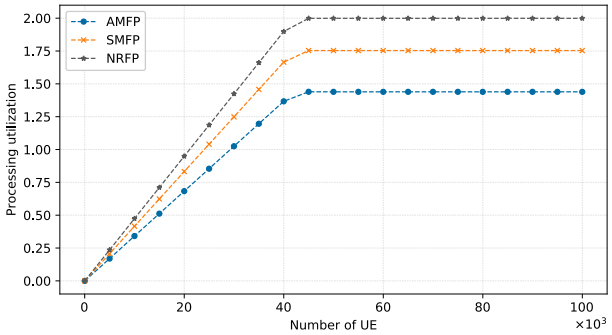


FIGURE 8. Processor utilization of session establishment 5GNSL for configuration ($N_{nrf}, N_{smf}, N_{amf}$) = (1, 1, 1) and two processors allocated per NF instance. Utilization is measured as the population of the second state of processing (*e.g.*, $Amfp_2$).

Fig. 8 shows the processor utilization results of AMFP, SMFP, and NRFP with two processors allocated per NF instance for the basic configuration of the session establishment 5GNSL. NRFP reaches its maximum utilization (1.97) for 45,000 concurrent users, explaining the overload of the session establishment 5GNSL from this point (see Figures 4 and 5). This slice is overloaded from 45,000 users, although AMFP and SMFP are underused, 1.45, and 1.75, respectively. In summary, these results show that if in a sequential chain, an NF becomes a bottleneck, the request processing chain fails thoroughly.

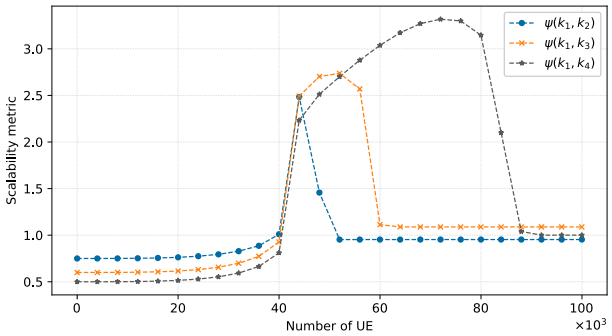


FIGURE 9. Scalability metric of session establishment 5GNSL for configurations (1,1,1), (2,1,1), (2,2,1), and (2,2,2).

Fig. 9 presents the scalability results when configurations $k_1 = (1, 1, 1)$, $k_2 = (2, 1, 1)$, $k_3 = (2, 2, 1)$, $k_4 = (2, 2, 2)$ use two processors in each VM that hosts a VNF. The scalability results reveal some facts. First, the basic configuration is useful to attend less than 40,000 users because $\psi(k_1, k_2)$, $\psi(k_1, k_3)$, and $\psi(k_1, k_4)$ are less than 1; similar productivity with less cost which is related to the number of NF instances that constitute a configuration. Second, the configuration (2,1,1) is appropriate to attend between 40, 000 and 45, 000

concurrent users because $\psi(k_1, k_2) > 1$; this configuration costs less than upper configurations. Third, the configuration (2,2,1) meets between 45, 000 and 54, 000 concurrent users because $\psi(k_1, k_3) > 1$; this configuration costs less than (2,2,2) configuration. Fourth, the configuration (2,2,2) is useful to attend more than 54,000 concurrent users.

Summarizing, the main insights from the results above described are; first, our method allows NSPs to dimension 5GNSLs. Second, a single NF in saturation state generates a failure in a sequential service chain; thus, for NSLs containing this kind of chain, it is necessary to scale the number of NF instances coordinately. Second, horizontal scaling allows improving the performance of the slice regarding the maximum number of concurrent users, throughput, and average response time. However, a more significant number of instances implies an increasing cost. Therefore, there must be a balance between performance (QoS to be supported) and the number of NF instances.

D. VALIDATION WITH LAYERED QUEUING NETWORK

We validate the accuracy of our PEPA-based model by LQN [27] since LQN is a well established and accepted model for performance evaluation of distributed systems, such as the core network in 5GNSLs. Accuracy is the difference between the measured obtained by our model and the got by LQN (see Equation 4 that follows the definition of percentage relative error proposed in [14]). LQN is a model for Extended Queuing Networks, which allows analyzing client-server architectures with nested multiple resource possession in which successive depths of nesting define the layers. Information for constructing LQN models is available in [28]. PEPA can be mapped to LQN by the process-algebraic proposed in [14]. We followed this process to model in LQN the session establishment 5GNSL.

$$Error(\%) = \left| \frac{PEPA_{metric} - LQN_{metric}}{LQN_{metric}} \right| \times 100 \qquad (4)$$

Fig. 10 presents the LQN model for the session establishment 5GNSL, including tasks, processors, entries, calls, and demands. Tasks (large parallelograms) are interacting entities (*e.g.,* software services) that carry out operations defined by their entries (services). We map PEPA components in NFs as Tasks. A task has a host processor (ovals) that models the computational resource used to carry out service operations; we map PEPA processing entities as LQN processors. Each processor has a queue, a discipline for executing its tasks (*e.g.,* First-In-First-Out), and a multiplicity (noted as $< N_{nf} >$); this multiplicity represents the number of NF instances in the horizontal scaling. We map UE as a Reference task that does not receive any request. In this validation, UE is a load generator that cyclically creates requests for the AMF task.

A task has one or more entries (smaller parallelograms), representing different operations it may perform. We map operations served by NFs: session establishment, session management context creation, and discovery as entries. The

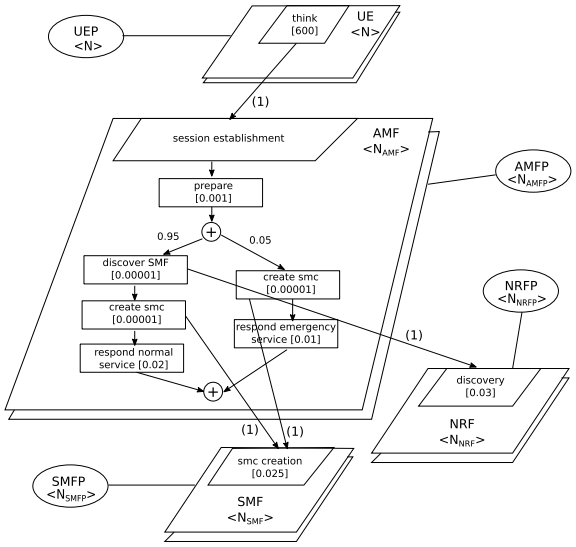


FIGURE 10. LQN model of the session establishment 5GNSL.

session establishment entry performs six activities (rectangles) that model the fork between regular and emergency service requests and operate in sequence, as indicated by the arrows. The activities *discover SMF* and *create smc* call the entries in NRF and SMF, respectively. A call may be synchronous, asynchronous, or forwarding. We map synchronous communication in PEPA as synchronous calls between tasks UE, AMF, SMF, and NRF. LQN demands by the total average amounts of host processing and the average number of calls for service operations required to complete an entry. We map service rates with time demands.

For solving the LQN model of the session establishment 5GNSL, we used the analytical solver LQNS proposed by [29]. In LQN, the response time of an entry is the time spent answering a single request; it includes the running time and the blocking time (waiting for a processor or waiting for nested lower services to complete). The utilization of a single-threaded task is the fraction of time that such task is busy (executing or blocked), meaning not idle. A multi-threaded task may have several services underway at once, and its utilization is the mean number of busy threads. The throughput is the number of service requests of an entry served by a task in a time unit. Summarizing, average response time, throughput, and processor utilization match with the performance metrics defined for the PEPA modeling (see Subsection IV-B).

Table 4 presents the validation results averaged over five independent runs for configurations $(N_{nrf}, N_{smf}, N_{amf}) =$ (1, 1, 1), (2, 1, 1), (2, 2, 1), and (2,2,2) for 20,000 UEs and 80,000 UEs. The evaluation results corroborate the accuracy of our method. It measures performance metrics with negligible difference in comparison to LQN. The most significant differences occur in the average response time measures, mainly when the number of users is close to the overload point, as in 80,000 UE in configuration (2,2,2).

TABLE 4. Model validation (* 20,000 UE, ** 80,000 UE, relative error in %).

2*Config.	Throughput			Average response time			AMFP Utilization			SMFP Utilization			NRFP Utilization		
	PEPA	LQN	Error	PEPA	LQN	Error	PEPA	LQN	Error	PEPA	LQN	Error	PEPA	LQN	Error
$(1,1,1)$*	33.3	33.2	0.30	0.0746	0.0827	9.79	0.7004	0.6998	0.09	0.8331	0.8323	0.10	0.9498	0.9489	0.09
$(1,1,1)$**	70.2	70.1	0.14	1.0263	1.0265	0,02	1.4745	1.4742	0.02	1.7538	1.7534	0.02	1.9993	1.9989	0.02
$(2,1,1)$*	33.3	33.2	0.30	0.0746	0.0782	4.60	0.7004	0.6998	0.10	0.8332	0.8324	0.10	0.9498	0.9489	0.10
$(2,1,1)$**	79.9	80.0	0.13	0.9003	0.9002	0.01	1.6808	1.6810	0.01	1.9992	1.9994	0.01	2.2790	2.2793	0.01
$(2,2,1)$*	33.3	33.2	0.30	0.0746	0.0757	1.45	0.7004	0.6998	0.09	0.8331	0.8324	0.08	0.9498	0.9489	0.09
$(2,2,1)$**	94.9	95.1	0.21	0.7581	0.7570	0.15	1.9962	1.9991	0.15	2.3742	2.3777	0.15	2.7066	2.7106	0.15
$(2,2,2)$*	33.3	33.2	0.30	0.0746	0.0745	0.13	0.7004	0.6998	0.10	0.8332	0.8324	0.10	0.9498	0.9489	0.10
$(2,2,2)$**	133.3	133.2	0.08	0.0746	0.1523	51.0	2.8021	2.8014	0.02	3.3328	3.3318	0.03	3.7995	3.7983	0.03

These differences are because the fluid approximation used in our PEPA-based model presents a more pronounced transition to overload than the LQN Solver. Readers can found an exhaustive comparison between PEPA and LQN in [14], [19].

V. CASE STUDY: USER REGISTRATION V2X 5GNSL

This case study aims two-folds. First, presenting the use of our method by modeling the user registration V2X 5GNSL. Second, analyzing an auto-scaler functionality to dimension the capacity of this 5GNSL. We also used the PEPA Eclipse plugin running on an Intel Core i5 PC (1.7 GHz) with 4 GB of RAM.

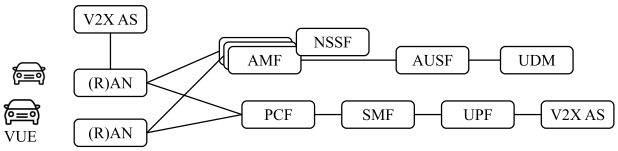


FIGURE 11. Service layer of the V2X 5GNSL.

A. MODELING

Let us consider the V2X 5GNSL designed in [9]. V2X aims at supporting advanced driving assistance services, such as cooperative driving based on sensors data and driving intentions [30]. Also, let us analyze the registration procedure that is supported by the service layer depicted in Fig. 11. For registration to the network slice, the Vehicular User Equipment (VUE) provides the network slice selection information to AMF. Then, AMF cooperates with the Network Slice Selection Function (NSSF) to perform the slice selection. Once NSSF has validated the subscriber information, VUE is authenticated cooperatively by AUSF and the Unified Data Management (UDM). After successful authentication, VUE can use network slice services. It is noteworthy that a V2X network slice needs multiple NF instances to avoid overloading, which can increase service latency. In this case study, we reuse the specification of the PEPA-based model of NRF (Listing 4) for the discovery of AUSF. NRF allows associating the most appropriate authentication method at run time, such as EAP-AKA or EAP-AKA', which is useful for 3GPP and non-3GPP accesses.

As in the previous case study, the modeling includes: *1)* the service rates of the operations performed by NFs (*i.e.,* VUE, AMF, NSSF, NRF, AUSF, and UDM) and the processors of VMs, *2)* the PEPA components of NFs and processors; and *3)* the system model of the user registration V2X 5GNSL. In particular, we model the registration procedure without the Policy Charging Function (PCF). We consider that PCF affects the registration performance negligibly due to charging rules are relatively straightforward compared to other core functions. Thus, we assume that the operation of PCF does not significantly impact the performance of the slice registration process.

Service rates. Table 5 presents the service rates (r) in VUE, AMF, NSSF, NRF, AUSF, and UDM. It also introduces the service rates in processors of VMs that host such NFs.

TABLE 5. Service rates for modeling PEPA actions in the V2X 5GNSL.

Service rate	Description
r_{iat}	$1/r_{iat}$ (inter-arrival time) is the time that VUE interposes between successive requests of registration.
r_r	$1/r_r$ is the average time demand for preparing a response message by AMF.
r_s	$1/r_s$ is the average time that takes the selection of a slice.
r_d	$1/r_d$ is the average time that takes the discovery service performed by NRF.
r_a	$1/r_a$ is the average time that takes authentication in AUSF.
r_{rd}	$1/r_r$ is the average time that takes the recovery of vue data.
r_p	$1/r_p$ is the average execution demand on VM's processors.
v	v is a high service rate that models almost instantaneous service calls and replies between NFs; *e.g.,* AMF calling the AUSF's authentication service.

PEPA components. In the next paragraphs, we describe the model of VUE, AMF, NSSF, AUSF, and UDM as PEPA components and the model of the processors of VMs. In this case study, we do not present the NRF (Listing 4) and NFRP (Listing 1) models because they are similar to those described in the previous case study.

Component *VUE* models VUE requesting the registration to the V2X 5GNSL (see Listing 10). Between successive requests, VUE interposes an IAT defined by r_{iat} representing the duration that VUE remains registered to a previous network slice. VUE has two states. The first one (Vue_1) gets access to the processor using the action (get_{vuep}, r_p), and then, performs the action ($stayRegistered, r_{iat}$). The second state (Vue_2) models the synchronous communication between VUE and AMF using the actions of request (req_{reg}, v) and response (rep_{reg}, v).

Listing 10. VUE.

$$Vue_1 \stackrel{def}{=} (get_{vuep}, r_p).(stayRegistered, r_{iat}).Vue_2$$
$$Vue_2 \stackrel{def}{=} (req_{reg}, v).(rep_{reg}, v).Vue_1$$

Component *AMF* processes registration requests from VUE. The action sequence is defined for a set of states Amf_i, $i = \{1, \ldots, 9\}$ as follows. Amf_1 models the service request. Amf_2 starts a call to NSSF for slice selection. Amf_3 models the synchronous communication between AMF and NSSF. Amf_4 starts a call to NRF for discovering the appropriate AUSF. Amf_5 represents the synchronous communication between AMF and NRF. Amf_6 starts a call to AUSF for authentication of VUE. Amf_7 models the synchronous communication between AMF and AUSF. Amf_8 prepares a message response to send back to VUE once the registration procedure finishes. Amf_9 models the registration reply to VUE. Listing 11 presents the PEPA-based model for AMF.

Listing 11. AMF.

$$Amf_1 \stackrel{def}{=} (req_{reg}, v).Amf_2$$
$$Amf_2 \stackrel{def}{=} (get_{amfp}, r_p).(call_{nssf}, v).Amf_3$$
$$Amf_3 \stackrel{def}{=} (req_{ss}, v).(rep_{ss}, v).Amf_4$$
$$Amf_4 \stackrel{def}{=} (get_{amfp}, r_p).(call_{nrf}, v).Amf_5$$
$$Amf_5 \stackrel{def}{=} (req_d, v).(rep_d, v).Amf_6$$
$$Amf_6 \stackrel{def}{=} (get_{amfp}, r_p).(call_{ausf}, v).Amf_7$$
$$Amf_7 \stackrel{def}{=} (req_{auth}, v).(rep_{auth}, v).Amf_8$$
$$Amf_8 \stackrel{def}{=} (get_{amfp}, r_p).(respond, r_r).Amf_9$$
$$Amf_9 \stackrel{def}{=} (rep_{reg}, v).Amf_1$$

Component *NSSF* models the network slice selection service offered by NSSF using three states. The state $Nssf_1$ defines the request from AMF. The state $Nssf_2$ represents the access to the processor and the local operation that performs the network slice selection. The state $Nssf_3$ models the response to AMF. Listing 12 presents the PEPA-based model for NSSF.

Listing 12. NSSF.

$$Nssf_1 \stackrel{def}{=} (req_{ss}, v).Nssf_2$$
$$Nssf_2 \stackrel{def}{=} (get_{nssfp}, r_p).(selectSlice, r_s).Nssf_3$$
$$Nssf_3 \stackrel{def}{=} (rep_{ss}, v).Nssf_1$$

Component *AUSF* handles authentication requests. The action sequence is defined for a set of states $Ausf_i$, $i = \{1, \ldots, 5\}$. $Ausf_1$ models the service request. $Ausf_2$ starts a call to UDM for retrieving administrative data of VUE. $Ausf_3$ models the synchronous communication between AUSF and UDM for data retrieving. $Ausf_4$ represents the local operation that authenticates VUE. $Ausf_5$ is the service reply. Listing 13 presents the PEPA-based model for AUSF.

Listing 13. AUSF.

$$Ausf_1 \stackrel{def}{=} (req_{auth}, v).Ausf_2$$
$$Ausf_2 \stackrel{def}{=} (get_{ausfp}, r_p).(call_{udm}, v).Ausf_3$$
$$Ausf_3 \stackrel{def}{=} (req_{rd}, v).(rep_{rd}, v).Ausf_4$$
$$Ausf_4 \stackrel{def}{=} (get_{ausfp}, r_p).(authenticateVue, r_a).Ausf_5$$
$$Ausf_5 \stackrel{def}{=} (rep_{auth}, v).Ausf_1$$

Component *UDM* models the data recovery service. UDM is modeled using actions that represents data recovery requests and replies, which are performed in cooperation with AUSF. These actions are (req_{rd}, v) and (rep_{rd}, v), for request and reply, respectively. In addition, a state (Udm_2) defines the access to the processor and the local operation that performs the data recovery service. Udm_2 is modeled using the sequential actions (get_{udmp}, r_p) and $(recoverData, r_{rd})$. Listing 14 presents the PEPA-based model for UDM.

Listing 14. UDM.

$$Udm_1 \stackrel{def}{=} (req_{rd}, v).Udm_2$$
$$Udm_2 \stackrel{def}{=} (get_{udmp}, r_p).(recoverData, r_{rd}).Udm_3$$
$$Udm_3 \stackrel{def}{=} (rep_{rd}, v).Udm_1$$

Components *VUEP*, ***AMFP***, ***NSSFP***, ***NRFP***, ***AUSFP***, and ***UDMP*** model the processing entities on which VUE, AMF, NSSF, NRF, AUSF, and UDM execute. We use the pattern of processing (see Subsection III-C) to model these entities, in which each component hast two states. The first state gets access to the processor, whereas the second state performs the actions deployed on the processor. Listings 15, 16, 17, 18, and 19 present the PEPA-based models for VUEP, AMFP, NSSFP, NRFP, AUSFP, and UDMP, respectively.

Listing 15. VUEP.

$$Vuep_1 \stackrel{def}{=} (get_{vuep}, r_p).Vuep_2$$
$$Vuep_2 \stackrel{def}{=} (stayRegistered, r_{iat}).Vuep_1$$

Listing 16. AMFP.

$$Amfp_1 \stackrel{def}{=} (get_{amfp}, r_p).Amfp_2$$
$$Amfp_2 \stackrel{def}{=} (call_{nssf}, v).Amfp_1 + (call_{nrf}, v).Amfp_1 + (call_{ausf}, v).Amfp_1 + (respond, r_r).Amfp_1$$

Listing 17. NSSFP.

$$Nssfp_1 \stackrel{def}{=} (get_{nssfp}, r_p).Nssfp_2$$
$$Nssfp_2 \stackrel{def}{=} (selectSlice, r_s).Nssfp_1$$

Listing 18. AUSFP.

$$Ausfp_1 \stackrel{def}{=} (get_{ausfp}, r_p).Ausfp_2$$
$$Ausfp_2 \stackrel{def}{=} (call_{udm}, v).Ausfp_1 + (authenticateVue, r_a).Ausfp_1$$

Listing 19. UDMP.

$$Udmp_1 \stackrel{def}{=} (get_{udmp}, r_p).Udmp_2$$
$$Udmp_2 \stackrel{def}{=} (recoverData, r_{rd}).Udmp_1$$

Overall system model. We model the entire system of the user registration V2X 5GNSL (see List 20) by using the operator of cooperation (Subsection III-B).

In List 20, N_{vue} is the number of VUE, and N_{amf}, N_{nssf}, N_{nrf}, N_{ausf}, and N_{udm} are the number of available instances of AMF, NSSF, NRF, AUSF, and UDM, respectively. Also, N_{amfp}, N_{nssfp}, N_{nrfp}, N_{ausfp}, and N_{udmp} are the number of processors that are allocated to each instance of AMF, NSSF, NRF, AUSF, and UDM, respectively. It is noteworthy that each processor can handle a set of concurrent threads, which is noted by N_t. Thus, the product $N_{nf} \cdot N_{nfp} \cdot N_t$ represents the

total number of threads of an NF. The product $N_{nf} \cdot N_{nfp}$ is the total number of processors allocated to an NF. Also, if a VUE has a single processor, N_{vue} and N_{vuep} are equal.

Listing 20. **Overall system model.**

$$
\begin{aligned}
&((((((Vue_1[N_{vue}] \underset{L_1}{\bowtie} Amf_1[N_{amf} \cdot N_{amfp} \cdot N_t]) \underset{L_2}{\bowtie} \\
&Nssf_1[N_{nssf} \cdot N_{nssfp} \cdot N_t]) \underset{L_3}{\bowtie} Nrf_1[N_{nrf} \cdot N_{nrfp} \cdot N_t]) \\
&\underset{L_4}{\bowtie} Ausf_1[N_{ausf} \cdot N_{ausfp} \cdot N_t]) \underset{L_5}{\bowtie} \\
&Udm_1[N_{udm} \cdot N_{udmp} \cdot N_t]) \underset{L_6}{\bowtie} \\
&(((((Vuep_1[N_{vuep}] \underset{\emptyset}{\bowtie} Amfp_1[N_{amf} \cdot N_{amfp}]) \underset{\emptyset}{\bowtie} \\
&Nssfp_1[N_{nssf} \cdot N_{nssfp}]) \underset{\emptyset}{\bowtie} Nrfp_1[N_{nrf} \cdot N_{nrfp}]) \underset{\emptyset}{\bowtie} \\
&Ausfp_1[N_{ausf} \cdot N_{ausfp}]) \underset{\emptyset}{\bowtie} Udmp_1[N_{udm} \cdot N_{udmp}]))
\end{aligned}
$$

$$
\begin{aligned}
L_1 &= \{req_{reg}, rep_{reg}\} \\
L_2 &= \{req_{ss}, rep_{ss}\} \\
L_3 &= \{req_d, rep_d\} \\
L_4 &= \{req_{auth}, rep_{auth}\} \\
L_5 &= \{req_{rd}, rep_{rd}\} \\
L_6 &= \{get_{vuep}, stayRegistered, get_{amfp}, call_{nssf}, call_{nrf}, \\
&\quad call_{ausf}, respond, get_{nssfp}, selectSlice, get_{nrfp}, \\
&\quad discover, get_{ausfp}, call_{udm}, authenticateVue, \\
&\quad get_{udmp}, recoverData\} \\
\emptyset &= \{\}
\end{aligned}
$$

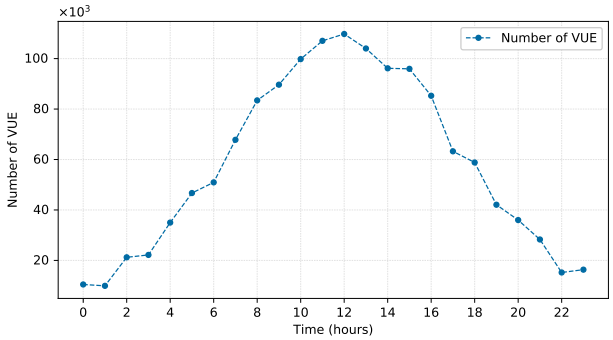


FIGURE 12. **Variable workload (number of VUE) in a period of 24 hours.**

B. RESULTS AND ANALYSIS

In this experiment, we consider the automatic scaling of the V2X 5GNSL to face workload variations of the Fig. 12, assuming the performance policy: "Each NF (*i.e.,* AMF, NSSF, NRF, AUSF, and UDM) must operate without overloading between 40 and 70 percent of its processor utilization". The following scaling policy may be defined to meet the performance policy afore-mentioned: "The scaling system increases by one instance when the utilization is above the upper threshold (70%). On the other hand, it decreases in one instance when the utilization is less than the lower threshold (40%)". We assumed that each NF instance uses two processors. Table 6 presents the service rates used in this evaluation.

Fig. 13 shows that applying the scaling policy allows keeping the processor utilization in the user registration V2X 5GNSL into the target range most of the time. At the start and end of the workload pattern, when the number

TABLE 6. **Service rates used for numerical experimentation.**

Service rate	Value	Description
r_{iat}	$1/600$	Rate between subsequent requests.
r_r	$1/0.02$	Rate of reply by AMF.
r_s	$1/0.025$	Rate of slice selection by NSSF.
r_d	$1/0.015$	Rate of discovery by NRF.
r_a	$1/0.03$	Rate of authentication by AUSF.
r_{rd}	$1/0.001$	Rate of data recovery by UDM.
r_p	$1/0.00001$	Rate of processing on VMs.
v	$1/0.00001$	Rate of almost instantaneous communication.

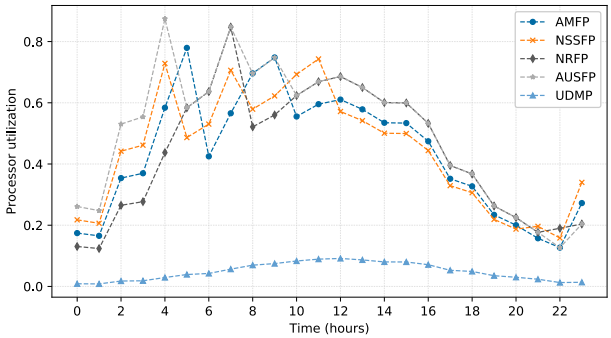


FIGURE 13. **Processor utilization as the scaling policy is applied.**

of users is relatively low, the utilization is less than 40%, which indicates that all NFs need just one instance (see Fig. 14). At some hours (*e.g.,* 4, 5, and 9), the utilization is higher than 70% because the scaling system is reactive. However, since 70% is a conservative value, the NFs of the user registration V2X 5GNSL are not overloaded. It is noteworthy that UDM uses only one instance for the entire experimentation time due to its high rate to process data recovery requests ($r_{rd} = 1/0.001$).

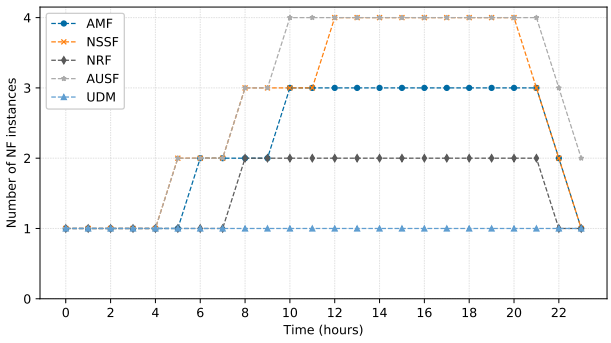


FIGURE 14. **Number of NF instances in horizontal scaling to accomplish the performance policy.**

Fig. 14 plots the number of NF instances required to comply with the performance policy in the user registration V2X 5GNSL. It is noteworthy that NFs with lower service rates, such as NSSF and AUSF, are scaled to a greater extent. In particular, NSSF and AUSF need up to four instances, NRF up to three instances, NSSF up to two instances, and UDM only one (it has a very high service rate).

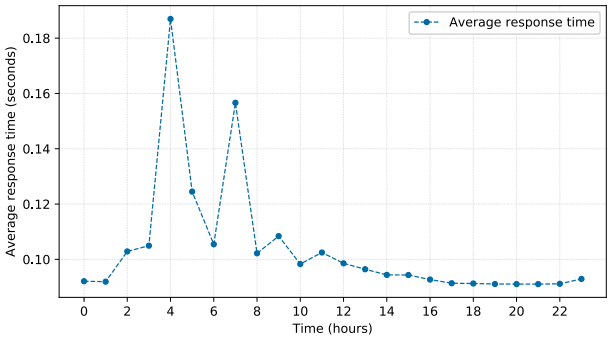


FIGURE 15. Average response time as the scaling policy is applied.

Fig. 15 presents the average response time of NFs used in the user registration V2X 5GNSL, revealing that this time is low when AMF, NSSF, NRF, AUSF, and UDM are not overloaded. The performance and scalability policies considered in this case study are simple, but they still allow for low response times; the maximum average response time is around 185*ms*. A performance policy that sets a more demanding objective in response time will need a more complex scaling mechanism than the here presented.

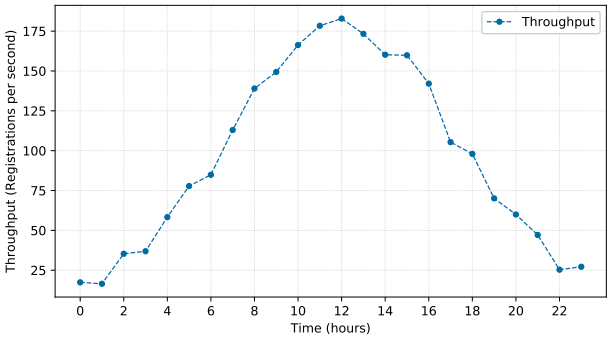


FIGURE 16. Throughput (registrations per second) as the scaling policy is applied.

Fig. 16 shows the throughput of the user registration V2X 5GNSL. As expected, the maximum throughput (number of VUE registrations per second) is supported when the slice operates with their highest number of instances. In particular, the slice achieved a maximum of 180 registration per second at 12 hours with the following configuration: 3 AMF, 4 NSSF, 4 AUSF, 2 NRF, and 1 UDM.

Summarizing, the main insights from the results above described are; first, our method allows network service providers to implement automatic scaling in 5GNSLs. Second, the service rates are significant to NFs since, to implement policies, they can determine their number of instances as in the UDM case.

VI. CONCLUSION AND FUTURE WORK

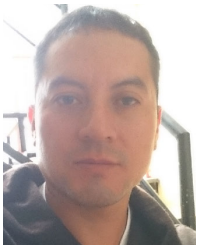
This paper presented a PEPA-based method to analyze the scalability and performance of core network services in 5GNSL. This method allows specifying 5G network functions as sequential process components and their interaction forming a cooperation process. We introduced new composite structures based on PEPA useful to model and evaluate 5G network core procedures. We illustrated the use of the proposed method by presenting the modeling and assessment regarding scalability and performance metrics of the session establishment 5GNSL and the user registration process V2X 5GNSL. The evaluation results in the two case studies corroborated the usefulness of our method to model 5G core network services and dimension the capacity of 5GNSLs. Furthermore, the accuracy validation results corroborated that our PEPA-based method measures performance metrics (throughput, average response time, and processor utilization) with negligible difference regarding LQN.

As future work, we intend to validate our method against measurements from a 5G core network prototype to verify the accuracy in the modeling. We will also consider more complex scenarios, such as registration and handover procedures in simultaneous execution.

REFERENCES

[1] *Description of Network Slicing Concept*, Final Deliverable, NGMN Alliance, Frankfurt, Germany, 2016.
[2] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017.
[3] H. Tullberg, P. Popovski, Z. Li, M. A. Uusitalo, A. Höglund, Ö. Bulakci, M. Fallgren, and J. F. Monserrat, "The METIS 5G system concept: Meeting the 5G requirements," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 132–139, Dec. 2016.
[4] H. Zhang, N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. M. Leung, "Network slicing based 5G and future mobile networks: Mobility, resource management, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 138–145, Aug. 2017.
[5] C. Campolo, A. Molinaro, A. Iera, and F. Menichella, "5G network slicing for vehicle-to-everything services," *IEEE Wireless Commun.*, vol. 24, no. 6, pp. 38–45, Dec. 2017.
[6] J. Prados-Garzon, J. J. Ramos-Munoz, P. Ameigeiras, P. Andres-Maldonado, and J. M. Lopez-Soler, "Modeling and dimensioning of a virtualized MME for 5G mobile networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 5, pp. 4383–4395, May 2017.
[7] *System Architecture for the 5G System*, document TR 23.501, V15.3.0, Release 15, 3GPP, Sophia-Antipolis, France, 2018.
[8] R. Trivisonno, M. Condoluci, X. An, and T. Mahmoodi, "MIoT slice for 5G systems: Design and performance evaluation," *Sensors*, vol. 18, no. 2, p. 635, Feb. 2018.
[9] C. Campolo, R. Fontes, A. Molinaro, C. E. Rothenberg, and A. Iera, "Slicing on the road: Enabling the automotive vertical through 5G network softwarization," *Sensors*, vol. 18, no. 12, p. 4435, Dec. 2018.
[10] S. Schneider, A. Sharma, H. Karl, and H. Wehrheim, "Specifying and analyzing virtual network services using queuing Petri nets," in *Proc. IFIP/IEEE IM*, Washington, DC, USA, Apr. 2019, pp. 116–124.
[11] J. Hillston, "PEPA—Performance enhanced process algebra," Dept. Comput. Sci., Univ. Edinburgh, Edinburgh, U.K., Tech. Rep. 1, Mar. 1993.
[12] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Barcelona, Spain, Nov. 2015, pp. 384–389.
[13] J. Hillston, M. Tribastone, and S. Gilmore, "Stochastic process algebras: From individuals to populations," *Comput. J.*, vol. 55, no. 7, pp. 866–881, Jul. 2012, doi: 10.1093/comjnl/bxr094.
[14] M. Tribastone, "Relating layered queueing networks and process algebra models," in *Proc. 1st Joint WOSP/SIPEW Int. Conf. Perform. Eng. (WOSP/SIPEW)*. New York, NY, USA: Association for Computing Machinery, 2010, pp. 183–194, doi: 10.1145/1712605.1712634.

[15] *Procedures for the 5G System*, document TR 23.502, V15.2.0, Release 15, 3GPP, Sophia-Antipolis, France, 2018.
[16] *5G System; Access and Mobility Management Services; Stage 3*, document TS 29.518, V15.3.0, Release 15, 3GPP, Sophia-Antipolis, France, 2019.
[17] H. S. Oluwatosin, "Client-server model," *IOSRJ Comput. Eng.*, vol. 16, no. 1, pp. 2278–8727, 2014.
[18] J. Hillston, "A compositional approach to performance modelling," Ph.D. dissertation, Univ. Edinburgh, Edinburgh, U.K., 1994.
[19] M. Tribastone, "A fluid model for layered queueing networks," *IEEE Trans. Softw. Eng.*, vol. 39, no. 6, pp. 744–756, Jun. 2013.
[20] C. D. Williams and A. Clark, "A case study in capacity planning for PEPA models with the PEPA eclipse plug-in," *Electron. Notes Theor. Comput. Sci.*, vol. 318, pp. 69–89, Nov. 2015.
[21] S. Donatelli, M. Ribaudo, and J. Hillston, "A comparison of performance evaluation process algebra and generalized stochastic Petri nets," in *Proc. 6th Int. Workshop Petri Nets Perform. Models*, Durham, NC, USA, 1995, pp. 158–168.
[22] M. Tribastone and S. Gilmore, "Scaling performance analysis using fluid-flow approximation," in *Rigorous Software Engineering for Service-Oriented Systems*, M. Wirsing and M. Hölzl, Eds. Berlin, Germany: Springer-Verlag, 2011, pp. 486–505.
[23] M. Tribastone, A. Duguid, and S. Gilmore, "The PEPA eclipse plugin," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 28–33, Mar. 2009.
[24] P. Jogalekar and M. Woodside, "Evaluating the scalability of distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 6, pp. 589–603, Jun. 2000.
[25] E. M. R. Oliveira, A. C. Viana, K. P. Naveen, and C. Sarraute, "Mobile data traffic modeling: Revealing temporal facets," *Comput. Netw.*, vol. 112, pp. 176–193, Jan. 2017.
[26] C. H. T. Arteaga. *A 5G Core Network Prototype*. Accessed: Nov. 12, 2019. [Online]. Available: https://github.com/carloshtobar/A5GCoreNetworkPrototype
[27] G. Franks, T. Al-Omari, M. Woodside, O. Das, and S. Derisavi, "Enhanced modeling and solution of layered queueing networks," *IEEE Trans. Softw. Eng.*, vol. 35, no. 2, pp. 148–161, Mar. 2009.
[28] M. Woodside, "Tutorial introduction to layered modeling of software performance—Edition 4.0," RADS Lab., Carleton Univ., Ottawa, ON, Canada, Tech. Rep. 1, 2013.
[29] G. Franks, P. Maly, M. Woodside, D. C. Petriu, A. Hubbard, and M. Mroz, "Layered queueing network solver and simulator user manual," Dept. Syst. Comput. Eng., Carleton Univ., Ottawa, ON, Canada, Tech. Rep. 11145, 2013.
[30] *Enhancement of 3GPP Support for V2X Scenarios*, document TR 22.186, V16.0.0, Release 15, 3GPP, Sophia-Antipolis, France, 2018.



CARLOS HERNAN TOBAR ARTEAGA (Member, IEEE) received the bachelor's degree in electronics and telecommunications engineering and the M.Sc. degree in electronics and telecommunications from the Universidad del Cauca, Popayán, Colombia, in 2004 and 2012, respectively, where he is currently pursuing the Ph.D. degree in telematics engineering. His research interests include network and service management, performance modeling, and machine learning.

ARMANDO ORDOÑEZ received the bachelor's degree in electronics and telecommunications engineering and the Ph.D. degree in telematics engineering from the Universidad del Cauca, Popayán, Colombia, in 2003 and 2014, respectively. He is currently a Postdoctoral Fellow with the Universidad del Cauca. His research interests include software-defined networking and machine learning.

OSCAR MAURICIO CAICEDO RENDON (Senior Member, IEEE) received the bachelor's degree in electronics and telecommunications engineering and the M.Sc. degree in telematics engineering from the Universidad del Cauca, Popayán, Colombia, in 2001 and 2006, respectively, and the Ph.D. degree in computer science from the Federal University of Rio Grande do Sul, Porto Alegre, Brazil, in 2015. He is currently a Full Professor with the Department of Telematics, Universidad del Cauca, where he is also a member of the Telematics Engineering Group. His research interests include network and service management, network virtualization, and software-defined networking.