This article has been accepted for publication in a future issue of IEEE Transactions on Vehicular Technology. Citation information: DOI 10.1109/TVT.2016.2608942, IEEE Transactions on Vehicular Technology. A preprint version of this manuscript is available at: http://ieeexplore.ieee.org/document/7565650/

Modeling and Dimensioning of a Virtualized MME for 5G Mobile Networks

Jonathan Prados-Garzon, Juan J. Ramos-Munoz, Pablo Ameigeiras, Pilar Andres-Maldonado,
Juan M. Lopez-Soler

***Abstract*—Network Function Virtualization is considered one of the key technologies for developing the future mobile networks. In this paper, we propose a theoretical framework to evaluate the performance of an LTE *virtualized Mobility Management Entity* (vMME) hosted in a data center. This theoretical framework consists of i) a queuing network to model the vMME in a data center, and ii) analytic expressions to estimate the overall mean system delay and the signaling workload to be processed by the vMME. We validate our mathematical model by simulation. One direct use of the proposed model is vMME dimensioning, i.e., to compute the number of vMME processing instances to provide a target system delay given the number of users in the system. Additionally, the paper includes a scalability analysis of the system. In our study we consider the billing model and a data center setup of Amazon Elastic Compute Cloud service, and estimate experimentally the processing time of MME processing instances for different LTE control procedures. For the considered setup, our results show that a vMME is scalable for signaling workloads up to 37000 LTE control procedures per second for a target mean system delay of 1 ms. The database performance assumed imposes this limit in the system scalability.**

***Index Terms*—NFV, 5G, Scalability, virtualization, LTE, EPC, vMME**

I. Introduction

Nowadays, the telecom industry is considering Network Virtualization as one of the key technologies in the future 5G cellular networks. Network Functions Virtualization (NFV) offers the possibility of running the network functions on industry standard high volume servers (so-called commodity hardware) instead of using expensive, special purpose, and vendor-dependent hardware [1][2]. The decomposition of a service in a set of Virtual Network Functions (VNF) which can be executed in standard servers, allows for instantiating these VNFs in different network locations as needed. Concretely, NFV promises to enable organizations to: i) reduce capital and operational expenditures, ii) accelerate time-to-market of new services, iii) deliver agility and flexibility, and iv) scale up services on demand [1].

By way of illustration, nowadays the cellular networks are over-dimensioned in order to face the expected increase in the traffic load for the next years and considering the peak hours. The network entities are statically deployed and configured. Hence, there is a lack of network elasticity to deal with highly dynamic traffic patterns that might result in a waste of resources. Since NFV paradigm allows to create and scale network components on-demand, it can put an end to this problem. With the adoption of NFV, the mobile operators could adapt and optimize their resources in accordance with the given traffic conditions.

This work aims at performing a dimensioning and scalability analysis of an LTE virtualized Mobility Management Entity (vMME) in a data center. On the one hand, the purpose of the vMME dimensioning is to determine the minimal number of processing instances required at the data center so that a given target mean system response time can be guaranteed. Please note that dynamic resource provisioning is not addressed in this work, though the dimensioning may be part of such algorithms [3]. On the other hand, the scalability analysis of the vMME is intended to assess the productivity of the system, which depends on its running costs and performance in terms of throughput and delay.

In order to achieve these goals, in this paper we develop a mathematical framework to assess the mean system response time of a vMME given the control messages arrival rate and the system service rates. It will allow for estimating the time for a control message to be serviced. Our approach considers an 1:N mapping VNF implementation for the vMME [4]. Using this architectural option, the vMME processing instances are stateless facilitating the resources scaling, high availability, and load balancing. Since this implementation design follows the multi-tiered web services deployment scheme for cloud-based applications [4], we use a queuing network similar to [5] and [6] for modeling the vMME in a data center. In addition, we also provide analytic expressions to estimate the signaling workload to be processed by the vMME. We validate all this mathematical framework by simulation.

The main contributions of this paper are the following:

- The proposal of a theoretical model to compute the system response time of a vMME running in a data center.
- A theoretical characterization of the signaling workload generated by both the user's activity and the Machine-Type Communications (MTC) devices. We provide mathematical expressions to compute the rates of the LTE control procedures that generate most signaling load.
- Using our theoretical framework we perform dimensioning of a vMME. We verify by simulation that the proposed framework is useful for that use case. This might

The authors are with the Department of Signal Theory, Telematics and Communications of the University of Granada, Granada, 18071 Spain (e-mail: jpg@ugr.es, jjramos@ugr.es, pameigeiras@ugr.es, pam91@correo.ugr.es, juanma@ugr.es).

Manuscript received [date]; revised [date].

be a first step in designing dynamic resource provisioning algorithms [3].

- We also provide a scalability study of a vMME to investigate the evolution of the system productivity when it scales up its resources. In this work we consider the productivity as a metric that relates the performance, in terms of throughput and delay, and the running costs of the system.

In our study we consider both Human-Type Communications (HTC) and Machine-Type Communications (MTC). We use the data center setup and billing model of Amazon Elastic Compute Cloud (EC2). Based on this, we estimate experimentally the servicing rates of vMME processing instances for different LTE control procedures. We evaluate the vMME model by means of simulations considering the dense urban information society scenario of the METIS project [7] for 5G networks. We carried out the vMME scalability analysis and our results show that a vMME is scalable for signaling workloads up to 37000 LTE control procedures per second for a target average system delay of 1 ms. This limit in the system scalability is imposed by the database performance considered.

The paper is organized as follows. The next section provides some background and summarizes relevant literature. Section III presents the system model. Section IV reviews the LTE standard control procedures that are considered in this work. Section V describes the adopted traffic models. In Section VI, we propose the queuing model for the vMME, derive analytic expressions to compute the signaling workload and the mean system delay. This section also includes a simple vMME dimensioning algorithm. Section VII provides a theoretical analysis to assess the virtualized MME scalability. The system is simulated and evaluated in Section VIII where the proposed theoretical models are also validated. Finally, Section IX draws the main conclusions of the paper.

II. Background and Related Works

The Mobility Management Entity (MME) is the key control entity for the LTE EPC. It interacts with the evolved NodeB (eNodeB), Serving Gateway (S-GW), and Home Subscriber Server (HSS) within the EPC to realize functions such as Non-Access Stratum (NAS) signaling, user authentication and authorization, mobility management (e.g. paging, user tracking), and bearer management [8], among others.

Traditionally, MME was dimensioned to cope with the signaling workload expected for next years and considering the busy hours. Once the MME capacity was close to its limit (e.g., CPU load of 70%), its hardware was upgraded to meet future needs while maintaining the same software and architectural design. One of the main drawbacks of the traditional approach is the lack of elasticity. To put an end to this issue, MME can leverage NFV paradigm to scale up and down depending on the current signaling workload. Furthermore, NFV makes viable the adoption of distributed architectures for MME [9] by reducing provisioning costs.

Since NFV promises to bring substantial benefits to forthcoming mobile networks, there exists an intensive research work focused on this topic. There are works that have tackled the architectural and implementation issues of NFV in LTE Evolved Packet Core (EPC). In [10] the authors discuss the challenges and requirements imposed by the adoption NFV paradigm in mobile networks. Furthermore, they propose an NFV framework for EPC and suggest a regrouping of its VNFs in order to reduce the control signaling. The authors in [4] demonstrates that the implementation of EPC over a cloud infrastructure and providing it ”as a Service” is feasible. They also present different architectural options and carry out a thorough analysis comparing these options. The reference [11] describes a scheme for virtualization-based scaling of stateful network entities without interrupting user session continuity. Following this scheme and in order to prove its benefits, the authors design and implement an LTE virtualized Mobility Management Entity (vMME). The authors in [12] design and implement an architecture of a virtualized EPC tailored to the needs of the machine-to-machine services. They probe that their architecture proposal reduces CPU time consumption by up to 27% by reducing control message volume.

Others works have addressed the study of the feasibility of the virtualization of the EPC. For instance, the authors of [13] implement an entire EPC in general purpose processors. They argue the improved use of computational resources provided by NFV paradigm and show that servicing the synthetic workload generated by 50000 users is viable. In [14], the authors point out potential bottlenecks of a virtualized EPC (vEPC). To that end, they combine experimentation and analysis to demonstrate that the control plane signaling may severely interfere with the user plane packet processing.

As far as the NFV-based applications are concerned, Project Clearwater [15] is an open source implementation of the IP Multimedia Subsystem (IMS) standard. Clearwater proposes a cloud-oriented design tailored for deployment in NFV ecosystem, which claims to be massively scalable and exceptionally cost-efficient. It makes use of stateless load balancing, which allows all components scaling out horizontally.

Regarding the modeling of virtualized networks, the author in [16] proposes a model for service capabilities of composite network-Cloud service provisioning systems. Using deterministic network calculus, the aforementioned paper models these systems considering Latency-rate profile for the service components and a leaky bucket shaper to conform the user data traffic.

III. System Model

In this work, we assume a general access cellular network architecture based on NVF, which also supports mobility and MTCs. Although this architecture reuses the entities defined in LTE/EPC, we simplify them to allow its extension to other cellular architectures.

The overall system considered in this work is depicted in Fig. 1. The main entities are explained next.

A. The User Equipment (UE)

Let N_U be the number of UEs in our system. UEs are the terminals which allow each user to connect to the network via the eNodeB base stations. We assume that UEs move

This article has been accepted for publication in a future issue of IEEE Transactions on Vehicular Technology. Citation information: DOI 10.1109/TVT.2016.2608942, IEEE Transactions on Vehicular Technology. A preprint version of this manuscript is available at: http://ieeexplore.ieee.org/document/7565650/

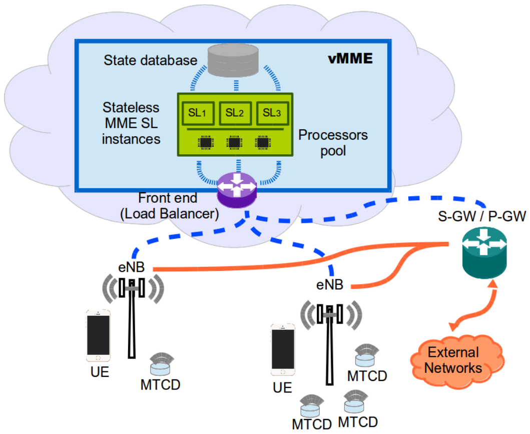


Fig. 1. Overall system model.

following a fluid-flow mobility model. The UEs run the users' applications which generate or consume network traffic, as described in Sec. V. The UE is able to initiate requests to the network by using control messages. The activity of the UE and the generation of network traffic also trigger the network control procedures.

B. MTC devices (MTCDs)

Let N_D denote the number of MTCDs in the system. We assume the following assumptions: MTCDs are placed in fixed locations, they send small data packets to centralized servers infrequently, and additionally they use the same procedures as the UEs do to send their data.

C. eNodeB stations (eNB)

They receive the UEs signaling and forwarding messages to the vMME. Each eNB contains a *user inactivity timer* with an expiration time of T_I. Using this timer, the eNB detects the users' inactivity (i.e., the user does not perform any data communication over a period of length T_I) and can release network resources.

D. The virtualized Mobility Management Entity (vMME)

The vMME is the main control entity of the network. It is in charge of maintaining the mobility state of the UE, bearer management, and user authentication and authorization, among other functions. To support this functionality, LTE standard defines several signaling procedures (i.e., NAS procedures), which imply an exchange of signaling messages between the vMME and other LTE entities (e.g., eNB, S-GW and HSS). When the vMME receives one signaling message, it processes it, and later the vMME sends a new message to the another entity (such as eNB or S-GW). If the procedure requires several steps, the entity sends another response message to the vMME. Let T_{IM} be the time between the vMME sends a control message to other LTE entity and the response message arrives at the vMME from that entity, where applicable. This time models the network delays and processing delay of the entity interacting with the MME.

As far as the vMME implementation is concerned, we consider the *1:N mapping* architectural option [4]. Thus, the vMME is split into 3 logical components: front-end (FE), MME service logic (SL), and state database (SDB). The FE acts as the communication interface with other entities of the network and balances the load among several MME SLs, which implement the processing of the different control messages. In this way, the vMME is seen as a single component by the rest of the network. The SDB stores the user session state making the MME SLs stateless.

With regard to the operation between SDB and SLs, we assume that when an MME SL instance finishes processing a control plane message, it saves the transaction state and or the updated user context into the SDB. When a subsequent request arrives at an MME SL instance, it first gathers the user context (e.g., for deciphering the message) and transaction state from the database to continue from. The user context consists of a set of information elements associated with the user that can be categorized into user ID, user Location, Security, and EPS Session/Bearer information [17]. As an example, let us consider the last message of a Handover (HR) procedure to be processed by the vMME. When an MME SL finishes processing this message, it will have to update some information of the user context (e.g., eNB UE S1AP ID, E-UTRAN Cell Global Identifier, and S1 Tunnel Endpoint Identifier for downlink) in the SDB.

This differs from vMME implementation based on Elastic Core Architecture [11], and it allows fully stateless MME SLs. Different messages of the same procedure for the same user can be processed by different MME SL instances. Therefore, the number of MME SL instances, denoted as m, can grow without affecting on in-session users.

When the processing capacity assigned to the vMME cannot withstand with the current control load, a new MME SL instance must be instantiated and a new processor is added to the processing resources pool. We presume *dedicated hosting*, i.e., each MME SL instance runs on a server or a subset of servers and a server is allocated to at most one MME SL instance at any given time [18]. For simplicity, we will assume that every processor in the data center facility provides the same computational power. Moreover, we consider that there are a single SDB and FE instances.

IV. Control Plane Procedures

There exist several signaling procedures in LTE that allow the control plane to manage the UE mobility and the data flow between the UE and *Packet Data Network Gateway* (P-GW). From all of them, we only concentrate on the ones that generate most signaling load [13].

In the following subsections, we describe the processing carried out by the MME during the control plane procedures that are considered in this work [19].

1) Service Request (SR): When a UE does not have available resources and new traffic is generated, either from this UE or from the network to this UE, the UE performs a

This article has been accepted for publication in a future issue of IEEE Transactions on Vehicular Technology. Citation information: DOI 10.1109/TVT.2016.2608942, IEEE Transactions on Vehicular Technology. A preprint version of this manuscript is available at: http://ieeexplore.ieee.org/document/7565650/

Service Request (SR) procedure. We focus on the UE-triggered SR. During this procedure the MME receives three different messages: an Initial UE Message (SR_1), an Initial Context Setup Response (SR_2), and a Modify Bearer Response (SR_3).

To process the Initial UE Message (SR_1) the MME has to carry out UE integrity check and message decrypting. Additionally, it generates identifiers for the bearers to be established. Moreover, it stores and retrieves parameters and variables related to the UE context. Some of them are included in the subsequent Initial Context Setup Request message. During the processing of the Initial Context Setup Response message (SR_2), the MME also retrieves information of the UE context and includes this information in the subsequent Modify Bearer Request message. The processing of the Modify Bearer Response (SR_3) is minimum as this message is only a confirmation.

2) Service Release (SRR): The Service Release (SRR) procedure is triggered by user inactivity. Its purpose is to release data radio bearers and downlink S1 bearer in the data plane, and radio and S1 signaling connections in the control plane for a UE. During the SRR, the MME processes three messages: a UE Context Release Request (SRR_1), a Release Access Bearers Response (SRR_2), and a UE Context Release Complete (SRR_3).

To process both the UE Context Release Request message (SRR_1) and the Release Access Bearers Request (SRR_2), the MME needs to retrieve information of the UE context and include this information in the subsequent messages. The processing of the UE Context Release Complete message (SRR_3) mainly implies the deletion of the bearer's context information by the MME.

3) X2-Based Handover (HR): The MME participates in the X2-based Handover (HR) during the handover completion phase. Its purpose is to switch the bearers' end point from the source to the target eNB. The MME receives two messages during this phase: a Path Switch Request message (HR_1) and a Modify Bearer Response (HR_2).

To process both the Path Switch Request message (HR_1) and the Modify Bearer Response (HR_2), the MME also needs to retrieve information of the UE context and include this information in the subsequent messages. To process the Path Switch Request message, the MME also needs to store new information such as the IDs of the new serving cell and new tracking area.

V. Traffic models

This section describes the traffic models considered in this work along with their statistical characterization.

A. Human type communication (HTC) traffic model

Let us define a *session* as the user activity elapsed between the instant the user launches a network application and the time instant he closes or stops it (Fig. 2).

Likewise, *Application Activity Period* (AAP) is defined as the time interval in which the application sends or receives all necessary data to perform a single task, such as download a web page, stream a video, or make a call. UE applications generate or consume traffic during the *application activity periods* of a *session*.

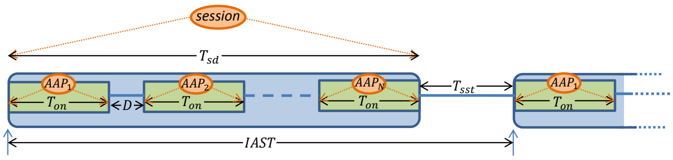


Fig. 2. HTC traffic model.

A session consists of N $AAPs$ of length T_{on} separated by $N-1$ *reading times*. A *reading time* (of length D) is the time period elapsed between two successive activity periods within the same *session*. During the *reading time*, the user does actions such as reading the downloaded web page or deciding the next video to watch.

Let us define *Inter-Arrival Session Time* ($IAST$) as the time interval between the start of two consecutive sessions. And let T_{sst} denote the *session standby time*, i.e., the time elapsed from the end of a session to the beginning of the next one.

We assume T_{sst} follows an exponential distribution with mean $\overline{T}_{sst} = (\overline{IAST} - \overline{T}_{sd})$ seconds, where T_{sd} is the session duration and $\overline{A}$ denotes $E[A]$, for any A.

Assuming that N, D and T_{on} are statistically independents, it holds that:

$$\overline{T}_{sd} = \overline{N} \cdot \overline{T}_{on} + (\overline{N} - 1) \cdot \overline{D}. \qquad (1)$$

Whenever a session begins, the user chooses a certain application with a given probability P_{app} (see Table I). Three types of applications are considered in this work: i) web browsing, ii) HTTP progressive video and iii) video calling. The specific values of P_{app} used for each application were computed from the percentages of total network traffic generated per type of traffic given in TC2 scenario of the METIS project [7]. Similarly, to estimate the data rates of the future mobile traffic, we have followed the predictions assumed in the METIS project [7]. Table I summarizes the statistical characterization of the considered application models, which are briefly described below.

1) Web Browsing: The characterization of this traffic is described in [22]. The amount of data downloaded for an *application activity period* (i.e., web page size) of a web browsing session is determined by the main object size (i.e. the HTML file), the number of embedded objects and their sizes. During a session, the number of downloaded web pages per session is set to follow a geometric distribution [23].

The download time is determined by the web page size, the link data rate, and the *parsing time*. The *parsing time* is defined as the time interval the web browser takes to parse the embedded objects.

We estimate the future web pages sizes by extrapolating the data series of [24], and scaling main objects size, accordingly.

2) HTTP progressive video: For this type of traffic, we adopt the YouTube model of [21], in which a video is transferred at a constant and limited rate during a *throttling phase* after an initial period of high downloading rate, called *initial burst*. The number of downloaded video clips per session is

This article has been accepted for publication in a future issue of IEEE Transactions on Vehicular Technology. Citation information: DOI 10.1109/TVT.2016.2608942, IEEE Transactions on Vehicular Technology. A preprint version of this manuscript is available at: http://ieeexplore.ieee.org/document/7565650/

TABLE I
TRAFFIC MODELS CHARACTERIZATION

Com. Type	Traffic Type	Parameters	Statistical Characterization
HTC ($\overline{IAST} =$ 1200 s [20])	Web browsing (HTTP) $P_{app} = 0.74$	Main Object Size	Truncated Lognormal Distribution: μ=15.098 σ=4.390E-5 min=100Bytes max=6MBytes
		Embedded Object Size	Truncated Lognormal Distribution: μ=6.17 σ=2.36 min=50Bytes max=2MBytes
		Number of Embedded Objects per Page	Truncated Pareto Distribution: mean=22 shape=1.1
		Parsing Time	Exponential Distribution: mean=0.13seconds
		Reading Time	Exponential Distribution: mean=30seconds
		Number of pageviews per session	Geometric Distribution: p=0.893 mean=9.312
	HTTP progressive video $P_{app} = 0.03$	Video Encoding Rate	Uniform distribution with ranges: $(2.5, 3.0)$Mbps / (4.0,4.5)Mbps / $(12.5, 16.0)$Mbps / $(20.0, 25.0)$Mbps, for equiprobable itags: 137 / 264 / 266 / 315 respectively.
		Video Duration	Distribution extracted from [21]
		Reading Time	Exponential Distribution: mean=30seconds
		Number of video views per session	Geometric Distribution: p=0.6 mean=2.5
	Video calling $P_{app} = 0.23$	Call Holding Time	Pareto Distribution: k=-0.39 s=69.33 m=0
		Number of calls per session	Constant = 1
MTC	Infrequent small data transmissions (Packet Size = 100 B)	Discretization time interval	$\Delta_T = 1$ sec
		Markov chain state transition matrix	$P = \begin{pmatrix} 1-p & q \\ p & 1-q \end{pmatrix}$ where $p = 6.75 \times 10^{-5}$ and $q = 1.47 \times 10^{-4}$
		Markov chain state rates	$\lambda_1 = 0.0015$ packets/s; $\lambda_2 = 0.065$ packets/s

set to follow a geometric distribution [25]. We assume that the *reading times* for this model and for web browsing are identically distributed.

The size of each video is calculated from its duration and encoding rate. The video encoding rate depends on the video format selected. Each video format, identified by an *itag* number, determines a container file format, an encoding algorithm, and a video resolution. To meet the METIS predicted data rates, we have considered the YouTube video formats with the highest encoding rates and resolutions.

The video download time (i.e., activity period) is determined by the bottleneck link data rate during the initial burst and limited by the media server during the throttling phase [21].

3) Video calling: In this application, a session starts when the user opens a video calling client app and makes a single call to someone else. This application generates constant bit rate traffic at 1.5 Mbps which is the recommended download-/upload speed of Skype for HD video calling.

The call duration or *call holding time* determines the application activity period duration. The statistical characterization for the call duration has been extracted from [26].

B. Machine type communication (MTC) traffic model

In this work we implement the MTC traffic model based on Markov-modulated Poisson processes (MMPPs) from [27], but without taking into consideration the coordinated behavior for MTC devices.

In this model, each MTC device with index $j = \{1, 2, ..., N_D\}$ is modeled by an MMPP. Let us define n as the time index resulting of time discretization $n = \frac{t}{\Delta_T}$ for any constant time interval Δ_T. An MMPP is a Poisson process modulated by the rate $\lambda_j^{MTC}[n]$, which is given by the state of a Markov chain $s_j[n]$. Then, $\lambda_j^{MTC}[n] = \lambda_i$ when $s_j[n] = i$, where $i = \{1, 2, ..., I\}$ denotes the index of Markov state and λ_i denotes a constant rate associated to the state i.

Assuming a constant packet size of 100 bytes for MTC devices, we use the parameters listed in Table I for this model. This setup is extracted from [27], which corresponds to a fleet management service case.

VI. VMME QUEUING MODEL

A. Model description

To model a vMME with a 1:N mapping architecture as described in Section III, we consider a queuing system based on [5] which models a typical cloud processing chain. We assume that all the MME SL instances have the same computation power. Table II provides the notation and main definitions for describing the queuing system.

On the one hand, the state database, the FE, which balances the control requests among the MME SL instances, and the output network interface are modeled with single processor queues, with service rates respectively denoted by μ_{SDB}, μ_{FE} and μ_{OI} (Fig. 3). On the other hand, the MME SL pool is modeled by a set of queues and processors that allow the parallel processing of the control messages.

B. Arrival rate calculations for signaling requests

In this section, we derive mathematical expressions to predict the arrival rate of signaling procedure requests to the vMME. It will depend on the activity of the HTC UEs and MTCDs. Let λ be the aggregate control messages arrival rate. Then, from the description of the control procedures of Section IV, λ is calculated as

$$\lambda = 3 \cdot \lambda_{SR} + 3 \cdot \lambda_{SRR} + 2 \cdot \lambda_{HR} \qquad (2)$$

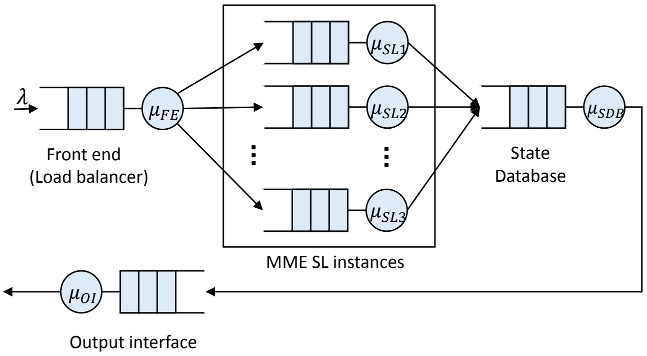


Fig. 3. vMME queue model.

TABLE II
PRIMARY DEFINITIONS.

Notation	Description
D	Reading time length.
N	Number of AAP per session.
N_U	Total number of HTC UEs.
N_D	Total number of MTC devices.
m	Number of MME SL instances.
λ	Total arriving rate.
λ_{HR}	Arriving rate of *Handover* requests.
λ_{SR}	Arriving rate of *Service Request* requests.
λ_{SRR}	Arriving rate of *Service Release Request* requests.
r_{cc}	Cell crossing rate.
T	Total system processing time.
T_{max}	Target mean system delay.
T_{DB}	Database processing time.
T_{FE}	Front-end processing time.
T_{SL}	MME SL processing time.
T_{on}	Duration of the AAP.
T_{OI}	Output interface processing time.
T_{sd}	Session duration.
T_{sst}	Duration of the session standby time.
μ_{SDB}	Average database service rate.
μ_{FE}	Average front-end service rate.
μ_{OI}	Average output interface service rate.
μ_{SL}	Average MME SL instance service rate.
O_{bw}	Egress link capacity.
O_{size}	Average response packet size.

Where λ_{SR}, λ_{SRR}, and λ_{HR} denote the mean arrival rates for SR, SRR and HR procedures, respectively. These rates can be expressed in terms of mean arrival rates per user λ_U^h and per MTC device λ_S^h for procedure $h \in \{SR, SRR, HR\}$. Then,

$$\lambda_h = N_U \cdot \lambda_U^h + N_D \cdot \lambda_S^h \tag{3}$$

We suppose no mobility for MTCDs, thus $\lambda_S^{HR} = 0$. We describe each arrival rate in the following paragraphs.

1) λ_U^{SR} and λ_U^{SRR} calculations: An SR procedure occurs whenever a UE is going to start an AAP without having network resources assigned. When an AAP finishes, a *user inactivity timer*, whose value is denoted as T_I, starts. Let X denote the time elapsed between the end of an AAP and the beginning of the next one, regardless these activity periods belong to the same session or not. If $X \geq T_I$, the SRR procedure is triggered.

Since each SR has a corresponding SRR, it holds that

$$\lambda_U^{SRR} = \lambda_U^{SR} \tag{4}$$

Let $\overline{N}_S^{SR}$ be the mean number of SRs procedures per session, which is given by:

$$\overline{N}_S^{SR} = \overline{N} \cdot P(X > T_I) \tag{5}$$

where $P(X > T_I)$ is the probability that the inactivity timer expires.

Finally, let us $\lambda_S = 1/\overline{IAST}$ denote the sessions rate, i.e., the mean number of sessions per unit time. It holds that

$$\lambda_U^{SR} = \lambda_S \cdot \overline{N}_S^{SR} \tag{6}$$

Since for the first activity period $X = T_{sst}$ and for the following $N-1$ ones $X = D$, then

$$\lambda_U^{SR} = \lambda_S \cdot ((\overline{N} - 1) \cdot P(D > T_I) + P(T_{sst} > T_I)) \tag{7}$$

2) λ_U^{HR} calculation: Assuming that each eNodeB serves only one cell, an HR procedure takes place when a user performs a cell change while being active. A user is considered active from the triggering of the SR procedure to the triggering of the associated SRR event. Let P_{UA} denote the probability that a user is active at a given time, and let $\overline{r}_{cc}$ be the mean user cell crossing rate, i.e., the average number of cell crossings per unit time. Thus, the mean arrival rates per user for HR is

$$\lambda_U^{HR} = \overline{r}_{cc} \cdot P_{UA} \tag{8}$$

Assuming that each user moves according to the fluid-flow mobility model, i.e., at constant speed with random direction uniformly distributed between $[0, 2\pi)$, it holds that

$$\overline{r}_{cc} = \frac{\overline{v} \cdot B}{\pi \cdot S} \tag{9}$$

where $\overline{v}$ is the mean user speed, and B is the perimeter of the cell coverage area S.

To compute P_{UA}, let T_{ua} denote the temporal extension of an AAP, defined as the time interval elapsed from the end of an AAP to the inactivity timer expiration or to the beginning of the next activity period, whichever comes first, that is, $T_{ua} = X$ if $X \leq T_I$ and $T_{ua} = T_I$ otherwise. Then, T_{ua} will follow the same distribution as X, but upper truncated to the value of T_I. Thereby, the expected value of T_{ua} can be computed as

$$\overline{T}_{ua}(X) = T_I \cdot P(X > T_I) + \int_0^{T_I} x \cdot f_X(x)\, dx \tag{10}$$

Therefore, P_{UA} is λ_S times the amount of time that a user is active within a session:

$$P_{UA} = \lambda_S \cdot (\overline{N} \cdot \overline{T}_{on} + (\overline{N} - 1) \cdot \overline{T}_{ua}(D) + \overline{T}_{ua}(T_{sst})) \tag{11}$$

3) λ_S^{SR} and λ_S^{SRR} calculations: An SR procedure occurs whenever an MTC device is going to transmit a new packet without having network resources allocated. Let $P(t_r > T_I)$ denote the probability that the time interval between two packets transmission for any MTC device t_r be greater than inactivity timer value T_I. It holds that

$$\lambda_S^{SR} = P(t_r > T_I) \cdot \sum_{i=1}^{I} \lambda_i \cdot \pi_i \tag{12}$$

where π_i is the probability of the state i and I the number of states of the Markov chain. For our case $I = 2$, $\pi_1 = \frac{q}{p+q}$ and $\pi_2 = \frac{p}{p+q}$.

Again, it verifies that

$$\lambda_S^{SRR} = \lambda_S^{SR} \tag{13}$$

C. vMME Response Time

To estimate the system response time, we suppose that there is a single Poisson arrival stream with arrival rate λ which represents the control plane messages sent to the vMME (see Fig. 3), and that all the processing elements of the system are exponential servers with a service rate μ_i calculated from their mean service time $\bar{t}_i$ as $\mu_i = [\bar{t}_i]^{-1}$. Although these are strong assumptions, it allows us to assume a Jackson's open network, what eases obtaining analytical expressions. As we will show later in Section VIII, the proposed vMME analytic model provides a fairly good approximation to the values obtained by simulation.

Let $\overline{T}_D$ denote the mean response time of the FE node and let $\overline{T}_{OI}$ denote the output interface mean response time. Let $\overline{T}_{SL}$ denote the mean response time of the servicing nodes. And let $\overline{T}_{DB}$ denote the average processing time of the database. If the modeled system is assumed to be an open Jackson network, the mean response time $\overline{T}$ of the entire system of Fig. 3 can be estimated by

$$\overline{T} = \overline{T}_{FE} + \overline{T}_{SL} + \overline{T}_{DB} + \overline{T}_{OI} \tag{14}$$

1) $\overline{T}_{FE}$ calculation: : The front-end is modeled with an *M/M/1* queue. Therefore, $\overline{T}_{FE}$ can be calculated as

$$\overline{T}_{FE} = \frac{(\mu_{FE})^{-1}}{1 - \lambda/\mu_{FE}} \tag{15}$$

where the μ_{FE} is the mean service rate of the front-end node.

2) $\overline{T}_{SL}$ calculation: The services nodes are modeled with an $M/M/m$ queue, and therefore their mean response time are computed as

$$\overline{T}_{SL} = \mu_{SL}^{-1} + \frac{C(m,\rho)}{m \cdot \mu_{SL} - \lambda} \tag{16}$$

where $\rho = \frac{\lambda}{\mu_{SL}}$, and $C(m,\rho)$ represents the Erlang's C formula calculated as

$$C(m,\rho) = \frac{\left(\frac{(m\cdot\rho)^m}{m!}\right)\cdot\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{m-1}\frac{(m\cdot\rho)^k}{k!} + \left(\frac{(m\cdot\rho)^m}{m!}\right)\cdot\left(\frac{1}{1-\rho}\right)} \tag{17}$$

The average service rate of the NFV procedures, μ_{SL} is equal to $[\bar{t}_{SL}]^{-1}$. We estimate the average service time $\bar{t}_{SL}$, by weighting the processing time of each procedure according to its frequency, as explained in Section IV. Consequently,

$$\begin{aligned}\bar{t}_{SL} =& \frac{\lambda_{SR}}{\lambda}\cdot(t_{SR_1} + t_{SR_2} + t_{SR_3}) + \\ & \frac{\lambda_{HR}}{\lambda}\cdot(t_{HR_1} + t_{HR_2}) + \\ & \frac{\lambda_{SRR}}{\lambda}\cdot(t_{SRR_1} + t_{SRR_2} + t_{SSR_3})\end{aligned} \tag{18}$$

where t_{SR_i} is the processing time of the i^{th} step of the Service Request procedure, t_{SRR_i} is the processing time of the i^{th} step of the Service Release Request procedure, and t_{HR_i} is the processing time of the i^{th} step of the Handover procedure.

3) $\overline{T}_{DB}$ calculation: The processing time of the database stage can be estimated by:

$$\overline{T}_{DB} = \frac{1/\mu_{SDB}}{1 - \lambda/\mu_{SDB}} \tag{19}$$

where μ_{SDB} is the service rate of the database server.

4) $\overline{T}_{OI}$ calculation: Finally, the output interface service rate μ_{OI} is calculated as O_{bw}/O_{size}, where O_{bw} is the output link bandwidth and O_{size} is the average packet size of responses. Therefore,

$$T_{OI} = \frac{(O_{bw}/O_{size})^{-1}}{1 - (\lambda)/(O_{bw}/O_{size})} \tag{20}$$

D. vMME dimensioning criterion

The analytic proposed model is useful for dimensioning of a virtualized MME. For example, it can provide the minimum number of MME SL instances m needed to achieve a target mean system delay $\overline{T}_{max}$ for a given number of UEs N_U and MTCDs N_D. Assuming the services rates μ_{SDB}, μ_{FE}, and μ_{OI} are high enough to withstand the required MME signaling load, it holds that:

$$m = \min\{M : \overline{T}(\lambda, M) \leq \overline{T}_{max}, M \in \mathbb{N}\} \tag{21}$$

Therefore, m can be found with a simple algorithm that interactively increases the number of MME SL instances until it obeys that $\overline{T}(\lambda, m) \leq \overline{T}_{max}$.

VII. vMME Scalability Analysis

The virtualized MME proposed in this work is a distributed system. To complete the study of this system, we assess its scalability in this section. To that end, we adopt the scalability metric defined in [28]. This metric is based on productivity: the distributed system is scalable if the productivity is maintained as the system scale changes.

The scalability metric for one system at two different scale factors k_2 and k_1, noted as $\psi(k_1, k_2)$, is defined as the ratio between the productivity of two systems at scale k_2 and k_1 [28]:

$$\psi(k_1, k_2) = F(k_2)/F(k_1) \tag{22}$$

where the productivity $F(k)$ represents the throughput delivered by the system over cost incurred per second for the scale factor k, calculated as:

$$F(k) = \frac{\lambda(k)\cdot f(k)}{C(k)} \tag{23}$$

where $\lambda(k)$ denotes the average throughput attained at scale k, $C(k)$ denotes the running costs of the system at scale k, and $f(k)$ is a function of some appropriate system measures. In this work we use the $f(k)$ function defined in [28], which calculates the average response time $\overline{T}(k)$ compared to a target value $\widehat{T}$:

$$f(k) = 1/(1 + \overline{T}(k)/\widehat{T}) \tag{24}$$

This article has been accepted for publication in a future issue of IEEE Transactions on Vehicular Technology. Citation information: DOI 10.1109/TVT.2016.2608942, IEEE Transactions on Vehicular Technology. A preprint version of this manuscript is available at: http://ieeexplore.ieee.org/document/7565650/

Generally, for a given system, the scalability metric is defined in absolute terms and denoted by $\psi(k)$, since the value of $\psi(k_1, k_2)$ at k_1 is fixed at a known value. In this case, $\psi(k)$ is interpreted as follows. If $\psi(k) = 1$, the system perfectly scales with k. If $\psi(k) > 1$, then the system scales positively with k. If $\psi(k) < \gamma$, the system does not scale. In this work, we adopt $\gamma = 0.8$ as in [28].

A strategy for scaling up the system is defined by the scaling factor k and several scaling variables which depend on k. In this work, we set as the scaling variable $m = k$. Therefore, the reference scale factor k_1 corresponds to the system with one NFV instance. Additionally, for a given k factor, the other system variables are configured to serve the maximum number of UE within a T_{max} service delay budget.

To consider a realistic cost function for our vMME cloud-based system, we consider the Amazon EC2 Service billing model [29]. To that end, let $C_{ci}(m)$ denote the per instance computing cost, let $C_b(m)$ denote the load-balancer service cost, and let $C_{db}(m)$ denote the database accessing cost. Then, the total cost $C(m)$ is

$$C(m) = C_b(m) + m \cdot C_{ci}(m) + C_{db}(m) \tag{25}$$

where m is the number of virtualized MME SL instances. Each element's cost includes a rental fee, a storage charge, and a per transaction or throughput price, as we describe next. The exact cost calculation depends on the cloud's billing model. To complete our study, Section VIII includes a numerical example that shows the practical applicability of the conducted analysis.

Cost $C_{ci}(m)$ includes a per unit time billing costs depending on the type of processor $C_{ci_{type}}(m)$, the cost of the outgoing traffic sent to Internet $C_{ci_{thro}}(m)$ per unit time, and the per computing instance storage cost $C_{ci_{stor}}(m)$:

$$C_{ci}(m) = C_{ci_{type}}(m) + C_{ci_{stor}}(m) + C_{ci_{thro}}(m) \tag{26}$$

The database accessing cost $C_{db}(m)$ includes a rental fee per unit time $C_{db_{type}}(m)$, the cost per data capacity $C_{db_{stor}}(m)$, and a fee per transactions per unit time $C_{db_{trans}}(m)$.

$$C_{db}(m) = C_{db_{type}}(m) + C_{db_{stor}}(m) + C_{db_{thro}}(m) \tag{27}$$

The considered cloud service provides a load balancer service. Its cost $C_b(m)$ is charged by activation time $C_{b_{type}}(m)$ and served throughput $C_{b_{thro}}(m)$.

$$C_b(m) = C_{b_{type}}(m) + C_{b_{thro}}(m) \tag{28}$$

VIII. Numerical results

In this section, some numerical results are reported. It aims at validating the proposed mathematical framework to model a vMME and evaluating its scalability.

Our evaluation framework includes two software tools: a generator of procedure calls and a queuing system simulator.

The generator of procedure calls is implemented in the ns-3 simulator [30]. It implements the traffic models presented in Section V and the corresponding network signaling. The simulation scenario considered for each user is based on the dense urban information society scenario of the METIS project [7]. It is composed of 12 eNodeBs distributed regularly in a $4x3$ grid over a rectangular area of size $387\,m\,x\,552\,m$. The coverage area for each eNB is rectangular with dimensions of $138\,m\,x\,129\,m$. The users move across the area following a fluid-flow mobility model. The user speed is uniformly distributed between 0 and $4.2\,m/s$.

TABLE III
Parameters Configuration

RAN topology	
eNBs layout	Regular Grid 387 m x 552 m
eNB coverage area	138 m x 129 m
Number of eNBs	12
UE mobility	
Mobility model	Fluid-flow model
Speed	Uniform distribution (0, 4.2) m/s
EPC delays	
One-way delay (eNB → vMME)	7.5 ms
T_{IM} (vMME $\rightleftharpoons$ [eNB | S-GW])	15 ms
$\overline{T}_{max}$	1 ms
Service rates	
FE service rate (μ_{FE})	120000 packets per second
SDB service rate (μ_{SDB})	100000 transactions per second
OI service rate (μ_{OI})	5000000 packets per second

The percentage of traffic generated for each type of application has been adjusted to meet the simulation guidelines of METIS project (see Table I) [7]. All users have an independent and constant uplink and downlink data rate of $300\,Mbps$ [7]. During the simulation, each control procedure generates control messages which are dumped to a trace file.

The queuing system simulator implements the queuing model presented in Section VI-A using the Matlab Simulink framework. The queuing model is fed with the traces produced by the previous tool. The load balancer has a service rate of 120000 packets per second [31]. The database service rate has been obtained by assuming that the database deployed in the Amazon Cloud is the Amazon Aurora database [29], which is reported to serve 100000 transactions per second [32]. The output interface is a 10G Ethernet that serves up to 5000000 packets per second (i.e., assuming an average packet size of 250 Bytes for the control messages generated by the vMME).

In the Simulink model, we included an infinite server ($G/D/\infty$ queue), which models the one-way transmission delay and processing times of the network nodes from any eNB to the vMME. This delay was set to 7.5 ms. Another similar server was used to implement the parameter T_{IM} (i.e., the time between the vMME sends a control message to other LTE entity and the response message arrives at the vMME from that entity). T_{IM} was fixed to 15 ms.

A. vMME processing time (T_{SL}) estimation

In this section we calculate the MME SL processing times for each control message, what will be used as the MME SL average service time (and its respective service rate μ_{SL}). Given a CPU processing capacity, we can estimate the delay of processing a message by assessing the average number of CPU instructions required for running a particular procedure.

To do this, we have considered the CPU characteristics of a real cloud service configuration from the *Amazon Elastic Compute Cloud (EC2)* [29]. Additionally, we have implemented in C the code of the functions which are invoked in the vMME for each procedure.

This article has been accepted for publication in a future issue of IEEE Transactions on Vehicular Technology. Citation information: DOI 10.1109/TVT.2016.2608942, IEEE Transactions on Vehicular Technology. A preprint version of this manuscript is available at: http://ieeexplore.ieee.org/document/7565650/

TABLE IV
NUMBER OF INSTRUCTIONS AND PROCESSING TIMES IN *m3.xlarge* INSTANCE.

Procedure	Number of Instructions	Processing Time, T_{SL} (μs)
SR_1	1.45e+06	127.4
SR_2	1.07e+06	94.0
SRR_1	1.07e+06	94.0
SRR_2	1.07e+06	94.0
SRR_3	1.06e+06	93.2
HR_1	1.07e+06	94.0
HR_2	1.07e+06	94.0

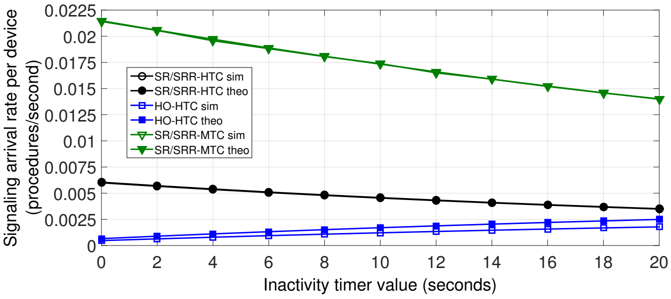


Fig. 4. Control procedures arrival rates versus user inactivity timer.

Although our implementation may differ from complete MME VNF implementations, we think that our version executes similar tasks as those ones.

After compiling the code, we measured the number of CPU instructions executed for every procedure by means of profiling tools. The results drew that the number of run instructions for the different control messages is very similar (see Table IV). Table IV also provides the delays calculated for the *EC2 m3.xlarge* virtual instance of the Amazon EC2 service [29]. The average computing capacity of this type of instance is $11.38 \cdot 10^9$ float operations per second [33].

B. Signaling Procedures Rate

To characterize the control messages arrival rate at the vMME, we generated signaling traces for 20000 UEs and 20000 MTCDs for several *inactivity timer* (T_I) values.

The obtained results are depicted in Fig. 4. It shows the mean arrival rates for the different signaling procedures, by using the provided theoretical model -referred to as label *theo*- and after the conducted simulations -referred to as label *sim*- as a function of the inactivity timer T_I.

Results show that the SRs and SRRs rates decrease with T_I for both HTC and MTC traffics. One possible explanation is because the higher the value of the inactivity timer, the smaller the probability the timer runs out within an inter AAP. Thus, the user stays in the connected state between consecutive AAPs, avoiding the need for triggering procedures to reserve and release resources. Conversely, the HRs rate slightly increases with the timer value, since in these cases the user remains connected longer after an AAP. Consequently, there is a higher chance that a user will be in connected state when a cell crossing event takes place.

TABLE V
RMSE FOR PREDICTED ARRIVAL RATE PER DEVICE (SEE FIG. 4).

$RMSE(\lambda_U^{SR})$	$RMSE(\lambda_U^{HR})$	$RMSE(\lambda_S^{SR})$
$4.07 \cdot 10^{-5}$	$15.0 \cdot 10^{-4}$	$6.65 \cdot 10^{-5}$

Please note that the amount of signaling traffic generated by MTCs is higher than for HTCs case. From Fig. 4, it is observed that $\lambda_U^{SR} = 0.0045$, $\lambda_U^{HR} = 0.0012$ and $\lambda_S^{SR} = 0.0173$ procedures per second and terminal for $T_I = 10s$. That means each MTC device generates about 3.5 times more control messages than an HTC UE.

For the schemes depicted in Fig. 4, Table V shows the root-mean-square error (RMSE) between the experimental rates (obtained after simulations) and predicted ones (using the proposed model). It demonstrates that the analytic expressions (e.g., Equations 7, 8, and 12) fit the experimental data obtained by simulation.

The higher prediction error for the HR procedure rate is due to the fluid-flow mobility model implementation: a bounce-back strategy is employed when a user reaches an edge of the geographical area. That decreases the $\overline{r}_{cc}$ per user in comparison with the predicted by the fluid flow model expression.

C. System Delay

Most mobile networks standards requirements define a delay budget to perform the control procedures. In order to evaluate the delay of our system and to compare the experimental results with the theoretical ones, we generated a signaling trace for 1200000 UEs and a $T_I = 10s$.

Two scenarios were considered: i) one with one MTC device per each UE (referred to as *scenario 1*), and ii) the other with three MTC devices per each UE (referred to as *scenario 2*).

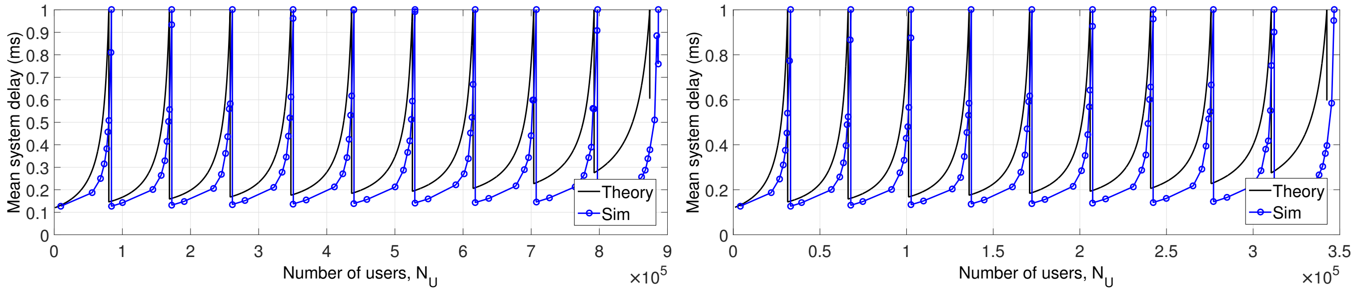
Figs. 6 depict the mean system delay versus the number of users for both theory (Equation 14) and simulation for the two scenarios. The curves were generated using the dimensioning algorithm introduced in subsection VI-D for $T_{max} = 1\,ms$.

As a general trend, given a number of MME SLs, the system delay grows with the number of users. There is a point where the number of MME SL instances cannot withstand the control messages arrival rate and the system delay shoots up. When $T = T_{max}$, in order to limit the system delay, a new MME SL must be instantiated to cope with the control plane workload. This fact explains the periodic spiky pattern of Fig. 6.

Visibly, though simulation and theoretical results show a similar shape, delays are smaller in case of the simulation. This is due to assumptions of the theoretical system differ from those adopted in our simulation model implementation. For instance, the simulation model considers deterministic service rates.

The root-mean-square error between simulation and theoretical results are 0.34 and 0.35 ms for scenario 1 and 2, respectively. Notably, it can be observed that the error increases with m (the number of MME SLs). That trend can be explained because the theoretical delay impact contribution

This article has been accepted for publication in a future issue of IEEE Transactions on Vehicular Technology. Citation information: DOI 10.1109/TVT.2016.2608942, IEEE Transactions on Vehicular Technology. A preprint version of this manuscript is available at: http://ieeexplore.ieee.org/document/7565650/



(a) One MTC device per each UE (Scenario 1).

(b) Three MTC devices per each UE (Scenario 2).

Fig. 5. Mean system delay vs number of users.

of the database begins to be noticeable, earlier than in the simulation setup.

D. vMME Dimensioning

A major application of the proposed theoretical framework is vMME dimensioning. That is, to predict the minimum number of MME SL instances required to guarantee $\overline{T} = \overline{T}_{max}$ given the number of UEs and MTC devices.

Let $N_U^{max}(m, \overline{T}_{max})$ denote the maximum number of UEs supported by the vMME (i.e., vMME capacity) depending on the number of SL instances m and the target mean system delay $\overline{T}_{max}$. To assess the goodness of our mathematical model, we computed $N_U^{max}(m, \overline{T}_{max})$ versus the number of MME SL instances m for $T_{max} = 1$ ms for both scenarios 1 and 2 (see Fig. 6(a)). The relative error between the theoretical and experimental curves ranges roughly from 0.5% to 5.5%. Therefore, we can conclude that the mathematical model is useful for dimensioning purposes. Furthermore, in general, the error decreases as m increases, except for $m = 10$. As it was mentioned in the previous section, this fact can be explained because the theoretical delay impact contribution of the database starts to be remarkable, earlier than in the simulation setup.

Since $\overline{T}_{max}$ might depend on the Quality of Service (QoS) requirements of the service and scenario considered, we assessed the vMME capacity $N_U^{max}(m, \overline{T}_{max})$ for different values of m and $\overline{T}_{max}$. Specifically, the considered ranges for $\overline{T}_{max}$ and m were 500 μs to 3 ms and 1 to 10 instances, respectively. The results obtained show that the vMME capacity does not differ significantly in the range of values considered for $\overline{T}_{max}$. For $\overline{T}_{max} = 500\,\mu s$ vMME capacity decreases 0.94% and 1.25% in comparison with $\overline{T}_{max} = 3\,ms$ case for Scenarios 1 and 2, respectively.

Finally, we also assessed the impact of average user mobility speed $\overline{v}$ on the vMME capacity (see Fig. 6(b)). As we assumed MTCs without mobility, we have considered Scenario 1 because in it the number of sensors per UE is lower. Thus, we can appreciate better the effects of mobility on MME capacity. Given that we assumed a fluid-flow model, the relationship between λ_U^{HR} and $\overline{v}$ is linear. Therefore, for the speeds considered $\overline{v} = 7.56\,km/h$, $\overline{v} = 25\,km/h$, and $\overline{v} = 50\,km/h$ the average handover rates per user are $\lambda_U^{HR} = 0.0012$, $\lambda_U^{HR} = 0.0040$, and $\lambda_U^{HR} = 0.0080$ procedures per second, respectively. We obtained that the capacity of the vMME decreases 6.26% for a doubling of $\overline{v}$.

E. Scalability Analysis

The scalability assessment of the system modeled depends on the exact cost function of the supporting cloud service.

As an example, we consider the Amazon EC2 Service, with the costs and configuration detailed in Table VI. We assume a medium sized CPU instance *m3.xlarge* with an average $11.38 \cdot 10^9$ float operations per second [33]. Our setup also includes the Amazon Aurora database [32], which is reported to provide 10^5 updates/s transactions. The target mean system delay is set $\overline{T}_{max} = 1\,ms$.

Assuming an on demand cloud service, Fig. 7 depicts the running costs of the system (measured in \$/s) for the selected configuration. It includes three scenarios for different UE to MTC device ratio. Interestingly, in general, it shows that the running cost of the virtualized vMMEs is almost linear with the number of users in the system. Nevertheless, note that the overhead costs of deploying new instances hinder the system scalability regarding the number of MME SL instances.

Finally, for testing the scalability of the proposed virtualized system in Fig. 8 the scalability metric $\psi(k)$ (Section VII) is depicted as a function of the number of MME SL instances.

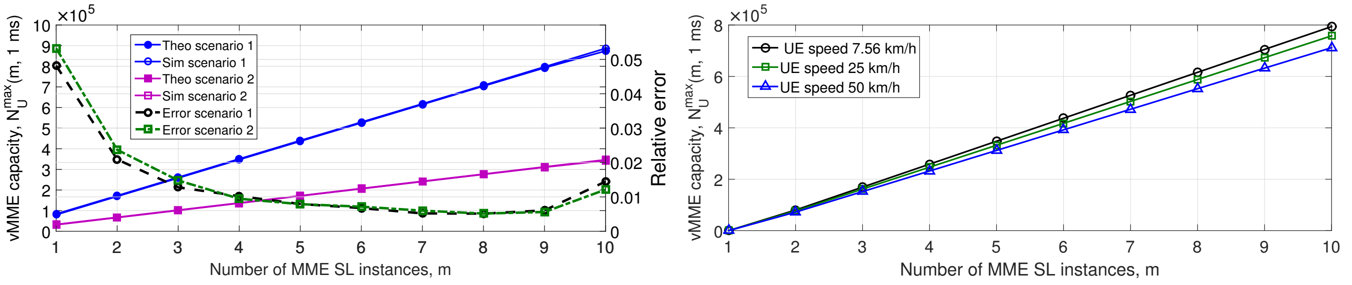
Interestingly, for the configuration considered, the system is positively scalable regarding the number of MME SL instances for $m < 10$. However, beyond that point, it holds that $\psi(k) < 1$; in other words for more than 9 MME SL instances, the system is not perfectly scalable. This limit stem from the database utilization reaches about 100% of its capacity. At that point, it would be necessary to scale up the database.

Nevertheless, recall from Fig. 6(a) that the system can serve roughly 900000 UEs and 900000 sensors for Scenario 1 and more than 325000 UEs and $3 \cdot 325000$ MTCDs for Scenario 2 (i.e., this is the equivalent of a signaling workload around 37000 LTE control procedures per second) for $\overline{T}_{max} = 1\,ms$ and $m = 10$, where the MME still scales positively. The vMME capacity obtained is in the same order of magnitude as non-virtualized MME solutions [34].

IX. Conclusions and Future Work

In this paper, we propose a theoretical framework to assess the system response time of a vMME running in a data center.

This article has been accepted for publication in a future issue of IEEE Transactions on Vehicular Technology. Citation information: DOI 10.1109/TVT.2016.2608942, IEEE Transactions on Vehicular Technology. A preprint version of this manuscript is available at: http://ieeexplore.ieee.org/document/7565650/



(a) vMME capacity versus the number of MME SL instances for both scenarios 1 and 2.

(b) vMME capacity versus the number of MME SL instances for different UE speeds (Scenario 1).

Fig. 6. vMME capacity.

TABLE VI
CLOUD SERVICE CONFIGURATION AND COST CALCULATION.

Cost	Configuration	Calculation
$C_{ci_{type}}(k)$	*m3.xlarge* instance rental (0.266$/hour)	$0.266/3600$
$C_{ci_{stor}}(k)$	Local storage per month (10GB), and optimized data access (0.025$/hour).	$10 \cdot 0.10 + 0.025/3600$
$C_{ci_{thro}}(k)$	Supposing $I_{size} = 200$ bytes, this is the data sent from the datacenter, calculated as $\lambda \cdot 200$	0.000($)/GB First GB/month 0.090($)/GB Up to 10 TB/month 0.085($)/GB Next 40 TB/month 0.070($)/GB Next 100 TB/month 0.050($)/GB Next 350 TB/month
$C_{db_{type}}(k)$	Aurora $db.r3.8xlarge$ instance (4.64$/hour)	$4.64/3600$
$C_{db_{stor}}(k)$	0.1$ per GB/month, for a total database size of $N_U \cdot 1KB$.	$(0.1 \cdot N_U \cdot 1024 \cdot \lambda/1e9)/2628000$
$C_{db_{thro}}(k)$	0.2$ per million transactions/month	$0.2 \cdot \lambda/1e6$
$C_{b_{type}}(k)$	Service fee of 0.025$/month	$0.025/2628000$
$C_{b_{thro}}(k)$	0.008$ per GB serviced, supposing $O_{size} = 200$ Bytes	$\lambda \cdot 0.008 \cdot 200/1e9$

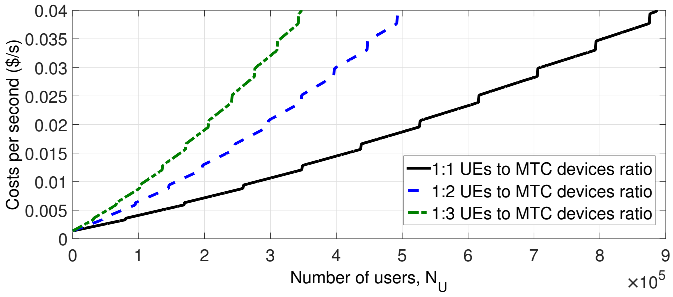


Fig. 7. Cost per second vs number of users.

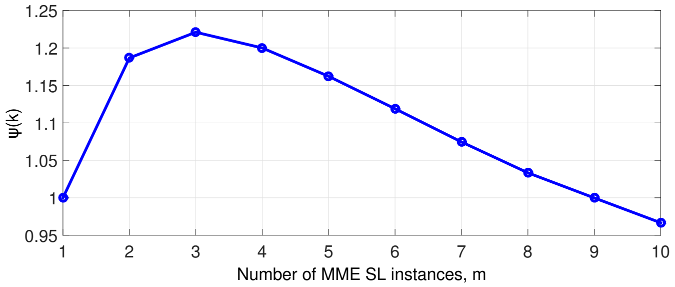


Fig. 8. Scalability $\Psi(k)$ vs number of MME SL instances.

This framework includes: i) a queuing network to model a vMME in a data center, and ii) expressions to estimate the rates of different signaling procedures (e.g., SR, SRR, and HR) per UE and MTC device, as well as to estimate the overall system delay. We have validated this framework by simulation. Additionally, we have estimated the processing time of a vMME for the different types of LTE control procedures considered.

Using this framework, a vMME dimensioning procedure is provided to predict the number of MME SL instances given a system delay budget (T_{max}), the number of UEs, and the number of MTC devices. After the conducted experiments, results show that our mathematical model is accurate for that purpose. Specifically, we have obtained a relative error between the theoretical and simulation results below 5.5%.

Finally, we have carried out a scalability analysis of the system. The reported results for the typical configuration considered suggest that MME virtualization is scalable for signaling workloads up to 37000 control procedures per second considering a data center with commodity hardware. This limit stem from the database utilization reaches its maximum. In order to continue the scalability analysis beyond this point, it would have to consider a strategy to scale the state database.

Regarding the future work, several challenges lie ahead. One of the main challenges is the design of a dynamic capacity provisioning algorithm for the vMME. This is to scale up or down the resources (e.g., vMME SL instances) assigned to the vMME depending on the fluctuations in signaling workload. Mobile network traffic exhibits long-term variations such as time-of-day or seasonal effects, as well as short-term fluctuations caused by unexpected events.

Acknowledgment

This work is partially supported by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund (project TIN2013-46223-P), and the Spanish Ministry of Education, Culture and Sport (FPU grant 13/04833).

References

[1] ETSI NFV ISG. (2012, October) Network Functions Virtualization-Introductory White Paper. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf. Last accessed: September 2015.

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[3] D. Pompili, A. Hajisami, and H. Viswanathan, "Dynamic provisioning and allocation in Cloud Radio Access Networks (C-RANs)," *Ad Hoc Networks*, vol. 30, pp. 128–143, July 2015.

[4] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "EASE: EPC as a service to ease mobile core network deployment over cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, March 2015.

[5] J. Vilaplana, F. Solsona, I. Teixido, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 492–507, 2014.

[6] P. Lama and X. Zhou, "Autonomic Provisioning with Self-Adaptive Neural Fuzzy Control for Percentile-Based Delay Guarantee," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 8, no. 2, pp. 9:1–9:31, July 2013.

[7] M. S. Guideline, "METIS Deliverable D6.1 Simulation Guidelines," October 2013.

[8] *General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Access*, 3GPP TS 23.401 Rel 12, 2014.

[9] X. An, F. Pianese, I. Widjaja, and U. Günay Acer, "DMME: A Distributed LTE Mobility Management Entity," *Bell Labs Technical Journal*, vol. 17, no. 2, pp. 97–120, September 2012.

[10] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: State of the Art, Challenges, and Implementation in Next Generation Mobile Networks (vEPC)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, December 2014.

[11] Y. Takano, A. Khan, M. Tamura, S. Iwashina, and T. Shimizu, "Virtualization-Based Scaling Methods for Stateful Cellular Network Nodes Using Elastic Core Architecture," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, December 2014, pp. 204–209.

[12] H. Baba, M. Matsumoto, and K. Noritake, "Lightweight Virtualized Evolved Packet Core Architecture for Future Mobile Communication," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2015, pp. 1811–1816.

[13] B. Hirschman, P. Mehta, K. B. Ramia, A. S. Rajan, E. Dylag, A. Singh, and M. Mcdonald, "High-Performance Evolved Packet Core Signaling and Bearer Processing on General-Purpose Processors," *IEEE Network*, vol. 29, no. 3, pp. 6–14, May 2015.

[14] A. S. Rajan, S. Gobriel, C. Maciocco, K. B. Ramia, S. Kapury, A. Singhy, J. Ermanz, V. Gopalakrishnanz, and R. Janaz, "Understanding the Bottlenecks in Virtualizing Cellular Core Network Functions," in *The 21st IEEE International Workshop on Local and Metropolitan Area Networks*. IEEE, April 2015, pp. 1–6.

[15] Project Clearwater. [Online]. Available: http://www.projectclearwater.org/. Last accessed: June 2016.

[16] Q. Duan, "Modeling and Performance Analysis on Network Virtualization for Composite Network-Cloud Service Provisioning," in *2011 IEEE World Congress on Services*, July 2011, pp. 548–555.

[17] Netmanias. (2014, March) EMM Procedure 6. Handover without TAU - Part 2. X2 Handover. NMC Consulting Group. [Online]. Available: http://www.netmanias.com/en/?m=view&id=techdocs&no=6257&vm=pdf. Last accessed: May 2016.

[18] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An Analytical Model for Multi-tier Internet Services and Its Applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 291–302, June 2005.

[19] S. Sesia, I. Toufik, and M. Baker, *LTE-The UMTS Long Term Evolution: From Theory to Practice*. Wiley Online Library, 2009.

[20] I. Tsompanidis, A. H. Zahran, and C. J. Sreenan, "Mobile Network Traffic: A User Behaviour Model," in *Wireless and Mobile Networking Conference (WMNC), 2014 7th IFIP*, IEEE. IEEE, May 2014, pp. 1–8.

[21] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Analysis and modelling of YouTube traffic," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 4, pp. 360–377, June 2012.

[22] NGMN Alliance, "Radio Access Performance Evaluation Methodology," January 2008.

[23] G.-f. Zhao, Q. Shan, S. Xiao, and C. Xu, "Modeling Web Browsing on Mobile Internet," *IEEE Communications Letters*, vol. 15, no. 10, pp. 1081–1083, October 2011.

[24] HTTP Archive. Interesting stats. [Online]. Available: http://httparchive.org/interesting.php. Last accessed: July 2015.

[25] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Characterizing User Sessions on YouTube," in *Proc. SPIE 6818, Multimedia Computing and Networking 2008, 681806*, January 2008.

[26] T. D. Dang, B. Sonkoly, and S. Molnár, "Fractal analysis and modeling of VoIP traffic," in *Telecommunications Network Strategy and Planning Symposium. NETWORKS 2004, 11th International*. IEEE, June 2004, pp. 123–130.

[27] C. Anton-Haro and M. Dohler, *Machine-to-Machine (M2M) Communications: Architecture, Performance and Applications*. Elsevier, 2014.

[28] P. Jogalekar and M. Woodside, "Evaluating the scalability of distributed systems," *IEEE Transactions on parallel and distributed systems*, vol. 11, no. 6, pp. 589–603, June 2000.

[29] Amazon Web Services. Amazon EC2 Instances. [Online]. Available: http://aws.amazon.com/ec2/instance-types/. Last accessed: September 2015.

[30] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in *Modeling and Tools for Network Simulation*. Springer, 2010.

[31] B. Adler. (2010, March) White Paper - Load Balancing in the Cloud. RIGHTSCALE. [Online]. Available: https://www.yumpu.com/en/document/view/2675240/load-balancing-in-the-cloud-tools-tips-and-techniques. Last accessed: September 2015.

[32] Amazon Web Services Inc. (2015) Amazon Aurora Performance Assessment. [Online]. Available: http://d0.awsstatic.com/product-marketing/Aurora/RDS_Aurora_Performance_Assessment_Benchmarking_v1-2.pdf. Last accessed: September 2015.

[33] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed systems*, vol. 22, no. 6, pp. 931–945, 2011.

[34] Ericsson, "Characteristics. TECHNICAL PRODUCT DESCRIPTION," 43/221 02-AXB 250 05/8 Uen BK, April 2012.