Improving traffic forecasting for 5G core network scalability: A Machine Learning approach

Imad Alawe, Adlen Ksentini, Yassine Hadjadj-Aoul, Philippe Bertin

Improving traffic forecasting for 5G core network scalability: A Machine Learning approach

Imad ALAWE*, Adlen KSENTINI†, Yassine HADJADJ-AOUL*, Philippe BERTIN*
* IRT b<>com Rennes - France
name.surname@b-com.com
† EURECOM Sophia Antipolis - France
name.surname@eurecom.fr

Abstract

5G is expected to provide network connectivity to not only classical devices (i.e. tablets, smartphones, etc) but also to the Internet Of Things (IOT), which will drastically increase the traffic load carried over the network. 5G will mainly rely on Network Function Virtualization (NFV) and Software Defined Network (SDN) to build flexible and on-demand instances of functional networking entities, via Virtual Network Functions (VNF). Indeed, 3GPP is devising a new architecture for the core network, which replaces point to point interfaces used in 3G and 4G, by a producer/consumer-based communication among 5G core network functions, facilitating deployment over a virtual infrastructure. One big advantage of using VNF, is the possibility of dynamically scaling, depending on traffic load (i.e. instantiate new resources to VNF when the traffic load increases, and reduce the number of resources when the traffic load decreases). In this paper, we propose a novel mechanism to scale 5G core network resources by anticipating traffic load changes through forecasting via Machine Learning (ML) techniques. The traffic load forecast is achieved by using and training a Neural Network on a real dataset of traffic arrival in a mobile network. Two techniques were used and compared: (i) Recurrent Neural Network (RNN), more specifically Long Short Term Memory Cell (LSTM); and (ii) Deep Neural Network (DNN). Simulation results showed that the forecast-based scalability mechanism outperforms the threshold-based solutions, in terms of latency to react to traffic change, and delay to have new resources ready to be used by the VNF to react to traffic increase.

I. Introduction

Network virtualization has paved the way for 5G to meet high flexibility and agility expectations. Network virtualization allows running complex network functions (namely, Virtual Network Function - VNF) on top of a virtualized infrastructure, hosted at central or edge telco cloud. Chained together, the VNFs allow building flexible and on demand virtual networks. Such flexibility is highly required in 5G, where services need to be created on demand and for a given duration. Relying on Network virtualization, standardization groups, like ETSI and 3GPP, are multiplying efforts to build a new environment for 5G. ETSI, with the Network Function Virtualization (NFV) reference architecture[1], has created a reference model to orchestrate and manage VNF running on top of Virtual Infrastructure. Meanwhile, 3GPP is developing specifications to deploy 3GPP elements, including the core network (CN) functions and some parts of the Radio Access Network (RAN) functions [1][2], on top of a virtualized infrastructure. Particularly, the envisioned 5G core network has been devised with the aim to run the new functions on a virtualized environment. One option of the 5G core architecture is to replace classical 3GPP interfaces (i.e. point to point) by a bus-like communication to interconnect the core network elements, following the concept of web services.

VNF dimensioning is one of the main challenges of running core network elements in a virtualized environment, in terms of resources, which highly impacts performances [3]. Indeed, the virtualized core network elements should perform similarly to their hardware version; supporting the same Quality of Service (QoS): supported number of users, provided data rate, etc. To reach the same performance as the hardware version, over-provisioning the allocated resources for a VNF could be envisioned but it might be costly. Alternatively, adaptive provisioning of virtual resources could represent an interesting solution; where the VNF is over sized (by increasing the number of CPUs or by multiplying the number of VMs instances of the VNF) when traffic load is growing, and down sized when traffic is decreasing. Such solution is known as a scale-up/down (if the number of CPU is increased/decreased)

[1]ETSI, Network Functions Virtualisation (NFV); Architectural Framework, https://bit.ly/2GGPZpN, accessed in march 2018

or scale-out/in processes (if the number of VMs serving the same VNF is augmented/reduced), and is highly used in cloud computing. To scale VNF resources, a solution to take scaling decisions is needed. While in cloud computing the decision is mainly based on system level information, like consumed CPU or memory, obtained via the virtulization platform, mobile networks require service level information, like the number of users connected to the system or the traffic load, which indicate if the VNF is overloaded or not.

In this paper, we are interested in the scalability of a 5G core network key element, namely the Access and Mobility Management Function (AMF). The latter represents the bottleneck of the mobile network control plane, as it handles UEs attach requests. Accordingly, VMs hosting the AMF should be well dimensioned in order to avoid increasing the attach duration of UEs, and the percentage of rejected requests. The AMF should be over sized (scale-out), if the number of attach requests is high, and down sized (scale-in) if the number of such requests is low. In [4], we proposed a mechanism to scale the AMF using a threshold-based algorithm, which was derived using control theory. The proposed mechanism uses a threshold (the number of attach requests received by the AMF) to take the decision to scale-out or -in. However, due to the time needed to scale-out (i.e. the time needed for a new VM to be operational), the proposed solution suffers from delay to react to an increase in the number of attach requests; which leads to an increase of the attach procedure duration, and hence the rejection of some requests. To overcome this issue, we propose in this paper to use Machine Learning (ML) to forecast requests' arrivals, and hence anticipate the scale-out process; avoiding the rejection of requests and keeping the attach duration low.

The remainder of this paper is organized as follows. Section II introduces the necessary background to understand the paper, focusing on the 5G core network architecture and related works. Section III first introduces our motivations to ensure AMF scalability; then, the machine learning model and its usage to predict the user attach requests, and the proposed ML-based mechanism to scale-out and -in the AMF. Section IV presents the result of using the ML algorithm to predict the user attach based on the Telecom Italia dataset[2]. Finally, we conclude the paper in section V.

II. Background

A. 5G core network architecture

The 5G core network architecture has been devised with the aims: (i) to keep compliance with the concept of 3GPP communication reference interfaces (point-to-point); (ii) to evolve in near future to a service-oriented architecture, where the 5G core control network functions communicate via a common bus, through the concept of producers and consumers. Indeed, 3GPP SA2 group has defined two architectures, one based on the reference interfaces, and the other one following a service oriented architecture. In this paper, we will focus on the second one as the new components are expected to be run on top of a virtualized environment. For the first architecture readers may refer to [1]. Regarding the 5G core functions, some are similar to 4G EPC ones and some are completely newly designed. Notably, the access control and session management functions are combined in the Mobility Management Element (MME) of EPC, but separated in 5G core to better support fixed access and ensure scalability and flexibility. The most relevant network functions, as defined in 5G Core, are:

- Core Access and Mobility Management Function (AMF), which handles access control and mobility. If fixed access is involved, the mobility management will not be required in the AMF. It is in charge of handling attach requests of UEs from several Radio Access Networks (RAN); as being the entry point to the core network control plane it creates bottleneck issues when undersized;
- Session Management Function (SMF), which sets up and handles sessions according to the network's policy;
- Policy Control Function (PCF), which corresponds to the Policy and Charging Rules Function (PCRF) in LTE. It is worth noting that this function will integrate also a policy framework for network slicing;
- User Plane Function (UPF), which can be deployed, based on the service type, in several configurations and locations to be able to handle and forward users' data traffic;
- Unified Data Management (UDM), which is similar to the Home Subscribe Server (HSS) in EPS. However, it is envisioned that it integrates subscriber information for both fixed and mobile access in the New Generation (NG) Core;

[2]Telecom Italia, Telecom Italia Big Data Challenge: http://www.telecomitalia.com/tit/en/bigdatachallenge/contest.html, accessed in march 2018

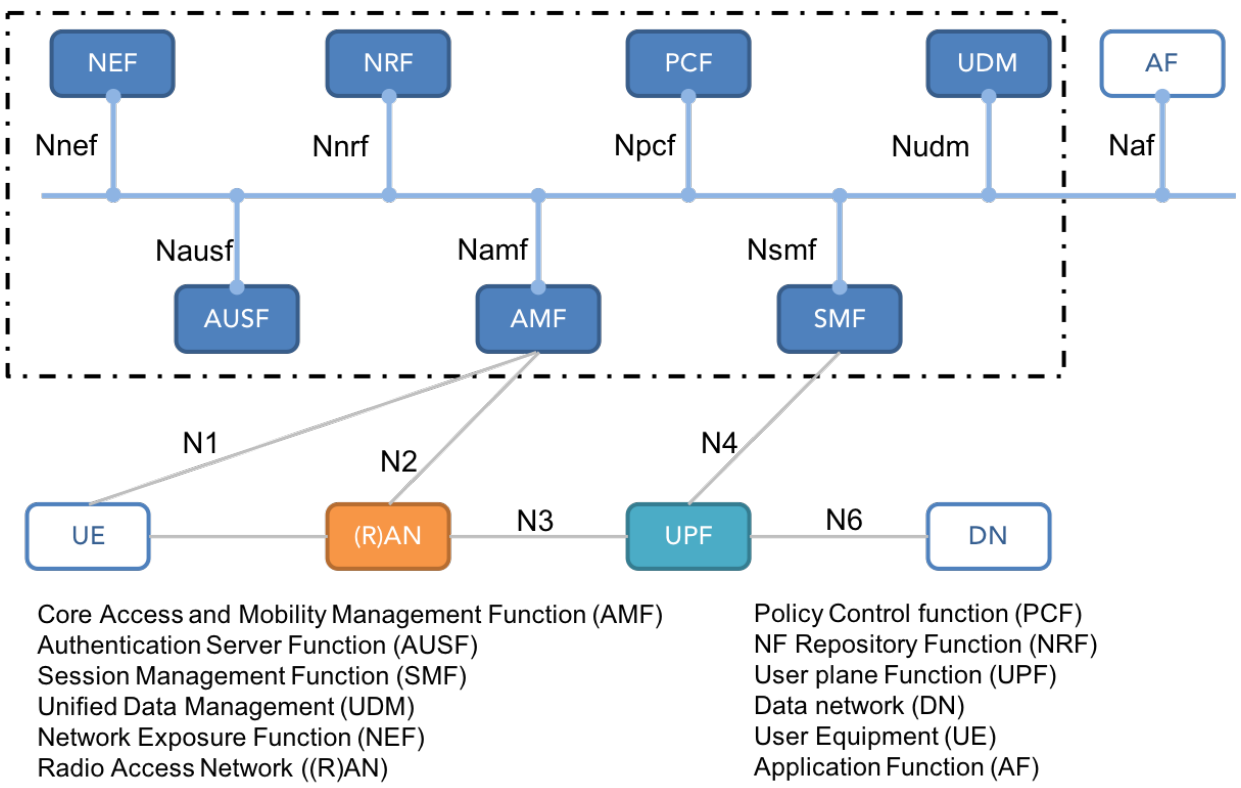


Fig. 1: Service oriented 5G Core network

- NF Repository Function (NRF), which is a new function. It provides registration and discovery functionality allowing the CN functions to discover each other and communicate via application programming interfaces (APIs).

In the following we shortly describe the AMF and SMF functions as this work focuses on the AMF scalability. For more details on the other functions reader may refer to [1]. The AMF ensures UE-based authentication, authorization, mobility management, etc. UEs, even using multiple access technologies, are connected to a single AMF, since the latter is agnostic to the access technology. The SMF is responsible for session management and IP address allocation to UEs. In addition, it selects and controls the UPF for data transfer. When a UE has multiple sessions, several SMFs may be allocated to each session aiming at managing the latter individually and possibly provide different functionalities per session. Based on operator policy or AF provided policy on packet flow, the PCF defines policies about mobility and session management, which will be enforced by the AMF and SMF.

Figure 1 illustrates the service oriented 5G core network architecture. Only the UPF is deployed on the legacy transport network infrastructure; the other core network functions handling the control plane are expected to run in more centralized telco cloud platform. While the communications with the UE, RAN and UPF are using the reference interfaces, the core control plane functions are connected together via a common communication bus. As stated earlier, the CN functions use the concept of producer/consumer, where a CN function (CNF) may register, using the NRF, for specific events provided by another CNF via an API. Thus, either 1:1 or 1:N communication is possible. For instance, the AMF service exposes information to the other CNFs regarding events and statistics related to mobility; or a PCF provides all the operations related to policy rules to other CNFs. The service-oriented 5G core architecture could be easily deployed in a virtualized environment (e.g. VM or container), wherein libraries of functions may be requested from a VNF catalog and composed into virtual Core Network on demand.

B. Related Work

As illustrated in Figure 1, the AMF is the entry point of the 5G core network; which makes it the system bottleneck. Thus, a special focus should be given to the dimensioning of the VNF(s) hosting the AMF to ensure

system scalability. Scalability in this case is more a software deployment issue rather than a hardware upgrade as most of the CNF are running as VNFs, on top of a virtualized infrastructure.

One of the most common scaling solutions consists in using static thresholds. In [5], [6] and [7] authors have proposed scalability based on thresholds. In the same spirit, we worked also on threshold based scalability in a previous attempt to manage AMF instances in [4]. The main idea is to deploy a new instance (scale-out) when the CPU usage (or other resources) exceeds a certain fixed "scale-out" threshold. Reciprocally, an instance is shut off (scale-in), when a considered threshold exceeds a fixed value. Static thresholds-based solutions are already deployed and widely used in commercial and open-source cloud solutions, like [8] and [9]. However, static thresholds-based approaches are not suitable for mobile networks, and have several limitations due to the fact that they react to events (i.e. reactive solution). Indeed, deploying a new instance of a VNF, when reacting to events, may take from one minute to a dozen of minutes depending on the data center architecture and the complexity of the VNF instance to scale; leading to delay the scaling-out process, hence increasing the attach duration of UE and rejection of some requests.

An alternative to static-threshold-based solutions is the usage of adaptive techniques, as proposed in [10]. In this work, the authors have proposed a mechanism that combines Q-Learning with Gaussian Processes-based system models, which allows to adapt to dynamic environments and improve the scaling policy before taking any action. The authors assume that their proposal reacts better than static threshold rules. However, it remains a reactive solution, and hence inherits the same weaknesses. Moreover, using Reinforcement learning, it may take a considerable time before starting to have the correct decisions, which is not acceptable in systems like 5G, due to the time constraint of certain type of applications. Note also the side effects associated to this type of approaches, consisting in the influence of the external environment on performance and therefore the risk of quality degradation.

Proactive solutions for scalability have been also proposed in the literature. The authors, in [11], suggested to pro-actively schedule resources for VNFs based on CPU usage forecasting from a historical dataset. Although this technique is interesting for its pro-activity, it does not offer adaptation in case of traffic pattern evolution and is not compliant with mobile networks requirements. Indeed, the proposed solution considers only system level information, like consumed CPU or memory, ignoring service level information. VNF scaling in mobile networks should take into account strict rules as defined in standards and therefore deploying only a new instance by a cloud orchestrator is not sufficient for scaling an end-to-end service. Consequently, mobile networks require service level information, like the number of connected users or traffic load when scaling-out or -in. Moreover, the parameters of the "offline" prediction equation, proposed by the authors, are determined statically and are completely independent from the considered dataset. This implies stability in the weights that are given for each day prediction, and thus a risk of divergence in the prediction with the slightest change of pattern. Concerning the "online" approach, the authors propose to use an average (exponential average) estimator. However following [12], when a pattern can be determined, this is clearly not the best solution.

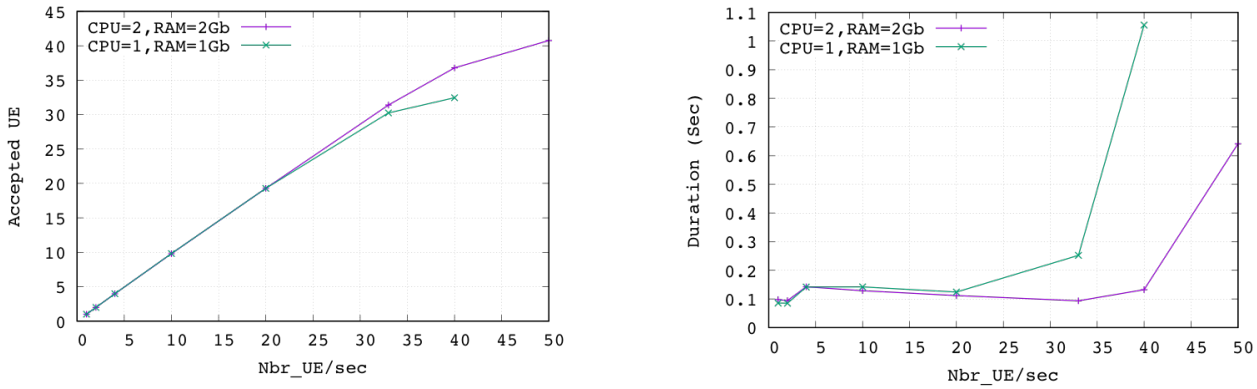
III. ML-BASED SCALABLE MECHANISM FOR 5G CORE NETWORK

A. Scalability issue of the AMF

To illustrate the need of an efficient scalability mechanism for the AMF, we conducted a set of experimentations using OpenAirInterface (OAI) solution[3]. OAI is a set of open source software tools that allows rapid prototyping of 5G mobile network; including the Radio Access Network and the Core Network functions. We used the emulated version of OAI, known as OAI simulator (OAISIM), which enables the emulation of a set of connected UEs and one eNodeB. The AMF was run on top of a VM, with two configurations: 1 CPU and 1 Gigabyte (Gb) of Memory (noted as first configuration); 2 CPU and 2 Gb of memory (noted as second configuration). It is important to note that we refer to AMF, even if we used the MME function of the OAI EPC. Indeed, our focus is mainly to compute the number of accepted UE and the time needed for a UE to attach to the network when varying the number of attach requests per second. Both metrics refer to the performance of authentication procedure, which is one of the common functions between the MME and AMF.

Figure 2(a) represents the accepted UE according to the number of UE attach requests per second. Clearly we observe that from a certain threshold, both configurations start rejecting UE attach. Obviously, stronger the

[3]http://www.openairinterface.org/, accessed in march 2018



(a) Accepted UE vs the number of UE attach request

(b) Attach duration vs the number of UE attach request

Fig. 2: Benchmarking result of the AMF

configuration of the VM, higher will be the threshold. In case of using the first configuration, the threshold is 20 UE per second, while for the other configuration is around 35 UE per second. We remark the same trend regarding the time needed by a UE to successfully attach to the network, which is presented in Figure 2(b). In this case, we observe that from the same threshold 20 UEs/sec, for the first configuration, and 40 UEs/sec second for the second one, the attach duration is exponentially increasing, which is violating the requirements specified by the 3GPP standard[4]. Obviously, there is a need to have a mechanism to solve this issue, which may consist on either: (i) adding more resources (CPU and memory) to the VM, i.e. scaling-up; or (ii) adding a new VM to the AMF function (with an internal load balancer), i.e. scaling-out.

B. Traffic forecasting: Deep learning-based approach

As explained above, there is a need of a mechanism to allow dynamic dimensioning of the AMF resources to avoid degrading the system QoS. The envisioned mechanism should be proactive to avoid the weakness of reactive solutions; hence requiring a traffic prediction solution. For this aim (i.e. traffic prediction), two Neural-networks-based techniques were envisioned and tested aiming at predicting the upcoming traffic load. The first one is based on Deep learning Neural Network (DNN), while the second one relies on Recurrent Neural Networks (RNN), more specifically on Long Short Term Memory cell (LSTM). In a deep neural network (DNN), a particular neuron can be activated by the contribution of several inputs at once, and therefore there is no real recognition of a traffic pattern. On the contrary, a recurrent neural network, and particularly an LSTM network, receives the inputs successively and through a memory and omission mechanism, it manages to memorize traffic patterns, as it has been demonstrated in several works [13].

The main procedures for traffic prediction using learning techniques are: getting the dataset, formatting the data, designing the neural network, training the neural network and then starting predicting (or testing your network). Each step is important in order to get the best accuracy of prediction compared to real values.

1) DataSet: The main key of prediction using deep learning, is the dataset. Nowadays, operators are able to collect those data easily as they are the owner of their infrastructure and have a clear idea of the evolution of the load over their network. In this work, we have used the dataset collected during the Big Data Challenge organized by Telecom Italia[2]. The data set is open source [14] and rich in information. It can be used to predict the evolution of the load in a core network.

The considered data set contains several kinds of data, like sms in/out, call in/out and the Call Detail Records (CDRs) with an interval of ten minutes for two months; including information on both control plane and data plane traffic. We consider in this work that at least 10% of the traffic is control plane traffic, and the remaining 90% is data plane traffic. Therefore, for the remainder of this paper, we will only use the control plane traffic (10% of the load in each period of ten minutes) for prediction.

[4]3GPP, Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3, https://bit.ly/2H0Tred, accessed in march 2018

2) Data Formatting: Once the dataset is collected and 10% of each period is extracted, the data should be formated in order to be injected in the neural network for the training phase. This part requires a high attention as the data format can affect the accuracy of prediction of the neural network. Formatting intelligently the data will allow the neural network to learn more in the training phase, and thus predicting with a high accuracy. However, adding lots of Key Performance Indicators (KPI) may lead to decreasing the prediction accuracy further, particularly for small datasets. Therefore a balance should be found between the dataset size and the KPI injected in the data formatting. In our case, we decided to format the dataset by classifying the load into 10 different classes, which can be interpreted as: class 1 is a load interval where (x) AMF instances are needed; Class 2 $(x + y)$ AMF are needed, and so on. Note that adding a much larger number of output classes would require a much larger dataset. Indeed, the addition of classes implies the increase of the number of parameters of the neural network which has an effect of requiring more data to have a significant precision.

3) Neural networks-based prediction: The first technique used for predicting the class of load of the next period is DNN. DNN is a neural network composed of three main layers: the Input layer, the middle layer(s) (hidden layer(s)) and finally the output layer. This type of neural network is also known as the Feed Forward Neural Network (FFNN). The data goes in the neural network from the input layer through the middle layer(s), and finally go out by the output layer.

The second technique tested for predicting the average load of the upcoming period of time is RNN, which is another type of neural network. Unlike DNN, RNN have loops. Those loops allow to inject previous events while predicting future events. It is a sort of memory block allowing to take decision depending on the previous event(s). Also, in RNN multiple kinds of cells are used today. In this paper we chose to rely on the LSTM cell. The advantage of the LSTM cell is to allow data patterns to be stored without degradation over time.

In order to compare the accuracy of the DNN and the RNN (LSTM), one DNN and one RNN (LSTM) were designed using the Tensor Flow library[5]. We divided and formatted the data into classes of average load. Then, we split the original dataset into two separate datasets. The first one contains 60% of the data, for training the two neural networks; while the remaining 40% are left to test the robustness of the two networks. Then both networks are trained with the 60% dataset and then asked to predict the remaining 40%. Figure 3 illustrates the prediction of both the DNN network and the RNN network. From Figure 3 we notice that RNN (LSTM) network performs better than DNN network. The RNN (LSTM) network achieves an accuracy of $\approx 90\%$, while the DDN network obtains $\approx 80\%$ of accuracy; hence the RNN is performing more than 10% better than DNN when predicting the class of average load of the upcoming period of time. The accuracy of the prediction is mainly due to the ability of LSTM networks to recognize a traffic pattern, unlike DNN networks. The accuracy of the DNN network can, however, be improved but this would require more neurons and more data, something that is not always available.

Based on these results, we will consider using RNN (LSTM) network for the AMF scalability mechanism.

C. Prediction-based scalability

Having described the prediction model, now we present the prediction-based scalability algorithm. Based on the ETSI NFV reference architecture, the scalability of a VNF is a decision taken by the NFV Orchestrator (NFVO), using the information provided by both the Virtual Infrastructure Manager (VIM) and the VNF Manager (VNFM). The former provides information on the VM execution environment (like CPU usage, Memory usage), while the latter provides information obtained from the Element Manager (EM) hosted in the VNF. The information provided by EM, represents the inside VNF service performances, like the number of attach requests or the size of the queue of the AMF, etc. In [4], we proposed that the EM of the AMF reports this information to the VNFM, hence the NFVO. The latter takes decisions to scale-in or -out using predefined thresholds. However, due the time needed to scale out (run a new VM), the proposed solution suffers from delay to react to the increase of UE attach; which leads to an increase in the time of the users' attach with some losses of requests.

To remedy to this problem, in this work, we will use the RNN to predict the needed number of AMFs that efficiently manages the traffic load, aiming at keeping the attach duration under an acceptable value (using the results shown in section III.A). The NFVO uses the RNN to predict the traffic (i.e. class of traffic), hence derives the optimal number of AMF to pro-actively deploy before the change of traffic load; anticipating any network load changes and removing the time needed to start a new VM instance, in case of scale-out process. Besides, in this

[5]TensorFlow, An open-source machine learning framework for everyone: https://www.tensorflow.org/, accessed in march 2018

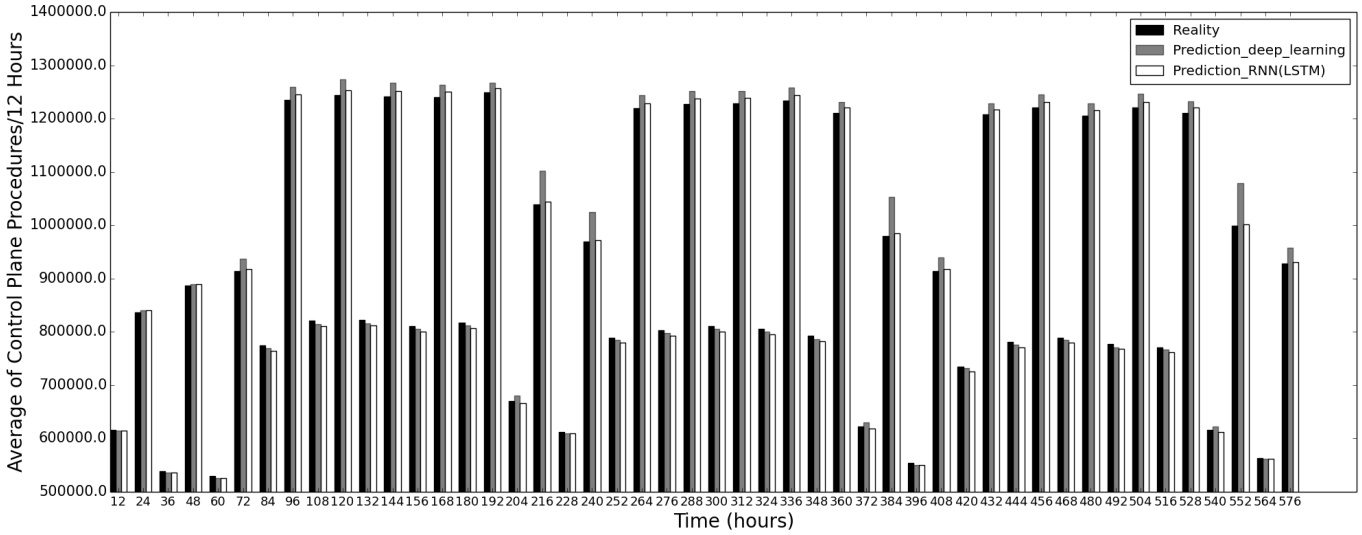


Fig. 3: Reality vs NN prediction

work there is no need to monitor the VNF performance via the VNFM, as the NFVO relies only on the trained RNN, which permits to reduce the communication load introduced by the exchanged messages between the EM and VNFM.

IV. Performance Evaluation

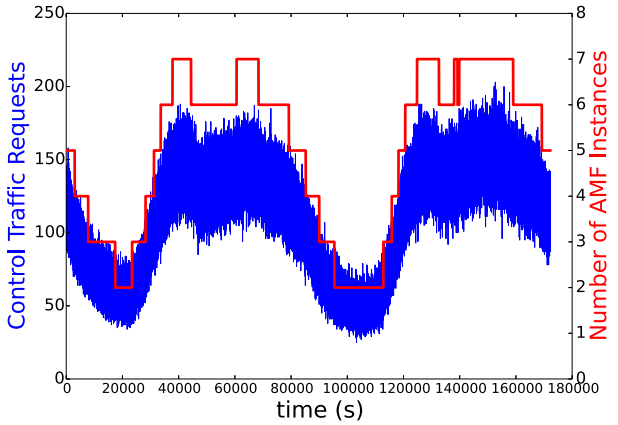
To evaluate the performances of the proposed solution and compare it with the threshold-based mechanism, we developed a discrete event simulator using Simpy[6]. The choice of Simpy was motivated by its efficiency when it comes to developing a simulator from scratch as well as its integration with other tools we have used, such as tensorflow. The simulator consists of:

- UE traffic generator, which allows to initiate UE attach requests according to the dataset
- an eNodeB that dispatches the arriving of control requests upon the deployed instances of AMF using a load balancing mechanism, if more than one AMF instance is used. It is based on the algorithm introduced in [4]
- each VNF (hosting an AMF component) can processes up to 20 UE attach requests per second.
- an NFVO that manages the scalability of the AMF.

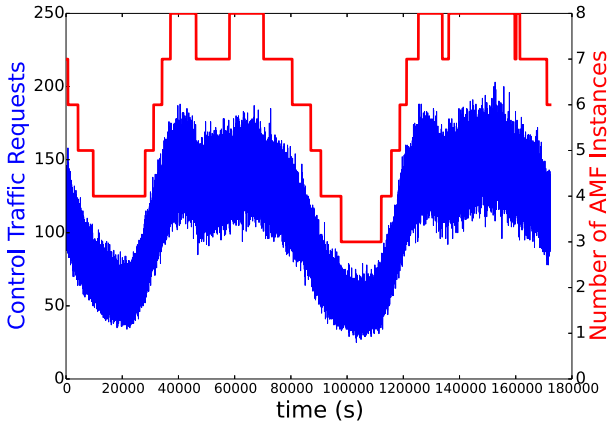
We launched several simulations to compare the performances of the proposed RNN-based scalability mechanism with the threshold-based solution. The former predicts the average load for the next period of time (10 minutes) and deploys the exact number of AMF instances needed to handle the upcoming load; while the latter launches a new instance (scale-out) when the overall arrival average load exceeds 80% of the overall system capacity (depending on the number of active AMF instances at the moment), and removes an instance (scale-in) when the overall average load decreases below 10% of the overall core network capacity. Both mechanisms were tested over 48 hours of arrivals data and with a deployment latency (i.e. time needed for the AMF instance to be operational) of 60 seconds. We chose this value according to [15], wherein the authors have studied the deployment latency for different VNF deployment scenarios on top of a private cloud infrastructure.

Figure 4 illustrates the attach requests rates vs the number of AMF instances deployed by both mechanisms; i.e. the threshold-based and the RNN-based. From Figure 4 we notice that both mechanisms are able to follow the arrival flow by deploying more AMF instances when the arrival load increases, and by reducing the number of AMF instances when the traffic load decreases. We remark that the RNN-based mechanism follows perfectly the arrival load. Furthermore, the RNN-based solution always anticipates the traffic load, and the number of AMF instances is always following the traffic trends. In contrast, the threshold-based mechanism takes more time to react. Indeed, we observe that in many cases, the two curves (Traffic load and number of AMF in Figure 4(a)) cross each other, meaning that the threshold-based solution is taking time to react; leading to a degraded network QoS.

[6]Team-SimPy, Discrete event simulation for python: https://simpy.readthedocs.io/en/latest/, accessed in march 2018



(a) Control traffic load vs the Number of AMF instances deployed by the threshold-based solution

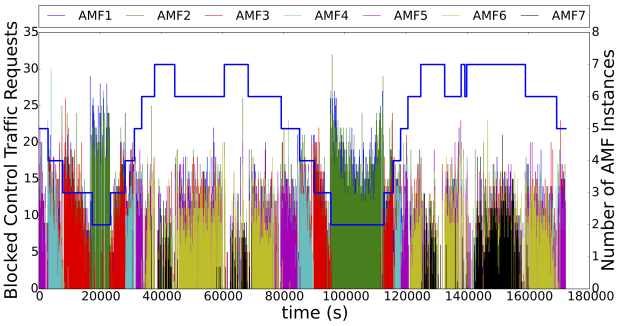


(b) Control traffic load vs the number of AMF instances deployed by the RNN-based solution

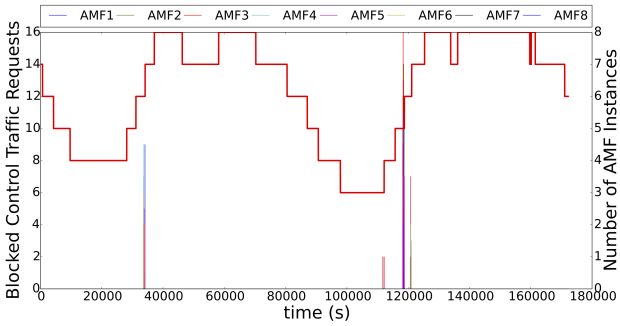
Fig. 4: Evolution of AMF instances number with control traffic Requests (a) threshold-based solution (b) RNN-based solution

This observation is confirmed in Figure 5, which illustrates the number of blocked attach requests in both solutions. Indeed, we remark that several requests are rejected in the case of the threshold-based solution compared to the RNN-based solution. The accurate prediction of the RNN-based solution allows to react in advance to the traffic changes by pro-actively instantiating the necessary VNF resources; avoiding overloading the AMF while waiting for the additional resource to be instantiated. Moreover, this result confirms that the RNN-based solution accurately estimates the needed number of AMF instances to manage a specific traffic load. For example when the load is maximal the RNN-based mechanism deploys up to 8 instances, while threshold-based solution deploys up to 7. Reciprocally, for the case when the arrival load is minimum, the threshold-based mechanism deploys at least 2 instances, while the RNN-based mechanism instantiates at least 3 AMF instances. Thus, the RNN calculates more accurately the number of instances needed to manage the load, allowing to stabilize the system and significantly decrease the number of rejected requests.

Figure 6 illustrates the Number of deployed AMF instances using the threshold-based mechanism vs the RNN-based mechanism. This Figure validates the trend seen in the precedent figures, where we clearly observe that the RNN-based mechanism is always in advance comparing to the threshold-based solution while reacting to the traffic change. Thanks to the traffic prediction, the deployment delays are removed, particularly for the scale-out process, which is not the case of the threshold-based mechanism that suffers from latency while deploying a new instance.



(a) Blocked control traffic requests vs the number of AMF instances deployed by the threshold-based solution



(b) Blocked control traffic requests vs the number of AMF instances deployed by the RNN-based solution

Fig. 5: Number of Blocked control traffic requests of AMF instances number with control traffic Requests (a) threshold-based solution (b) RNN-based solution

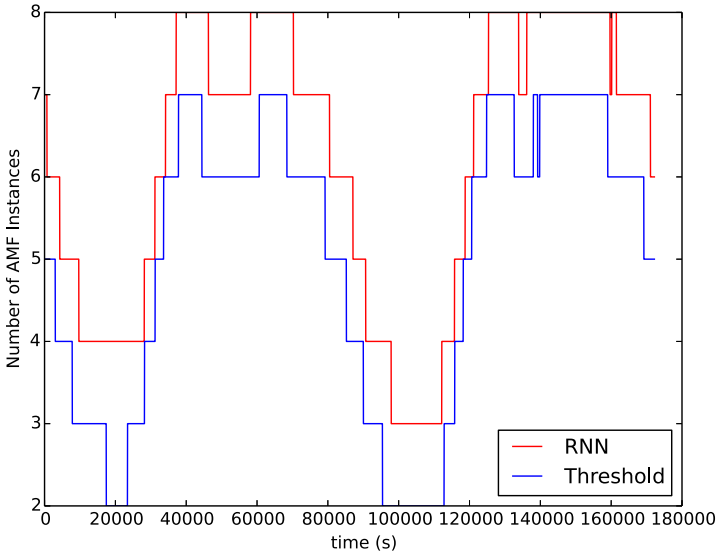


Fig. 6: Number of Deployed AMF instances
(Threshold Scaling vs RNN Scaling)

Indeed, as the RNN predicts the arrival load of the upcoming period, it deploys the AMF instances needed before the upcoming load in a way that the AMF instances are up and ready with the increase of the Load.

To sum up, the RNN(LSTM) technique performs better than the threshold technique as firstly it acts in a proactive way unlike the threshold technique. Secondly it allows to deploy accurately the exact needed number of AMF instances to absorb the upcoming load. Finally, it allows to remove the deployment latency by scaling out in advance the needed AMF instances. In that way, the mobile core is ready to absorb the upcoming load increase without suffering from overhead and extra latency for the control procedures.

V. Conclusion And Future Work

In this paper, we proposed a novel solution, based on machine learning, to scale-out and -in the AMF resources in a virtualized environment. Using a neural network, which has been trained on mobile network traffic dataset to forecast the user attach request rate, allows to predict the exact number of AMF instances needed to handle the upcoming user traffic. By being proactive, the proposed solution allows to avoid the deployment latency when scaling out resources. The latter may degrade the network performances, since the AMF is overloaded while waiting for the additional resource to be instantiated. Simulation results confirmed the efficiency of the RNN-based solution compared to a threshold-based solution. Future work aims to use the trained RNN to estimate the number of UPF (User plane 5G core network function) needed to handle the data plane traffic, in a virtualized environment.

References

[1] 3GPP, "System architecture for the 5g system." [Online]. Available: https://bit.ly/2GM9zgo
[2] ——, "Telecommunication management;study on management and orchestration of network slicing for next generation network." [Online]. Available: https://bit.ly/2qeZd1s
[3] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, P. Bertin, and A. Kerbellec, "On evaluating different trends for virtualized and sdn-ready mobile network," in *Cloud Networking (CloudNet), 2017 IEEE 6th International Conference on*. IEEE, 2017, pp. 1–6.
[4] I. Alawe, Y. Hadjadj-Aoul, A. Ksentini, P. Bertin, and D. Darche, "On the scalability of 5g core network: the amf case," in *IEEE Consumer Communications and Networking Conference (CCNC)*, 2018.
[5] S. Dutta, T. Taleb, and A. Ksentini, "Qoe-aware elasticity support in cloud-native 5g systems," in *ICC*. IEEE, 2016, pp. 1–6.
[6] G. A. Carella, M. Pauls, L. Grebe, and T. Magedanz, "An extensible autoscaling engine (ae) for software-based network functions," in *NFV-SDN*. IEEE, 2016, pp. 219–225.
[7] A. B. Alvi, T. Masood, and U. Mehboob, "Load based automatic scaling in virtual ip based multimedia subsystem," in *CCNC*. IEEE, 2017, pp. 665–670.
[8] Amazon, "Web services auto scaling." [Online]. Available: https://aws.amazon.com/autoscaling/

[9] A. CloudStack, "Open source cloud computing." [Online]. Available: http://cloudstack.apache.org/
[10] C. H. T. Arteaga, F. Rissoi, and O. M. C. Rendon, "An adaptive scaling mechanism for managing performance variations in network functions virtualization: A case study in an nfv-based epc," in *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 2017, pp. 1–7.
[11] A. Bilal, T. Tarik, A. Vajda, and B. Miloud, "Dynamic cloud resource scheduling in virtualized 5g mobile systems," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–6.
[12] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in *Chinese Association of Automation (YAC), Youth Academic Annual Conference of*. IEEE, 2016, pp. 324–328.
[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735
[14] T. Italia, "A multi-source dataset of urban life in the city of milan and the province of trentino dataverse." [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EGZHFV
[15] P. A. Frangoudis, L. Yala, and A. Ksentini, "Cdn-as-a-service provision over a telecom operators cloud," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 702–716, 2017.

VI. BIOGRAPHIES

Imad ALAWE is an engineering graduate Rennes school of engineer (ESIR) and a PhD student at TDF and IRT b<>com. He started at IRT b<>com in 2015. He first worked on Software Defined Networking (SDN) for optical networks in collaboration with Ekinops, than he joined TDF for the PhD. He works on the integration of SDN in the Evolved Mobile Core (EPC), with NFV and architecture evolutions toward 5G architecture.

Adlen Ksentini is a IEEE COMSOC distinguished lecturer. He obtained his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005. From 2006 to 2016, he was assistant professor at the University of Rennes 1. In March 2016, he joined the Communication Systems Department of EURECOM as an assistant professor. He has been working on Network Slicing and 5G in the context of two European projects on 5G, H2020 projects 5G!Pagoda and 5GTransformer.

Yassine HADJADJ-AOUL is working as an associate professor at University of Rennes (France), where he is a member of the IRISA Lab. and the INRIA Dionysos project-team. He received a Ph.D. degree in computer science from University of Versailles (France), in 2007. He was a post-doctoral fellow at University of Lille and University of Dublin (UCD) under the EUFP6 Marie Curie Action. His main research interests concern the fields of congestion control and QoS provisioning.

Philippe Bertin is a Senior Research Engineer at Orange Labs. He is managing research projects on networks architecture with b<>com Institute for Research and Technology. His research interests include the design of distributed and dynamic control and data planes for flexible and convergent 5G networks, leveraging on software defined networking and virtualization. He is graduated from Paris 6 University (MSc, 1993) and Rennes University (PhD on Computer Science, 2010).