ML-driven scaling of 5G Cloud-Native RANs

Akrit Mudvari, Nikos Makris, Leandros Tassiulas
Dept. of Electrical Engineering, *Yale University*,
New Haven, CT, USA
Email: akrit.mudvari@yale.edu, nikolaos.makris@yale.edu, leandros.tassiulas@yale.edu

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685874

***Abstract*—The evolution of the different network functions to a cloud-native configuration creates fertile ground for the efficient management and reconfiguration of the network. Through the wide application of softwarization and virtualization, cloud-native approaches can extend even to the RAN, that has been dominated by monolithic non-configurable hardware equipment in the past generations of mobile network access. As such, a cloud-native deployment can cover the end-to-end 5G network architecture, from the Core Network to the base stations, with the respective services benefiting from several advanced features, such as automatic scaling of the deployed functions based on monitored metrics. Through the application of Machine Learning, the evolution of the metrics can be predicted and thus the respective functions can be pro-actively scaled. In this work, we use an end-to-end real-world cloud-native deployment of a 5G network, and deal with two different types of scaling, applied at three different parts of the network: vertical scaling for the base station, and horizontal scaling for control and user plane functions of the core network. We use a real-world dataset for replicating traffic over our setup and closely monitor the evolution of metrics from different parts of the network. By applying Machine Learning methods, we accurately predict the future network load and use it to decide on the pro-active allocation of resources for the RAN and the Core Network.**

***Index Terms*—5G network, cloud-native, auto-scaling, Machine Learning, Kubernetes, OpenAirInterface**

I. Introduction

5G networks provide a radical shift from the traditional cellular networking paradigm, by introducing several concepts that add up to the flexibility, management and re-configuration of the network. Through the disaggregation of the cellular infrastructure in multiple levels, part of the networking stack can run at the network edge as a Virtual Network Function (VNF), allowing for flexible instantiation/tear-down of base stations in a single-click fashion [1]. The 5G Core Network (5GCN) Service Based Architecture (SBA) allows the off-the-shelf deployment of several 5G network components in a Cloud-native manner [2], while such functionality is further extending to the RAN with the disaggregation of the control and user plane functions of the base stations, through efforts such as the O-RAN alliance [3]. This disaggregation enables the flexible managementof the RAN network on the fly, by dynamically configuring the resource allocation per VNF.

Dynamic adaptation of the RAN resources can prove to be very beneficial in-terms of overall network efficiency, and can assist in the dynamic adaptation of the network, based on the workload that it is expected to cope with. For example, during night-time we expect lower use of the cellular networks compared to rush-hours, and this is reflected to the amount of computational resources needed for various processes in the RAN or the Core Network. Examples of such resources can include computational resources for LDPC en/decoding at the base station or resources for the User Plane Function (UPF) relaying traffic to the Data Network (DN). At the same time, the load in the network can be inferred from different RAN-level metrics, or by monitoring the entire micro-service that implements the service and extracting metrics such as traffic load that each Core Network component exchanges/relays to the DN. Metrics such as the Packet Data Convergence Protocol (PDCP) bytes or the MAC level SDUs for the Uplink (UL) or Downlink (DL) can reveal the load under which the cell is placed, and hence closed control loops at the base station level can be employed for re-organizing the cell, towards coping with the demand. Such re-organization can include adding computational resources to the cell, re-configuring the wireless network parameters, or re-allocating slices for the UEs.

Although such metrics provide insights for the current network utilization, predicting the evolution of a metric in the near future can provide substantial benefits for the self-organization of the network. Machine Learning (ML) application can help towards that goal through the estimation of specific features that combined can offer the required metric with high accuracy. In this work, we focus on a cloud-native cellular network, with the core network and the RAN being realized using micro-services using the OpenAirInterface [4] platform. By exploiting the Kubernetes framework and the advanced features it provides, we aim towards achieving the following goals:

- To extract metrics revealing the true network utilization for either the RAN or the Core Network.
- To apply ML for predicting future resource requirements, based on the observed traffic patterns.
- To define a proactive scaling mechanism for the RAN and the Core Network components, using the vertical and horizontal scaling that the framework provides.

The rest of the paper is organized as follows: in Section II we provide some relevant literature. In Section III we provide the system model and the related ML processes that we use. In Section IV we evaluate our framework, and in Section V we conclude our work and present our future directions.

II. Related Work

Network virtualization has enabled several key features that add up to the flexibility for managing the network and

This work was supported by the National Science Foundation under Grant CNS 2112562 and the Office of Naval Research under Grant N00014-19-1-2566.

allocating the resources in a meaningful manner. Machine Learning (ML) is a powerful tool that can assist in the allocation process, through effective predictions for the evolution of monitored network statistics. Authors in [5] employ ML for the allocation of resources in the cellular network, in a vehicular environment. They compare three different models, namely a Convolutional Neural Network (CNN), a Deep Neural Network (DNN) and a Long-Short Term Memory (LSTM) model, used to predict traffic and establish flows in an SDN controller managing the network, and evaluate it in terms of training cost and prediction accuracy. In [6], authors analyze the different learning models (supervised, unsupervised and reinforcement learning) that can be used for efficiently allocating slices over cellular infrastructure. In [7], the authors employ an unsupervised learning model for clustering nodes in a fog-networking domain towards minimizing latency.

Empowered by the flexibility of cloud-native applications and managing frameworks that enable automatic scaling of the deployed services, authors in [8] develop an ML classifier which learns from past VNF scaling decisions and spatial network traffic load to proactively scale the network. The authors compare their solution against other ML-based classifiers for training cost and accuracy, and present quantitative results regarding the leasing cost savings of their solutions. In [9], the authors extend their solution for service chains, and formulate the problem as a negotiation game between the network operator and the infrastructure tenants who participate in the scaling decision. Auto-scaling applications for the telecom cloud cellular network is proposed in [10]. Authors use a Markov Decision Process for Reinforcement Learning, and determine the policy in auto-scaling based on the traffic workload. Through extensive experiments, the authors prove that their policy-based solution outperforms target-based auto-scaling decisions, while ensuring QoS in the network.

Although the previous works focus on scaling the deployed services of a 5G network, none of them apply the process to the actual services that provide network access. Authors in [11] aim to fulfil this gap, by introducing a scaling algorithm based on Control Theory for the Access and Mobility Function (AMF) of the 5G Core Network (5G-CN). Their solution is optimizing only the Control Plane (CP) part of the network. They present an algorithm that efficiently scales-out the AMF function of the network when traffic peaks are observed, and traffic is subsequently balanced among the replicas, achieving better efficiency and less blocked users under high load. In [12] their solution is further extended with a ML approach using a LSTM model for predicting peaks in traffic, and proactively scale the AMF for absorbing future incoming traffic. Both solutions are evaluated using the 4G equivalent of the AMF, the Mobility Management Entity (MME).

Extending to the User Plane (UP) of the core network, authors in [13] analyze the different ML techniques that are applicable for scaling the User Plane Function (UPF) that provide data access in the 5G-CN architecture. In [14], an algorithm is proposed for auto-scaling the UPF equivalent in the 4G network architecture, namely the Serving Gateway (SGW) and Packet Data Network Gateway (PGW), towards ensuring constant bit-rate to each of the allocated network slices. Authors in [15] develop a ML approach enabling O-RAN managed networks to decide in near real-time their allocation based on network metrics from the base station.

In this work, we extend existing state-of-the-art by employing a LSTM model to predict traffic patterns observed in the Core Network, towards determining scaling decisions for both CP and UP functions. The scaling decisions for the Core Network are horizontal, enabling the replication of the service to multiple replicas that serve concurrently the attached UEs. Moving beyond existing state-of-the-art, and based on the fact that the base station is running in a virtualized environment, we propose scaling the base station as well towards ensuring seamless network service to bulks of UEs using the network. The scaling of the base station is vertical, meaning that more computational resources are added to the VNF, as under load several bottlenecks in scheduling/decoding exist [16].

In the following section we present our system architecture, and provide details for the ML framework that we use to predict the traffic patterns in the network. The network traffic is replicated from the open source dataset of Telecom Italia [17], which provides traffic data from the 4G networks in the city of Milan and the Province of Trentino.

III. System Model

In this section we present the system architecture and the ML approach used for predicting the metrics that define the scaling of the network.

A. System Setup

Towards creating a cloud-native RAN, we employ the OpenAirInterface (OAI) platform [4]. OAI is an all-software based implementation of the 4G and 5G RAN and Core Network, implementing the network over commodity equipment using a Software Defined Radio (SDR) front-end. Since the 5G-CN implementation is at the time of writing not mature yet, we focus on the LTE implementation of the network. The CN is realized using the Control and User Plane Separation (CUPS) architecture, allowing each function to run as a separate micro-service. The OAI implementation of the CUPS architecture is providing the Home Subscriber Service (HSS), the Mobility Management Entity (MME), a Service/Packet GW for CP (SPGW-C), a Service/Packet GW for UP (SPGW-U), and a Cassandra database for holding the subscriptions. We focus on scaling the RAN eNB function, the MME for CP traffic, and the SPGW-U for UP operations. The cases of scaling are mapped to the 5G implementation without changing any of the monitored metrics as follows: instead of the eNB we scale the gNB operation, instead of the MME we scale the AMF, and instead of the SPGW-U we scale the UPF.

All the different components have been containerized as docker micro-services, including the RAN function (eNB) which requires only access to an SDR-based front-end (either network or USB based). In order to enable management, monitoring and automatic-scaling of the functions, we use

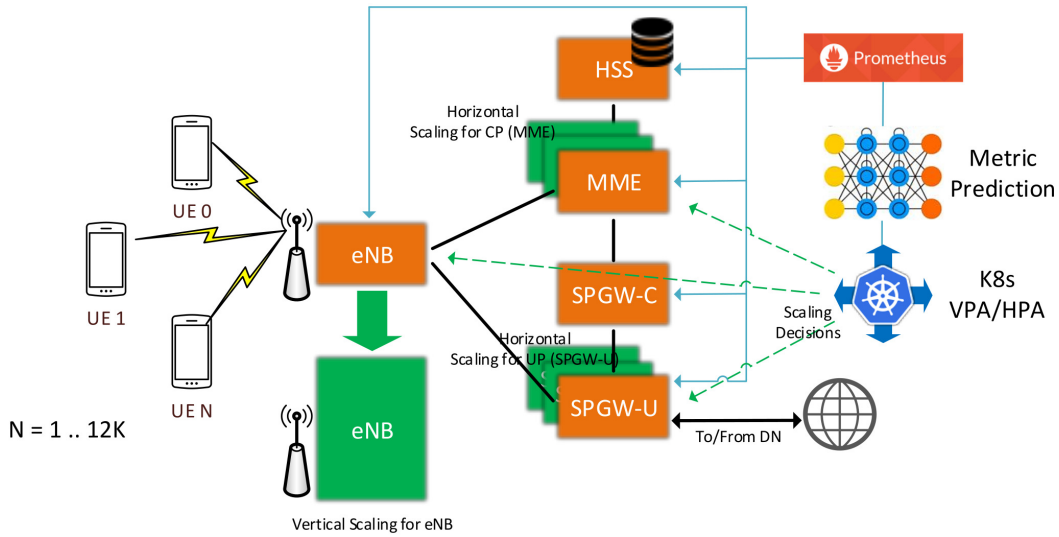


Fig. 1: System architecture for the under-study cloud native network: telemetry for the collected metrics is provided by Prometheus, and data are managed with our RNN; decisions on the number of replicas/resource allocation (in green color) are applied through the Kubernetes HPA and VPA.

the Kubernetes (K8s) framework [18] to deploy the network services over our infrastructure. K8s allows the effortless deployment of cloud-native services, allowing their automatic scaling using user-defined triggering metric targets. Scaling employed in this work is horizontal, realized by the Horizontal Pod Autoscaler (HPA) which spawns replicas of the service and redirects traffic to the pool of replicas, or vertical, realized by the Vertical Pod Autoscaler (VPA) which assigns more/less resources to a container triggered by a target metric.

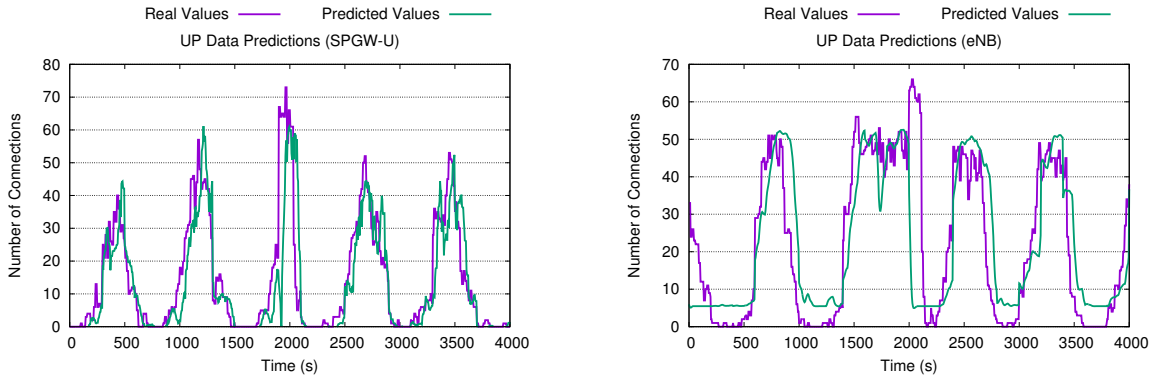
Towards extending the monitored metrics that drive our decisions for scaling, we deploy the Prometheus Operator [19] within our K8s cluster. Prometheus is a monitoring stack that integrates with the APIs of K8s allowing global services such as the HPA/VPA to take advantage of extended metrics from the deployed services. In order to get more precise measurements about the scheduling decisions and the load of the cell on the radio side, we also deploy and integrate in the monitoring solution the FlexRAN [20] platform. FlexRAN queries the eNB and can retrieve real-time statistics from the OSI layer-2 protocols of the eNB. In our setup, it was deployed as a separate container, and integrated to the Prometheus monitored metrics by exposing data through a REST interface, queried by a K8s service monitor. Figure 1 shows the components for our system-level cloud-native RAN architecture.

B. Machine Learning

For determining the predictability of different variables on the demand for VNFs, we analyze the collected metrics and employ a data driven approach for making predictions. We identify certain important VNFs in the cloud-native RAN paradigm that could benefit from the proactive scaling out/in/up/down processes. For instance, pressure in decoding received network traffic in eNodeBs could be relieved with vertical scaling up of the containers, while energy could be saved by keeping resources unexploited during a lack of workload. Similarly, the number of VNFs could be horizontally scaled out and in to match the demands and save resources and energy in the process. For the MME case, a bottleneck in the attach process due to resource-congestion [11] can cause drops in service for other users attaching to the cell in parallel. Such behavior could be mitigated by scaling the service accordingly by looking at specific traffic patterns. Towards identifying key metrics that reflect the load of the network inside specific functions of the cloud native RAN, we generated traffic based on an existing dataset [17] and monitored the affected metrics. For our case, we focused on the following variables which are representative of the pressure conditions for each VNF:

1) For the eNB, the UP bytes sent over the PDCP protocol.
2) For the SPGW-U, the UP traffic that is sent over the S1-U interface.
3) For the MME, the CP traffic that is exchanged over the S1-MME interface.

These metrics are constantly monitored and their values are predicted through our ML model towards identifying possible pressure in one of the components. The solution is receiving a time-series of the metrics, and estimating the number of clients that are sending traffic/requesting service from the network. For capturing the relationship between various network metrics and the demand for the VNFs, Deep Learning grants us the potential to capture the underlying relationships. This allows us to scale the resources to match the number of anticipated load, and in the process save energy/resources. It is evident that the network traffic from the past would suggest the number of expected connections at present, with recent statistics being a stronger predictor. The rate of fluctuations in traffic volume could also be a source of prediction. Hence, to consider the temporal nature of UP as well as CP traffic, we employed a Recurrent Neural Network (RNN) [21]. We implemented a Long Short-Term Memory (LSTM) [22] RNN, since they improve on the capability of RNNs to consider temporal memory with an added ability to overcome issues such as the vanishing



(a) Core Network UP predictions (SPGW-U)

(b) RAN UP Predictions (eNB)

Fig. 2: ML model evaluation at different parts of the network (Core/RAN) and types of traffic (CP/UP)

gradient problem. Prediction for different network functions followed a similar NN model, and here we describe the model applied in the SPGW-U case in further detail. In this case, the input is a vector (time-sequence) of relevant network traffic statistics (rate of UP traffic flow) through the SPGW-U VNFs, collected over a certain period of time at certain intervals, i.e. from $t-k$ to $t-l$ where $k > l > 0$. The output of this NN is the prediction of number of connections accessing the SPGW-Us at certain time t. Hence, the prediction is for future traffic demands based on past traffic data. During the learning phase, the output from this NN is compared against the real collected output for supervised learning. The connection statistics and the network traffic generated by the system at different times are used as training and testing data (with 80% of the data allocated for training and 20% for testing), with the test case showing how well the chosen network statistics can predict the demand for the corresponding network functions.

An LSTM structure consisting of 8 hidden layers, 2 stacked layers, followed by a leaky ReLu before the output was found to be adequate for learning, with test data showing that over-fitting was not a significant issue and accuracy was high. L2 regularization was employed to prevent over fitting, with Adam [23] as the gradient decent method and Mean Squared Error (MSE) as the loss function. To determine the ability of the neural network to make accurate predictions, Mean Absolute Error (MAE) between predicted and the actual number of connections was used. With other parameters remaining the same, the number of hidden layers and stacked layers were reduced to 4 and 2 respectively for predicting the CP load at the MME side, while no such changes were made for predicting the eNB demands. These design conclusions were verified through experimentation over different configurations.

As our results show in Section IV, the MME is quickly saturated when receiving high loads of traffic, as it processes the network attach requests in a serial manner. Therefore, scaling the MME can help to parallelize the attach process for more clients, and relieve the load from a single MME.

IV. Evaluation

For the evaluation of the developed functionality, we use an indoor testbed located in the Yale University premises. We use a single node setup, equipped with a USRP B210 SDR, where the core K8s framework is configured and the cloud-native RAN is instantiated. A second node is used with a LTE dongle as UE, that can attach to the deployed RAN. On the node that is attaching to the network, we use an open source dataset provided by Telecom Italia [17], which contains patterns of users requesting service from the network, spanning an entire week. The dataset was replicated as new connections for data service that are requested from the UE. Each new connection is classified in a random manner as low/medium/high network traffic; this is mapped to the traffic that is injected to the DL channel towards the UE. The three classes equal to 70/25/5 % of the traffic connections for low/medium/high traffic load categories respectively. Each of the UE connections triggers a DL UDP traffic stream from a docker located in the data network after the SPGW-U. The sum of the traffic going over the network is configured to reach the maximum capacity of the eNB cell at any time. This equals to approx. 35 Mbps, as the eNB is configured to operate in FDD band 7, with 10MHz channel bandwidth in a single antenna configuration. For all the experiment cases, we use the ML model to predict the number of connections that are active over the cloud-native network, based on the metrics described in Section III.

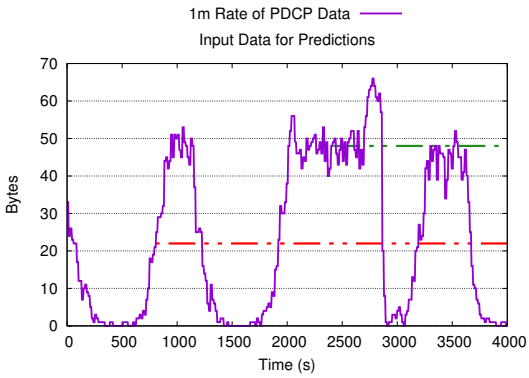


Fig. 3: Data used for eNB predictions; red and green dotted lines represent two cases where vertical scaling shall take place for accommodating the extra load in the network

The first objective was to record relevant UP traffic statistics at the core network (i.e. through Serving Gateway/PDN

Gateway (SPGW)) and figuring how they can be used to determine the number of active connections that the core is supposed to serve. The LSTM was used to predict how the changes in UP traffic can track the demand for the services. Based on the aforementioned dataset, randomized connections were generated, and for these connections, the system implementation allowed us to obtain the network traffic information. In an offline setting, the neural networks were trained in a similar manner. Test data were generated for the evaluation in the same system and manner. Initially, we consider the capability of our method to analyze the traffic statistics in the SPGW-U to accurately predict the number of connections that are expected to be served. Figure 2a shows how our system predicts the number of connections. As shown, the ML model is able to closely track the actual number of connections, which allows the system to accurately make scaling decisions. To quantify and compare the accuracy of this method, it was compared against a moving average approach, but the accuracy of our method was found to be significantly better. To measure the accuracy, MAE was calculated for the test data. We took the error calculated by the best training method as the baseline/tolerance, and measured the extra error generated on test case by different methods. For our approach this value was 0.08. In the implementation of the moving average method, the value was 1.8 with the same test data. This implies that in about an hour of test data, about 1238 instances would be predicted more efficiently. Since the inference time is about 2 msecs on a setup with CUDA and a Quadro M2200 graphics card, real time inference/decision making is not a concern for our model.

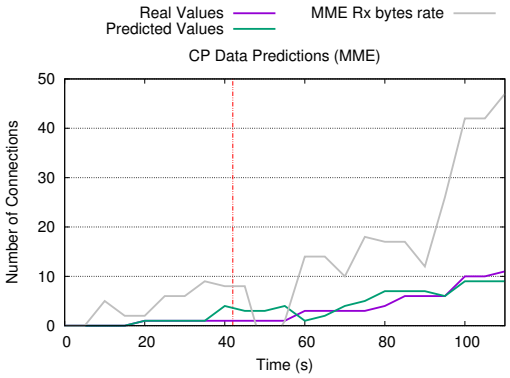


Fig. 4: Core Network CP predictions (MME): With a red dotted line we denote the point triggering the autoscaling process

Using an identical set of parameters, we measure the expected number of connections on the eNB side using the information reported on the PDCP data exchange. As seen in Figure 2b, the predictions are very close to the actual number of connections. Based on the measured connections, and thus the stress that the eNB is getting, the eNB can re-provision itself to either use more resources (e.g. CPU/MEM) or change its settings completely towards addressing the demand (e.g. enable a MIMO mode/enable carrier aggregation etc.). Similar to the case of SPGW-U, the values stayed at 0.09 for our approach and 2.3 for the moving average approach. In cases

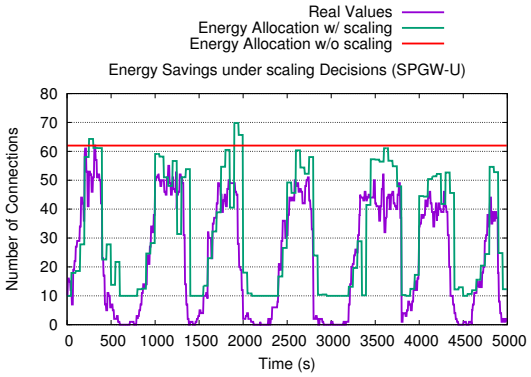


Fig. 5: Energy efficiency evaluation of the auto-scaling mechanism for the SPGW-U

that the demand is low, the network can reduce resources for cost savings. Towards determining the exact demand, we combine the estimated number of connections, together with the actual PDCP data that represent the UP eNB pressure, towards determining in a lower level the UEs that request most traffic from the network. Based on this information, we determine the exact scaling of the network. Figure 3 shows the actual data that was used for determining the number of connections active over the eNB. Based on predefined thresholds, we trigger the vertical scaling of the eNB for accommodating the load. This approach can extend for other uses, including the allocation of slices within the network, based on the demand that we infer from the monitored metrics. Figure 3 illustrates two different points where the scaling process can happen; with a red line, we denote a scaling decision for allocating more channel bandwidth (i.e. re-configuring the cell bandwidth from 10MHz to 20MHz), and with a green line a change in the antenna configuration, from SISO to 2x2 MIMO.

Regarding the CP experiments, since the number of UEs present in the dataset can reach up to 12K per each cell for the entire week period, we use a simulated RAN and UE provided by OAI, as OAISIM. OAISIM is integrated with the rest of the architecture, and substitutes the eNB and UEs in Figure 1 with a single container. The dataset is replicating traffic as new UEs entering the system, and exchanging S1AP messages with the MME. This approach is the closest possible to replicating CP traffic over a real network, as prior works in literature consider part of the traffic as CP, which is not realistic. Message exchanges that stress the MME happen only during the network attach process, and are only a small fraction of the traffic that is exchanged during a connection.

As shown in Figure 4, the model was able to predict the demand based on the CP data. Nevertheless, since the MME is working in a different manner for the CP traffic than the SPGW-U for UP, it can be observed that after a certain point the MME is saturated; this causes subsequent requests to be delayed for processing, causing delays in the attach process for the new UEs. With a gray line we denote the traffic pattern that was used for predicting the number of connections. It represents a scaled down version of the rate of received bytes, as monitored by Prometheus, averaged over the last minute.

As we see, beyond the point of approx. 42 secs, the rate is constantly high, meaning that the UEs more aggressively send attach requests. Nonetheless, the number of actual (and therefore predicted) UEs is not changing, denoting that there is a bottleneck in the attach process. In order to alleviate this issue, our framework timely identifies it and instructs the cluster to scale out the deployment with more MME replicas.

An important implication of making a real-time prediction of the demand for different VNFs is to prevent over-provisioning of the resources during deployment, while also ensuring that the services are adequately served and delays are avoided when a service needs to be scaled. Using the aforementioned predictions, we can optimize the number of hardware units dedicated to specific VNFs, allowing the system to free up the resources for energy savings or other applications that do not have stringent delay requirements. We therefore present some indicative results for the case of the SPGW-U VNF in our system. The prediction of demand is used to update the allocation for a certain period of time; in Figure 5, the allocation based on prediction closely tracks the real demand as opposed to an arbitrary allocation such as near the expected maximum (shown with a red-colored line). The illustrated results show that the allocation of resources is reduced by approx. 50.3% for the specific interval. We have further examined the resource allocation efficiency for different settings, subject to factors such as "how much to err on the side of over provisioning to avoid congestion?" and "how often to update the allocation of resources?", but most of those decisions in our experiments allowed an efficiency within the range of 37% and 58% savings.

V. Conclusion

In this work, we presented a real cloud-native RAN deployment using off-the-shelf tools for monitoring its operation. The cloud-native deployment extended to the actual RAN, by enabling the execution of the cell providing access as a microservice. Leveraging on the advanced functionalities for scaling the deployed VNFs, we introduced two types of scaling for the network: horizontal for the core network components, and vertical for the RAN. Exploiting telemetry tools for different parts of the deployed network, we developed and evaluated a ML model for predicting the impact of different metrics in the network, based on the load under which it is placed. Our results showed clear benefits for the developed framework as the ML solution is able to predict the number of active connections with high accuracy over the network, which reflects to pro-active scaling of different components for the User Plane or Control Plane. For Control Plane, efficient scaling results in more users being able to attach to the network concurrently, thus alleviating any blockages that a UE might experience when swarms of users request network service. For the User Plane, we illustrated that efficient scaling of the Core Network components can lead to high energy gains, while for the RAN components, vertical scaling can assist in accommodating the incoming demand. In the future, we foresee to extend the ML scheme towards identifying the applications that run over the network, and subsequently allocating the respective slices in an end-to-end manner.

References

[1] C.-Y. Chang, N. Nikaein, R. Knopp, T. Spyropoulos, and S. S. Kumar, "FlexCRAN: A flexible functional split framework over ethernet fronthaul in Cloud-RAN," in *2017 IEEE ICC*. IEEE, 2017, pp. 1–7.

[2] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5G evolution: A view on 5G cellular technology beyond 3GPP release 15," *IEEE Access*, vol. 7, pp. 127 639–127 651, 2019.

[3] L. Gavrilovska, V. Rakovic, and D. Denkovski, "From Cloud RAN to Open RAN," *Wireless Personal Communications*, pp. 1–17, 2020.

[4] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM Computer Communication Review*, vol. 44, 2014.

[5] S. Khan Tayyaba, H. A. Khattak, A. Almogren, M. A. Shah, I. Ud Din, I. Alkhalifa, and M. Guizani, "5G Vehicular Network Resource Management for Improving Radio Access Through Machine Learning," *IEEE Access*, vol. 8, pp. 6792–6800, 2020.

[6] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and T. Miyazawa, "Consideration on automation of 5G network slicing with machine learning," in *2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*. IEEE, 2018, pp. 1–8.

[7] E. Balevi and R. D. Gitlin, "Unsupervised machine learning in 5G networks for low latency communications," in *2017 IEEE 36th IPCCC*, 2017, pp. 1–2.

[8] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee, "Auto-Scaling VNFs Using Machine Learning to Improve QoS and Reduce Cost," in *2018 IEEE ICC*, 2018, pp. 1–6.

[9] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee, "Auto-scaling network service chains using machine learning and negotiation game," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1322–1336, 2020.

[10] P. Tang, F. Li, W. Zhou, W. Hu, and L. Yang, "Efficient auto-scaling approach in the telco cloud using self-learning algorithm," in *2015 IEEE GLOBECOM*. IEEE, 2015, pp. 1–6.

[11] I. Alawe, Y. Hadjadj-Aoul, A. Ksentini, P. Bertin, and D. Darche, "On the scalability of 5G core network: The AMF case," in *2018 15th IEEE CCNC*, 2018, pp. 1–6.

[12] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, "Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach," *IEEE Network*, vol. 32, no. 6, pp. 42–49, 2018.

[13] Y. Fu, S. Wang, C.-X. Wang, X. Hong, and S. McLaughlin, "Artificial intelligence to manage network traffic of 5G wireless networks," *IEEE Network*, vol. 32, no. 6, pp. 58–64, 2018.

[14] T. V. K. Buyakar, A. K. Rangisetti, A. A. Franklin, and B. R. Tamma, "Auto scaling of data plane VNFs in 5G networks," in *2017 13th CNSM*, 2017, pp. 1–4.

[15] H. Lee, J. Cha, D. Kwon, M. Jeong, and I. Park, "Hosting AI/ML Workflows on O-RAN RIC Platform," in *2020 IEEE Globecom Workshops*. IEEE, 2020, pp. 1–6.

[16] N. Budhdev, M. C. Chan, and T. Mitra, "Poster: IsoRAN: Isolation and Scaling for 5G RAN via User-Level Data Plane Virtualization," in *2020 IFIP Networking Conference*. IEEE, 2020, pp. 634–636.

[17] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific data*, vol. 2, no. 1, pp. 1–15, 2015.

[18] D. Bernstein, "Containers and cloud: From LXC to docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[19] J. Turnbull, *Monitoring with Prometheus*. Turnbull Press, 2018.

[20] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proceedings of the 12th CoNEXT*, 2016, pp. 427–441.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Tech. rep. ICS 8504. San Diego, California: Institute for Cognitive Science, University of California*, Sept, 1985.

[22] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 12 2014.