On Auto-scaling and Load Balancing for User-plane Gateways in a Softwarized 5G Network

Van-Giang Nguyen*†, Karl-Johan Grinnemo*, Javid Taheri*,
Johan Forsman†, Thang Le Duc†, and Anna Brunstrom*
*Department of Computer Science, Karlstad University, Sweden
{giang.nguyen, karl-johan.grinnemo, javid.taheri, anna.brunstrom}@kau.se
†TietoEVRY Product Development Services, Sweden
{johan.forsman, thang.leduc}@tietoevry.com

***Abstract*—In the fifth generation (5G) mobile networks, the number of user-plane gateways has increased, and, in contrast to previous generations they can be deployed in a decentralized way and auto-scaled independently from their control-plane functions. Moreover, the performance of the user-plane gateways can be boosted with the adoption of advanced acceleration techniques such as Vector Packet Processing (VPP). However, the increased number of user-plane gateways has also made load balancing a necessity, something we find has so far received little attention. Moreover, the introduction of VPP poses a challenge to the design of the auto-scaling of user-plane gateways. In this paper, we address these two challenges by proposing a novel performance indicator for making better auto-scaling decisions, and by proposing three new dynamic load-balancing algorithms for the user plane of a VPP-based, softwarized 5G network. The novel performance indicator is estimated based on the VPP vector rate and is used as a threshold for the auto-scaling process. The dynamic load-balancing algorithms take into account the number of bearers allocated for each user-plane gateway and their VPP vector rate. We validate and evaluate our proposed solution in a 5G testbed. Our experiment results show that the scaling helps to reduce the packet latency for the user-plane traffic, and that our proposed load-balancing algorithms can give a better distribution of traffic load as compared to traditional static algorithms.**

***Index Terms*—5G Core, Load Balancing, User Plane, Vector Packet Processing, VPP, Container, Auto-Scaling**

I. Introduction

Currently, the roll-out of commercial 5G networks [1] is happening worldwide. The rise of 5G brings with it a wider range of use cases with widely varying performance requirements, such as enhanced mobile broadband (eMBB), massive machine type communication (mMTC), and ultra-reliable and low-latency communication (URLLC). The realization of 5G is supported and simplified by advanced technologies such as Software Defined Networking (SDN) and Network Function Virtualization (NFV) [2]. With SDN and NFV, the architecture of traditional mobile networks, e.g., 4G, and their mobile packet core networks have been significantly changed. The control and user planes are now completely separated according to the principles of SDN. NFV allows the mobile core network functions to be virtualized instances, running in a cloud environment instead of running on proprietary hardware. On one hand, this enables a more flexible deployment closer to the end-users; scalability with an increased number of user-plane gateways; and, not least, cost-efficiency. On the other hand, advanced performance acceleration techniques such as Data Plane Development Kit (DPDK) [3] and FD.io VPP [4] will help bridge the performance gap between a software-based network function and its hardware-based counterpart.

However, the increased number of user-plane gateways makes the balancing of traffic among a set of user-plane gateways a necessity. In addition, employing high-speed packet processing techniques in the implementation of user-plane gateways faces a challenge in being auto-scalable. Although there have been several proposals for improving the performance of the user plane through a scalable design, only a few works are focusing on the load balancing aspect. Regarding the scaling of user-plane gateways, most existing proposals follow a threshold-based approach, using CPU utilization as a scaling indicator [5]. However, the use of CPU utilization as a scaling indicator in a DPDK/VPP environment is not as straightforward as it might seem. Particularly, the polling mode of DPDK/VPP causes a high CPU utilization, even at times when no packets are being processed.

In this paper, we focus on improving the auto-scaling and load-balancing functions of user-plane gateways in a softwarized, VPP-equipped 5G Non-Standalone (NSA) network. In our experiments, we saw a correlation between the VPP vector rate or VPP rate, i.e., the number of packets that can be processed simultaneously, and the traffic load experienced by a user-plane gateway. To this end, we propose to use the VPP rate as an indicator for triggering the auto-scaling process. We also use this indicator as a load indicator in two of our proposed load-balancing algorithms: the VPP Rate Based (VRB) and the Dynamic Weighted Least Bearer Based (DWLBB) algorithms. In the third load-balancing algorithm, we propose the Least Bearer Based (LBB) algorithm, in which the load-balancing is done on the basis of the number of active bearers [1] allocated to each user-plane gateways.

The work was supported by the 5GENESIS project funded by the European union.

[1]In 5G NSA, a bearer refers to a logical end-to-end user plane connectivity between the UE and an access point name through the Packet Gateway (PGW). In 5G Standalone (SA), Quality of Service (QoS) flows are used in stead of bearers. However, the LBB algorithm can still be applicable in which the number of allocated bearers are replaced by the number of created QoS flows.

We implemented our auto-scaling mechanism and dynamic load-balancing algorithms in a 5G testbed at TietoEVRY, Sweden [6]. This testbed provides an implementation of a softwarized 5G NSA network. It can be deployed in an automated manner in various kinds of virtualized infrastructures. We deployed the testbed on a single physical host, with all of its network functions containerized in Docker[2]. For performance comparison, we also implemented two static load-balancing algorithms: Round-Robin (RR) and Weighted Round-Robin (WRR). Our experiments show that a scaling out of user-plane gateways will help prevent the user plane from an overload situation, thus reducing the packet latency in comparison to running a single user-plane gateway. We also show in our experiments that the three proposed load-balancing algorithms are able to better distribute the load among the user-plane gateways as compared to the RR and the WRR algorithms.

The rest of the paper is organized as follows. Section II provides a brief overview of the 5G architecture and its two major deployment options. The section also introduces high-speed, packet-processing technologies in user space. Section III surveys related works on auto-scaling and load balancing in the user plane of a mobile packet core. The design and implementation of our proposed auto-scaling indicator and dynamic load-balancing algorithms are discussed in Section IV. Section V presents the TietoEVRY testbed and how our experiment was setup on this testbed. Section VI summarizes the performance evaluation of our proposed solution. Finally, Section VII concludes the paper and provides directions for future work.

II. Background

This section provides a brief background of 5G mobile networks and its two major ways of being deployed: NSA and SA. The section also provides a short introduction to the two key user-space, high performance packet processing techniques employed in the 5G user plane.

A. 5G Deployments

Figure 1 depicts the two major 5G deployment options: the NSA and the Standalone (SA) deployments [7]. As follows, the radio-access network comprises three main parts, each of which takes care of different portions of the radio protocol stack: a Radio Unit (RU), a Distributed Unit (DU), and a Central Unit (CU). It also follows that the CU in turn consists of two sub units: a CU-C unit in the control plane and a CU-U unit in the user plane. The mobile packet core network in the NSA deployment can be implemented on top of a virtualized 4G Evolved Packet Core (EPC) or as part of its enhanced version, a virtualized 4G EPC Control and User Plane Separation (CUPS) [8] architecture. In the CUPS architecture, user-plane gateways refer to Serving Gateway User plane (SGW-U) and PGW User plane (PGW-U) or a combined version of SGW-U and PGW-U. The main control

[2]https://docker.com

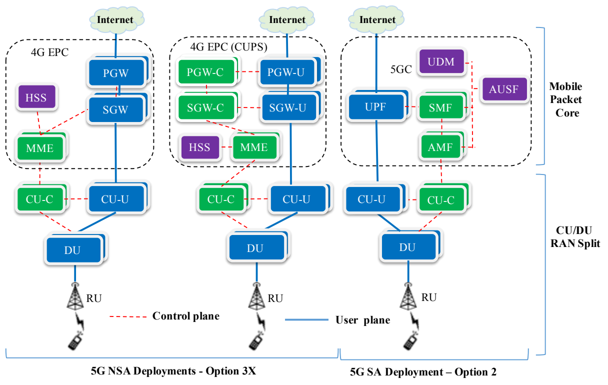

Fig. 1: 5G Deployments: NSA and SA Options

plane elements are the Mobility Management Entity (MME), the Home Subscribe Server (HSS), and the control plane of SGW and PGW, i.e., the SGW-C/PGW-C. As shown in Figure 1, a 5G SA deployment differs quite a bit from a 5G NSA deployment. The control plane introduces new elements including: the Access and Mobility Management Function (AMF), the Session Management Function (SMF), the Unified Data Repository (UDR), and the Authentication Server Function (AUSF). The AMF incorporates part of the functionality of the MME element in EPC and EPC CUPS, and is mainly responsible for connection management, authentication, and mobility management in these architectures. The SMF has part of the MME and the SGW-C/PGW-C functionality and responsible for managing session contexts and the selection of user-plane gateways. User-plane gateways in 5G SA refers to the User-Plane Function (UPF). The UPF contains functionalities of SGW-U/PGW-U such as tunnel encapsulation and decapsulation. The general packet radio service (GPRS) tunneling protocol (GTP) [9] is used by the control plane to set up tunnels in the user plane to forward the user data packets going between radio access networks and external packet data networks such as the Internet. Rajan et al. [10] have identified that the SGW function is a bottleneck in the user plane of a virtualized 4G EPC network.

B. High Performance Packet Processing

With the critical requirements of a low latency and/or prompt service, both the former 4G networks and the currently deployed 5G networks demand extremely fast packet processing. Following the adoption of the CUPS architecture, the service latency in 5G networks primarily depends on the packet processing and forwarding speed in the user plane. To this end, DPDK [3] has been widely employed in the user plane of 5G. DPDK is essentially a set of user-space libraries and drivers for fast packet processing in data plane applications. More specifically, DPDK is improving the packet throughput through two key techniques: (i) a fast-path between a NIC and an application that does away with any latencies incurred by the kernel-/user-space barrier and by internal handling in the kernel; and (ii) a poll-mode driver that enables a CPU to actively poll NICs for data packets instead of relying on

CPU interrupts. To further boost the packet processing of user-plane data packets in 5G, the so-called VPP [4] technology has recently found its use in the implementation of user-plane gateways. VPP is built on top of DPDK to leverage its fast packet capturing and forwarding capabilities. It employs an advanced technique to process a vector of packets, i.e., multiple packets at a time, and in so doing improves the packet-processing capacity.

III. Related Works

In this section, studies related to scalable design, auto-scaling and load balancing of mobile-packet core gateways are surveyed.

Scalable user-plane design: An et al. [11] and Jerez et al. [12] shared a common goal of re-architecturing the PGW in a scalable way. They structured the PGW as a set of OpenFlow switches and dedicated entities for tunneling processing. Rajesh et al. [13] proposed SCOPE, a system that efficiently manages the compute and network resources, and which provides flexible and efficient mechanisms to configure the SGW-U/PGW-U virtual machines across multiple data centers. Kumar et al. [14] investigated the offloading of the functions of an evolved packet gateway, i.e., a combined SGW-U/PGW-U, to a P4-capable programmable switch ASICs.

User Plane Load Balancing: The pooling concept has been used in the 4G EPC to provide redundancy and improved scalability, e.g., the pooling of SGW functions. The scheduling of SGWs within the pool is done by the MME function, and is often done statically using an approach based on the Domain Name System (DNS) [15]. With SDN and NFV, virtualized user-plane gateways are easy to scale and flexible to deploy in contrast to non-virtualized ones. Lee et al. [16] focused on the load balancing of the SGW function in an SDN-based EPC architecture, where SGWs are implemented as simple OpenFlow switches. The load balancing is similar to the one used in wired SDN networks. Tan et al. [17] focused on the load balancing of UPFs in a 5G Service Based Architecture (SBA) by proposing a load-balancing algorithm named Reliable Intelligent Routing Mechanism (RIRM) that considers the shortest path, link latency, and node loading in its load-balancing decisions.

User Plane Auto-Scaling: Luong et al. [18] proposed a solution for auto-scaling of SGW functions in a container-based environment using Kubernetes[3]. Auto-scaling is done when the CPU consumption increases and passes a threshold. Arteaga et al. [19] focused on auto-scaling of a virtualized EPC (vEPC). The CPU utilization of the servers in a vEPC is used for triggering the scaling action. Buyakar et al. [20] proposed a bitrate-aware auto-scaling (BAAS) algorithm that maintains a precise User Equipment (UE) bit rate requirement in the network slices without over-provisioning of user-plane resources. The throughput threshold for triggering the scaling is set to about 90% of the maximum capacity. Kumar et al. [21] presented a study on statically scaling a UPF in the public cloud. They defined a threshold for packet processing performance per core beyond which they propose to scale out or scale in units of compute and networking capacity.

[3]https://kubernetes.io/

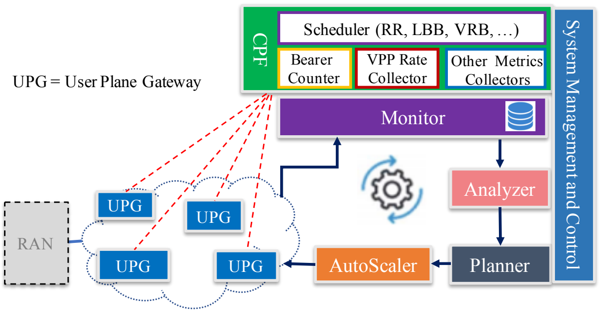

Fig. 2: Proposed System Architecture

In contrast to previous works, we propose a novel auto-scaling mechanism combined with several dynamic load-balancing algorithms for a VPP-based user plane in a softwarized 5G core. To our knowledge, this is the first study that suggests to make auto-scaling and load-balancing decisions on the basis of the VPP vector rate.

IV. System Design and Implementation

First, this section describes the components of our proposed system architecture. Next, it introduces our novel auto-scaling mechanism and how auto-scaling could be done on the basis of the VPP rate. Finally, the section proposes three dynamic and adaptive load-balancing algorithms.

A. System Components

Auto-scaling is often implemented according to the well-known Monitor-Analyze-Plan-Execute (MAPE) model [5]. As its name suggests, the MAPE model has four main phases: (1) monitoring, (2) analysis, (3) planning, and (4) execution. We also follow this model to design our proposed system architecture. The overall design of our proposed system architecture is shown in Fig. 2. In this figure, the four main components corresponding to the four phases of MAPE are the Monitor, the Analyzer, the Planner, and the AutoScaler. The Monitor is responsible for periodically collecting performance metrics from the system, i.e., the user-plane gateway cluster. The performance metrics can be gathered at system level, e.g., CPU utilization, memory, and at the application level, e.g., number of requests, VPP rate. The collected information is analyzed by the Analyzer. Different analysis techniques such as data smoothing techniques can be used in this component to identify the trend of the collected data, and to provide references to the Planner. The Planner will make plans for when and how the system should change, e.g., scaling-in or scaling-out, based on information from the Analyzer. In our case, we implement a threshold-based auto-scaling on the basis of the VPP rate information. The AutoScaler is the last component in this loop. It executes the scaling plan by sending commands to the user-plane gateway cluster, e.g., by calling the Docker API to create a new user-plane gateway container.

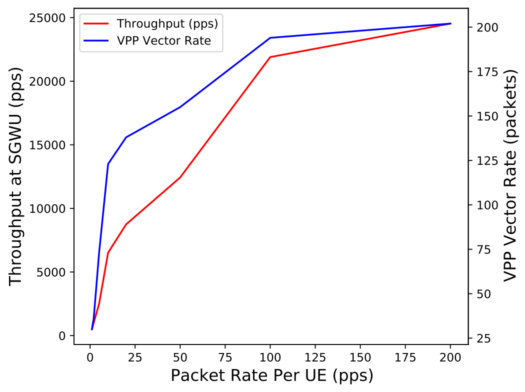

Fig. 3: Throughput versus VPP Vector Rate

The number of user-plane gateways in the cluster will be changed according to the auto-scaling mechanism, so a load balancing algorithm is needed to distribute the load within the cluster. We introduce into the Control Plane Function (CPF) entity, i.e., the MME or the SMF, three new modules: a Bearer Counter, a VPP Rate Collector and a Scheduler. The Bearer Counter is responsible for counting the number of active bearers for each user-plane gateway. The VPP Rate Collector is responsible for the collection of the VPP rates of each user-plane gateway through the Monitor. It should be noted that, our system can be easily extended to include other metric collectors. The Scheduler redirects the incoming traffic according to one of the proposed algorithms, which are described in Section IV-C. In our system, a System Management and Control component is proposed to configure and manage the other components in the system. For example, it can configure the monitoring interval in the Scheduler.

B. VPP Rate Aware Auto-Scaling (VRAAS)

As discussed before, doing auto-scaling for high-performance packet processing applications such as VPP-based applications is challenging since it is not possible to use the most commonly used performance indicator: the CPU utilization. To this end, in this paper, we propose the use of the VPP vector rate as a performance indicator and a trigger for an auto-scaling decision. We conducted an experiment to find out the correlation between the input traffic, the perceived throughput, and the VPP vector rate. The experiment uses the TietEVRY 5G testbed [6] (see Section V for a detailed description) running a single chain of a UESIM container, a CU container, a SGWU container, a PGWU container, and a CNSIM container. The total number of UEs is 500 with a packet rate per UE of 1, 2, 5, 10, 20, 50, 100, and 200 packets per second (pps). We capture the throughput received at the SGWU container and the vector rate of the VPP instance running inside that container. Figure 3 shows the correlation between the throughput and the VPP vector rate. It shows that the throughput increases as the VPP vector rate increases. On the basis of this study, we suggest the use of the VPP vector rate for auto-scaling of VPP-based user-plane gateways in a softwarized 5G network. We call our auto-scaling algorithm VPP Rate-Aware Auto-Scaling or VRAAS. For the current version of this algorithm, we adopt a threshold-based auto-scaling approach, where the threshold is estimated based on the VPP Rate information captured from the Monitor component as shown in Figure 2.

C. Load Balancing Algorithms

Load balancing is one of the most important operational tasks in any system, and aims to efficiently distribute incoming network traffic across a group of back-end servers. A good load-balancing algorithm helps to increase the overall system performance and prevents the system from an overload situation. The load-balancing decision of the user-plane gateways is made by the CPF. We have proposed several dynamic load balancing algorithms, Least-Bearer Based (LBB), VPP Rate Based (VRB), and Dynamic Weighted Least Bearer Based (DWLBB). We further employ two static algorithms: Round-Robin (RR) and Weighted Round-Robin (WRR), for performance comparison. The RR algorithm schedules the incoming traffic in a cyclic fashion, while the WRR algorithm schedules incoming traffic proportionally to a pre-configured weight assigned to each user-plane gateway. The remainder of this section gives a detailed description of our proposed dynamic load-balancing algorithms.

1) The LBB Algorithm: In mobile networks, an established connection between a UE and its serving node in an external packet data network, e.g., the Internet, is often referred to as a bearer. A single UE can have multiple bearers, e.g., when it uses different Internet services at the same time. The bearer information is monitored by user-plane gateways. A large number of bearers allocated at a user-plane gateway means that it possibly needs to handle a big volume of incoming traffic. With this in mind, we propose the LBB algorithm which allows the CPF function to select its corresponding user-plane gateways based on the number of bearers allocated to each of them. There is a counter for each user-plane gateway to keep track of the total number of bearers that is currently allocated. Whenever a new bearer is created for a UE at a user-plane gateway, its counter will be increased by one, and whenever a bearer is released, the counter value will be decreased by one. The counter value is then used as an estimate for the load of the user-plane gateway. The scheduler will select the user-plane gateway that has the lowest counter value to serve new incoming UE requests.

2) The VRB Algorithm: As discussed in the previous section, capturing load of a VPP-equipped network function is a challenge, and the use of CPU utilization is not applicable. Instead, we propose the use of the VPP average vector rate per node as a load indicator. It is essentially the number of packets that a VPP instance can process in a batch, and it shows how much the VPP instance is busy: A larger number means the VPP instance is more busy. In our design, each user-plane gateway will periodically query this information from its VPP instance, and then report the obtained VPP rate to the CPF function, i.e., the MME or the SMF function. Upon

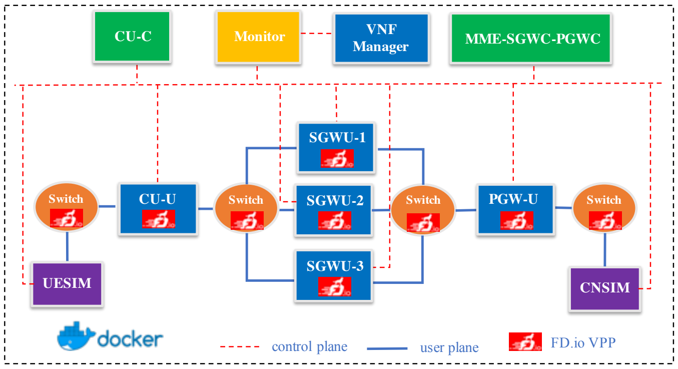

Fig. 4: Experiment Setup using TietoEVRY's 5G Testbed

reception of the VPP load-report messages from the user-plane gateways, the CPF will record the VPP rate. The scheduler will then select the user-plane gateway which has the lowest value of the VPP rate to serve the upcoming UE request.

3) The DWLBB Algorithm: The DWLBB algorithm is an improved version of the LBB algorithm where each user-plane gateway is assigned a dynamic weight using the VPP vector rate information described above. The Scheduler makes its decision on the basis of the ratio of the current number of bearers allocated to each user-plane gateway and its corresponding dynamic weight. The user-plane gateway with the lowest ratio is selected.

V. Experimental Testbed

This section presents our experimental environment. We provide a description of the main components of the TietoEVRY 5G testbed [6] and how this testbed is used in our experiment setup.

A. *TietoEVRY 5G Testbed*

To validate the efficiency as well as the applicability of our load-balancing and auto-scaling solutions, we carry out a set of experiments on the proposed algorithms within a 5G testbed developed by TietoEVRY, Sweden. The testbed includes almost all the key virtualized functional components to support the emulation of network function chains: 5G core and RAN virtualized components, network services/applications, UEs as well as their simulated service usage. Virtualized components include the CU-U, the CU-C, the SGW-Us, the PGW-U, and the MME which are implemented and delivered as full-fledged functions in both the user and control planes. UEs and network services are simulated through two simulators, the UESIM and the CNSIM, respectively. All these functional components are developed and deployed in a cluster using Docker as illustrated in Fig. 4. Every user-plane network function leverages DPDK and VPP in order to achieve high-performance packet processing and forwarding. To obtain a high-speed data transmission between all the functional components, VPP-enabled vSwitches are provided to establish connectivity between the components in the testbed. We extended the testbed with the implementation of our proposed auto-scaling and three proposed dynamic load balancing algorithms.

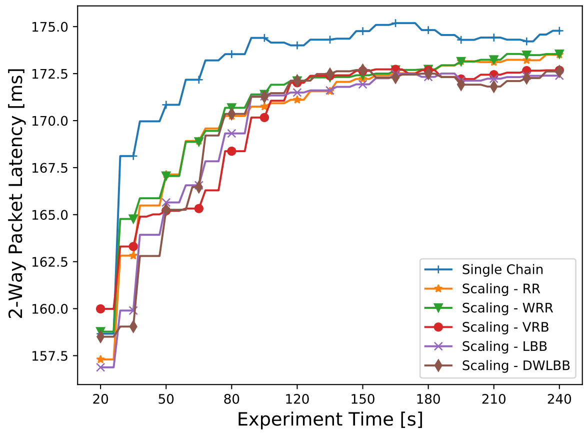

Fig. 5: Packet Latency Comparison

B. *Experiment Setup*

We run the testbed described above with a Docker Engine 19.0.13 running on a server that has 16 CPU cores and 32 GB of RAM. The UESIM, the CNSIM, and the CUU containers run on two CPU cores, while the PGW container runs on a single CPU core. Due to a limitation in the traffic generator used in the UESIM and CNSIM containers, it is not possible to generate enough load to saturate the system. To overcome that, we use a Docker command to limit the CPU resource for the SGWU-1, SGWU-2, and SGWU-3 to be 0.5, 1.0, and 1.5 CPUs, respectively. This CPU configuration allows us to assign static weights of 1, 2, and 3 to the SGWU-1, SGWU-2, and SGWU-3, respectively, when running experiments with the WRR load balancing algorithm. The VPP version 20.09 is used in our testbed.

VI. Experimental Evaluation

In this section, we provide a description of our experiment scenario and the experiment results. As the scaling-in feature has not yet been thoroughly tested on our testbed, we conduct experiments with the scaling-out scenario only. It means that at the current stage, we are only able to add new SGWUs into the running testbed.

A. *Experiment Scenario*

The goals of our experiment scenario are to (i) illustrate the benefit of running multiple SGWUs compared to running a single SGWU with the support of the auto-scaling, and to (ii) evaluate how well traffic gets balanced using our three proposed dynamic load-balancing algorithms, the LBB, the VRB, and the DWLBB algorithms, in comparison to the static RR and WRR algorithms. Due to the capacity of the Docker server, we can conduct experiments with a maximum of three SGWU containers. However, the same methodology can be easily used for a larger number of SGWU containers in a more powerful system. Our experiment scenario is divided into three periods, and in each period we increase the number of UEs attached to the network. A total of 6000 UEs spanning over

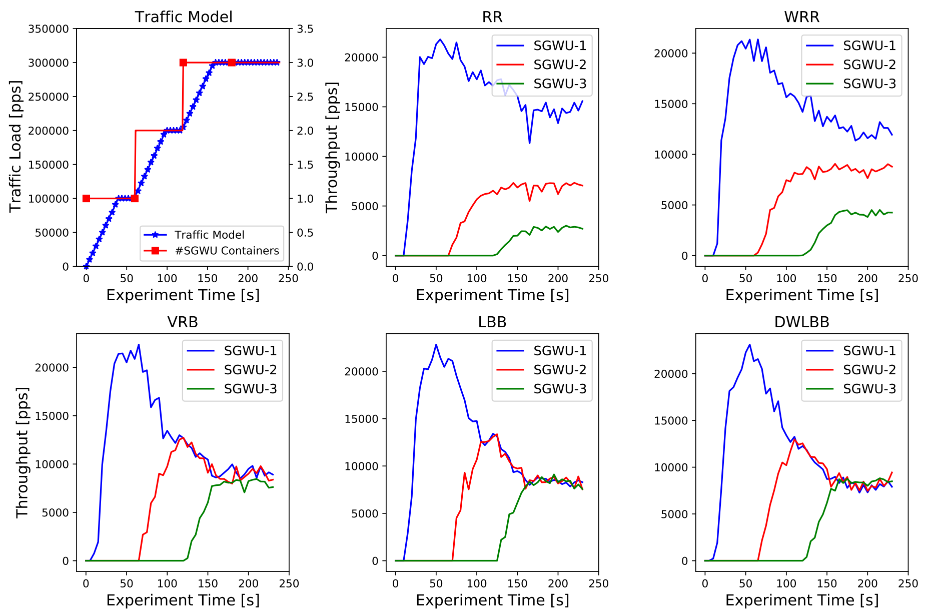

Fig. 6: Throughput Distribution Among SGWUs

a single cell are eventually attached to the network. In the first period, we create 2000 UEs that attach to the network with an attachment rate of 50 UEs/s. Once attached to the network, each active UE will generate data traffic with a fixed rate of 50 packets per seconds (pps). The second period starts 60 seconds after the first one, and add the same number of UEs, i.e., 2000 UEs will be created and attach to the network with the same attachment rate of 50 UEs/s. The last period starts 60 seconds after the second one and, similarly, 2000 UEs will attach to the network with the same attachment rate. This procedure means that every 60 seconds we add 2000 UEs to the network. The total experiment time is 240 seconds. The packet size is 1440 bytes. The packet rate and the number of UEs are chosen so that we are able to create a traffic load high enough to trigger the scaling out of new SGWUs. The VPP rate threshold that is used for triggering the scaling, is set to 225 over 20 seconds. Thus, we trigger the scaling in those cases the monitored VPP rate of a SGWU is consistently greater than 225 over the last 20 seconds.

B. *Preliminary Results*

In the following, we present our preliminary results from executing the experiment scenario described above. Firstly, the two-way packet latency is measured in order to compare the performance between the single-chain scenario, i.e., the scenario with only one SGWU container, and the multiple-chain scenario, i.e., the scenario with three SGWU containers and with scaling supported. In these experiments, a latency probe packet is sent from the UESIM to the CNSIM every three seconds, and is collected at the UESIM where the computation of the latency takes place. The result is plotted in Figure 5. As follows, the packet latency is higher in the single-chain scenario. Moreover, there is no significant difference in latency between the different load-balancing algorithms in the multiple-chain scenario. It should also be noted that the value of the packet latency depends heavily on the environment setup, however, it is sufficient to show that scaling out new SGWUs helps to reduce the packet latency.

In the next set of experiments, we compare the performance of our proposed dynamic load-balancing algorithms, i.e., the LBB, the VRB, and the DWLBB algorithms, and the two static load balancing algorithms, the RR and the WRR algorithms, during the scaling process. We compare them in terms of how well the throughput and the VPP rates are distributed among the SGWU containers. The throughput and the VPP rates are extracted every fifth second. The results are shown in Figure 6 and Figure 7. The upper-left corner sub-figures of each figure illustrate the offered traffic pattern used throughout the experiment. These figures also show the number of active SGWU containers serving traffic during the experiment. Overall, Figure 6 and Figure 7 show that our proposed LBB, VRB, and DWLBB load-balancing algorithms are indeed able to balance the traffic among the SGWU containers after every scaling trigger. As expected, the RR and the WRR algorithms are not able to balance out the traffic; the VRB algorithm shortens the time it takes to balance the throughput; and, the DWLBB algorithm, which combines the best of the LBB and the VRB algorithms, successfully balances the throughput.

VII. Conclusion and Future Work

In this paper, we consider the auto-scaling and load balancing of traffic in the user plane of a softwarized 5G core network. We propose a framework for managing the auto-scaling of the user-plane gateways using a MAPE approach. In our experiments, we show that a proper scaling of gateways in the user plane can help reduce the packet latency. By studying

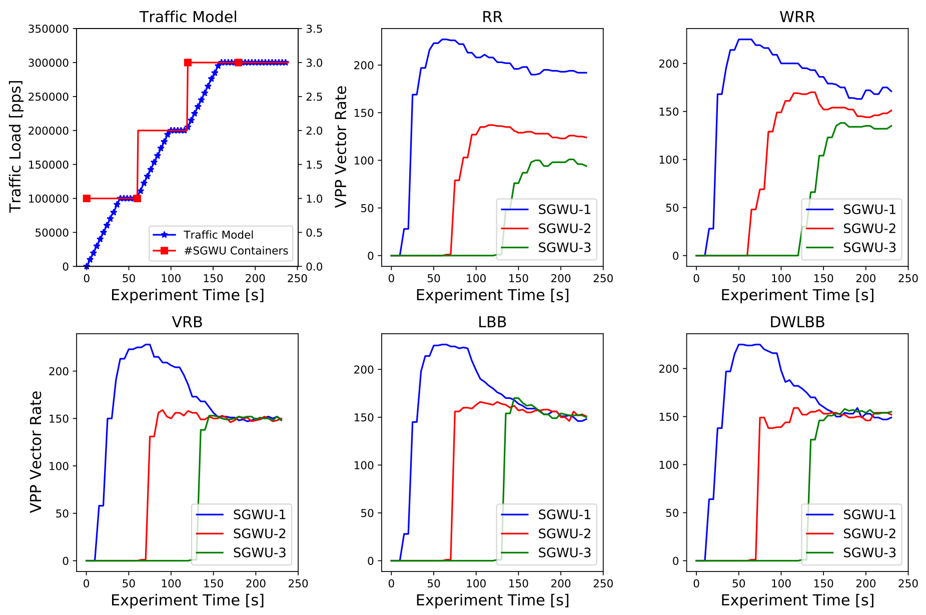

Fig. 7: VPP Vector Rate Distribution Among SGWUs

the correlation between the traffic load and the VPP vector rate, we propose the use of this parameter as a performance metric to support both the auto-scaling as well as the load-balancing decisions. The proposed auto-scaling method based on the estimation of the VPP rate helps overcome the challenge of estimating the load when VPP is used. Our proposed load balancing algorithms, the LBB, the VRB, and the DWLBB algorithms, perform well together with the scaling process. They are able to distribute the load within a user-plane gateway cluster much better than the RR and the WRR algorithms do. In our future work, we intend to extend our experiment to also include the evaluation of the scaling-in and the re-balancing features. We also plan to deploy the proposed auto-scaling framework using Kubernetes.

References

[1] V.-G. Nguyen *et al.*, "5G mobile networks: Requirements, enabling technologies, and research activities," in *Comprehensive Guide to 5G Security*, M. Liyanage, I. Ahmad, A. B. Abro, A. Gurtov, and M. Ylianttilla, Eds. John Willey & Sons Ltd., 2018, ch. 2, pp. 31–57.

[2] ——, "SDN/NFV-based mobile packet core network architectures: A survey," *IEEE Comm. Surveys Tuts.*, vol. 19, no. 3, pp. 1567–1602, 2017.

[3] "DPDK," [Online] Available: https://www.dpdk.org/.

[4] "Fd.io VPP," [Online] Available: https://fd.io/.

[5] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling web applications in clouds: A taxonomy and survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–33, 2018.

[6] M. A. López-Peña *et al.*, "Case studies in application placement and infrastructure optimisation," in *Managing Distributed Cloud Applications and Infrastructure*. Palgrave Macmillan, Cham, 2020, pp. 117–160.

[7] GSMA, "5G implementation guidelines," https://www.gsma.com/futurenetworks/wp-content/uploads/2019/03/5G-Implementation-Guideline-v2.0-July-2019.pdf, 2019, [Online; accessed 1-May-2021].

[8] 3GPP, "3GPP TR 23.714, study on control and user plane separation of EPC nodes, version 14.0.0," http://www.3gpp.org/DynaReport/23714.htm, 2016, [Online; accessed 1-May-2021].

[9] M. Olsson *et al.*, *SAE and the Evolved Packet Core: Driving the mobile broadband revolution*. Academic Press, 2009.

[10] A. S. Rajan *et al.*, "Understanding the bottlenecks in virtualizing cellular core network functions," in *The 21st IEEE LANMAN*. IEEE, 2015, pp. 1–6.

[11] X. An, W. Kiess, and D. Perez-Caparros, "Virtualization of cellular network EPC gateways based on a scalable SDN architecture," in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 2295–2301.

[12] L. J. Chaves, I. C. Garcia, and E. R. M. Madeira, "An adaptive mechanism for LTE P-GW virtualization using SDN and NFV," in *2017 13th International Conference on Network and Service Management CNSM*. IEEE, 2017, pp. 1–9.

[13] R. Mahindra *et al.*, "Orchestrating the data-plane of virtual LTE core networks," in *2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2017, pp. 1–9.

[14] S. K. Singh *et al.*, "Offloading virtual evolved packet gateway user plane functions to a programmable ASIC," in *Proceedings of the 1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms*, 2019, pp. 9–14.

[15] 3GPP, "3GPP technical specification (ts) 29.303; domain name system procedures; stage 3, (release 11)," https://www.3gpp.org/dynareport/29303.htm, 2012, [Online; accessed 1-May-2021].

[16] B.-H. Lee, E. K. Dewi, and M. F. Wajdi, "Using load balancing mechanism to reduce overload in LTE/EPC defined network," in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*. IEEE, 2019, pp. 1–5.

[17] T.-J. Tan *et al.*, "A reliable intelligent routing mechanism in 5G core networks," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–3.

[18] D.-H. Luong *et al.*, "Cloudification and autoscaling orchestration for container-based mobile networks toward 5G: Experimentation, challenges and perspectives," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. IEEE, 2018, pp. 1–7.

[19] C. H. T. Arteaga *et al.*, "A scaling mechanism for an evolved packet core based on network functions virtualization," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 779–792, 2019.

[20] T. V. K. Buyakar *et al.*, "Auto scaling of data plane vnfs in 5G networks," in *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 2017, pp. 1–4.

[21] D. Kumar, S. Chakrabarti, A. S. Rajan, and J. Huang, "Scaling telecom core network functions in public cloud infrastructure."