On Load Balancing for a Virtual and Distributed MME in the 5G Core

Van-Giang Nguyen, Karl-Johan Grinnemo, Javid Taheri, Anna Brunstrom
Department of Mathematics and Computer Science, Karlstad University, Sweden
Email: giang.nguyen, karl-johan.grinnemo, javid.taheri, anna.brunstrom@kau.se

***Abstract*—In this paper, we aim at tackling the scalability problem of the Mobility Management Entity (MME) which plays a crucial role of handling control plane traffic in the current 4G Evolved Packet Core as well as the next generation mobile core, 5G. One of the solutions to this problem is to virtualize the MME by applying Network Function Virtualization principles and then deploy it as a cluster of multiple virtual MME instances (vMMEs) with a front-end load balancer. Although several designs have been proposed, most of them assumed the use of simple algorithms such as random and round-robin to balance the incoming traffic without any performance assessment. To this end, we implemented a weighted round robin algorithm which takes into account the heterogeneity of resources such as the capacity of vMMEs. We compare this algorithm with a random and a round-robin algorithm under two different system settings. Experimental results suggest that carefully selected load balancing algorithms can significantly reduce the control plane latency as compared to simple random or round-robin schemes.**

I. Introduction

In recent years, the development of the next generation of mobile network system, 5G, has been actively carried out around the world. Many proposals and many technology candidates have been considered [1]. Among these technologies, Software Defined Networking (SDN) and Network Function Virtualization (NFV) have been widely considered as key enablers for the development of 5G, especially the mobile packet core [2]. The architecture of 5G core is being standardized by the 3rd Generation Partnership Project (3GPP) as a service-based architecture [3]. However, one of the most practical approaches currently adopted is to re-design the current 4G Evolved Packet Core (EPC) [4] architecture using SDN and NFV and let it serve as the core of 5G. In that way, the control and data planes are completely decoupled as prescribed by the SDN principle while the EPC's elements are virtualized as dictated by the NFV principle. Fig. 1 shows an SDN/NFV-based EPC architecture. In this new SDN/NFV-based EPC architecture, the Mobile Management Entity (MME) - one of the EPC's key control elements - remains unchanged but virtualized. It is still responsible for processing and handling of all signaling events coming from mobile devices such as attachment, detachment, handover, location update, etc. According to a white paper from Nokia [5], the signaling traffic is estimated to grow 50% faster than the growth

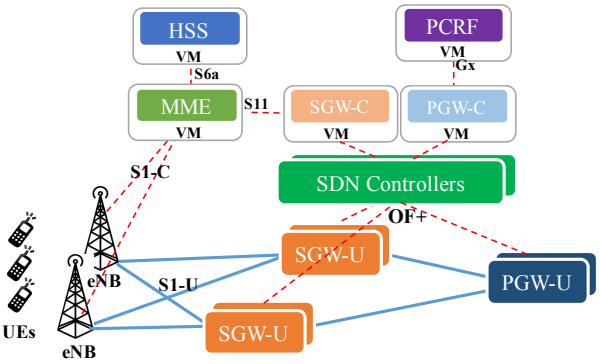


Fig. 1. An SDN/NFV-based EPC architecture [2].

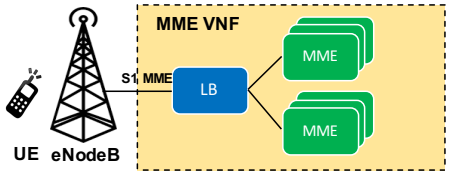
in data traffic. As the number of mobile devices keeps increasing with an unprecedented rate, the amount of signaling load the MME needs to handle increases significantly, thus resulting in a scalability issue. Moreover, any congestion or overload happening at the MME also has a severe impact on the performance of the user plane [6]. Therefore, it is necessary to investigate a new design of the control plane architecture, the MME in this case, so that it can be scaled dynamically according to any changes in the network, e.g., the number of connected devices.

One of the solutions is to deploy Virtual Network Functions (VNF) in a clustered fashion instead of a single virtualized server, which can not only improve the scalability but also provide the resiliency. This goes back to the concept of VNF pooling, which has been discussed in the Internet Engineering Task Force (IETF) [14]. Having a pool of VNFs requires a load balancer in front of the VNFs to distribute the incoming traffic among them. This way of designing the MME entails a cluster of virtual MME instances (vMMEs) and a front-end Load Balancer (LB). Although several designs have been proposed, they can be categorized into two main approaches: a stateful approach and a stateless approach. In the stateful approach, the user-related states are associated with vMMEs within the cluster. We call an MME architecture that follows this approach a two-tier vMME architecture. In the stateless approach, the user related states are no longer associated with vMMEs but

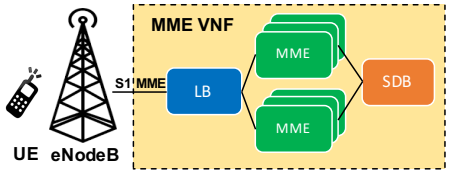
TABLE I
A SUMMARY OF CURRENT RESEARCH ON SCALABLE MME DESIGN

Approaches	Architecture	MME Type	LB Policy	Mapping Alg.	Mapping ID	Eval. Method
DMME [7]	2-Tier	Stateless	N/A	N/A	N/A	Analy. Model.
Yusuke et al. [8]	3-Tier	Stateless	RR	DHT	Group of IMSI	Prototype
Gopika et al. [9]	3-Tier	Stateless	RR	DHT	eNodeB UE S1AP ID	Prototype
SCALE [10]	2-Tier	Stateful	Least Loaded	CH	GUTI	Prototype
vMME [11]	3-Tier	Stateless	RR	N/A	N/A	Analy. Model.
Yousaf et al. [12]	2-Tier	Stateful	RR	N/A	N/A	Analy. Model.
Varudu [13]	2-Tier	Stateful	RD/RR	DHT	eNodeB UE S1AP ID	Prototype

RD = Random; RR = Round Robin; DHT = Distributed Hash Table; CH = Consistent Hashing



(a) Two-tier vMME architecture



(b) Three-tier vMME architecture

Fig. 2. Two types of vMME architecture.

stored in a separate centralized database called the State Database (SDB). An MME architecture that adheres to this approach is called a three-tier vMME architecture. These two types of vMME architectures are illustrated in Fig.2.

Despite the fact that several designs have been proposed and proved their benefits, they pay less attention to the load balancing aspect, which we found is one of the most important aspects of a distributed system. Indeed, most of the existing works assumed the use of simple algorithms such as Random (RD) and Round-Robin (RR) without any performance assessment. To the best of our knowledge, there has been no performance comparison of different load balancing algorithms used in these vMME architectures. In this paper, we propose using a Weighted Round-Robin (WRR) algorithm which takes into account the heterogeneity of resources such as link latency between the LB and vMMEs and the CPU capacity of vMMEs. We aim at comparing the performance of this WRR algorithm with a RD and a RR algorithm in a two-tier vMME architecture under different load conditions and different system settings. We focus on the two-tier vMME architecture due to the fact that the three-tier vMME architecture introduces an extra delay since user-related states need to be acquired from the SDB. Moreover, the state retrieval happens so often that the central database could become a bottleneck. We conducted experiments using the Open5GCore platform Release 2 [15]. We consider three different types of signaling events namely attachment, detachment, and handover. These signaling events generated by the Open5GCore's benchmarking tool follow a realistic signaling traffic pattern. The experimental results show that having multiple instances of MME not only improves the scalability but also helps reduce the latency for the control plane processing. Moreover, this reduction of control plane latency can be further improved by employing a specifically chosen load balancing algorithm, something which is demonstrated in two different scenarios. The first scenario entails using heterogeneous vMMEs, symmetric links between the LB and vMMEs, and high load. In this scenario, the WRR performs significantly better than the RD and RR schemes when the load is low. The second scenario is to use homogeneous vMMEs, asymmetric links between the LB and vMMEs. In this scenario, the WRR outperforms the other two schemes when the load is high. The obtained results provide insights for further improvement of the load balancing in the vMME architecture.

The rest of the paper is organized as follows. Section II presents an overview of related works. Section III describes the three studied methods to do load balancing of the MME. Section IV introduces the experimental scenarios and results. Finally, Section V concludes the paper and outlines future work.

II. RELATED WORKS

As previously discussed, there are two main approaches to improve the scalability of the MME, namely a stateful and a stateless approach. Table I summarizes all the existing proposals. They are compared in terms of architecture type, load balancing policy, mapping algorithm, user identifier (ID) used for mapping, and evaluation method. As the MME is now deployed as a distributed system, it is important to maintain a mapping between the UE and its corresponding vMME so that the LB knows to which vMME it should forward incoming requests. Different types of mapping algorithms and type of IDs are used. For example, a normal distributed hash

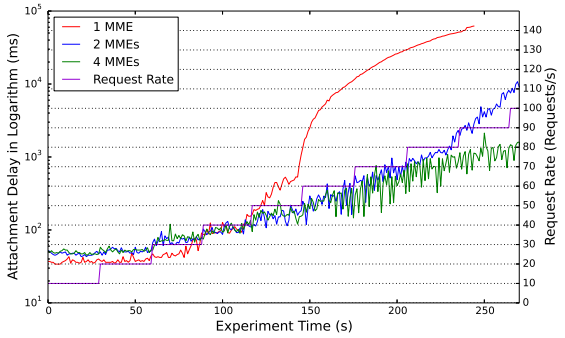


Fig. 3. Comparison of the attachment delay between the 1-MME, 2-MME, and 4-MME schemes.

table is used by Yusuke et al. [8], Gopika et al. [9], and Varudu [13], while the authors of SCALE [10] use consistent hashing to do mapping. Regarding the load balancing policy, as we can see from Table I, most of them rely on the simple load balancing mechanism, RR, due to its simplicity of implementation. In SCALE [10], state information of a UE is replicated twice using a consistent hashing algorithm and, a least loaded policy is used to balance the load among the two replicas which have UE states. Analytical modeling and prototyping are the two main methods used to evaluate the performance of the existing proposals. However, none of them reveals any performance evaluation of the load balancing algorithms. In contrast, we focus on evaluating the performance of different load balancing algorithms and their effect on the control plane latency under different load conditions and system settings.

III. MME Load Balancing

In this section, we present the motivation behind this work and MME load balancing approaches.

A. Motivation

It is well known that a clustered vMME architecture can improve the scalability and provide control plane resiliency in the SDN/NFV-based EPC. However, we would like to see the impact of this design on the control plane latency in comparison to a single vMME. To this end, we conducted an experiment using the Open5GCore platform with three different system settings: 1-MME, 2-MME, and 4-MME, which consists of 1, 2, and 4 vMME instances, respectively. In the 2-MME and the 4-MME settings, the load balancer randomly distributes the incoming requests among the vMME instances. All instances have the same capacity (2 CPUs, and an internal memory of size 4G). More information about the experimental environment is provided in Section IV. We start generate attachment requests with a starting rate of 10 requests/s. The rate is increased every 30 seconds, and the experiment stops when 10000 UEs are attached at an ending rate of 100 requests/s. The experiment results are shown in Fig. 3. From this figure, we can see that at the low attach rate (less than 30 requests/s) the 1-MME scheme has less attach delay than the other schemes. The reason to this is that adding a load balancer introduces an extra delay. However, when the rate increases, the 1-MME scheme performs worse and the attachment delay jumps up when the rate is about 50 requests/s while the 2-MME and 4-MME still perform well. At the rate of 80 requests/s, the 2-MME scheme starts performing worse than the 4-MME. Therefore, it is obvious that when we have more MMEs, the average attachment delay is significantly reduced. It should be noted that the load balancer distributes the incoming requests in a random manner. Therefore, one remaining question is that if the reduction of the control plane latency can be further achieved by having better load balancing solutions. The question will be answered in Section IV. In the following, we review some of the most commonly used methods to balance the control traffic in the MME.

B. Traditional approach

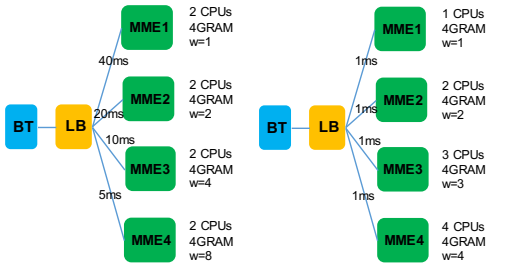
The S1-flex mechanism introduced by 3GPP is used to allow an eNodeB to connect to multiple MMEs within the same pool, thus providing support for redundancy and load sharing of control signaling traffic. Load balancing is achieved by setting a weight factor for each MME so that the probability of the eNodeB selecting an MME is proportional to its weight factor. The weight factor is typically set according to the capacity of an MME node relative to other MME nodes, and it is sent from the MME to the eNodeB via S1-AP messages. However, since an eNodeB connects directly to multiple MMEs, any changes in the MME pool must be informed to the eNodeB, thus reducing the flexibility of the pooling concept. Therefore, it is required to have a load balancer which is transparent to the radio access network (i.e., eNodeBs) and that can hide the operation of MMEs in the pool.

C. vMME approaches

As summarized in Table I, load balancing in the vMME architectures relies on a front-end load balancer which distributes incoming UE requests to the back-end vMMEs (either in a stateful or a stateless approach) based on different policies. So far, most of the existing approaches tend to use simple RD and RR algorithms which are well-known in distributed systems to do load balancing because of their simplicity. Also, since they pay less attention on load balancing of UE requests in such vMME designs, no performance on the load balancing aspect is evaluated. However, we argue that the RD and RR algorithms can only work well when the back-end servers are homogeneous and they also perform worse when there is the asymmetry on the

TABLE II
EXPERIMENTAL PARAMETERS

Parameters	Values	
	Scenario 1	Scenario 2
Number of UEs	10000	20000
Number of MMEs	4	4
Pre-Attach (UEs)	100	100
Rate (Request/s)	10, 30, 50, 80, 100, 150, 180, 200, 250, 300	50, 80, 100, 150, 200, 250, 300, 350, 400, 450
Metrics	Attach delay, CPU utilization	Attach delay, CPU utilization



(a) Scenario 1 (b) Scenario 2

Fig. 4. Detailed configuration of experimental scenarios.

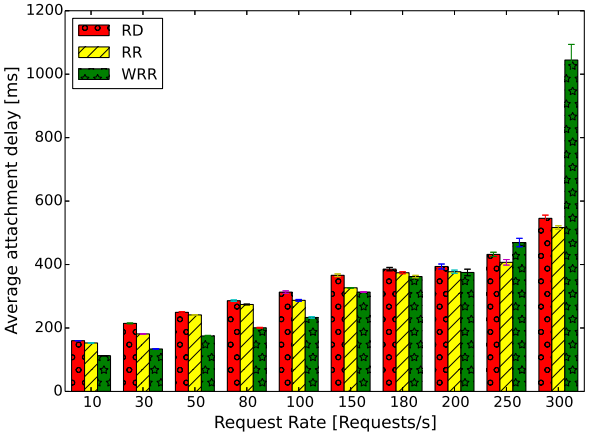


Fig. 5. Comparison of the attachment delay of the RD, RR, and WRR schemes in Scenario 1.

link latency between the LB and the back-end servers. To illustrate this, we implemented a WRR algorithm at the LB and carried out an extensive performance evaluation and comparison among the RD, RR, and WRR schemes. In the following, the basic concepts of these three algorithms are summarized.

1) RD Algorithm: As its name suggests, the RD load balancing algorithm randomly distributes incoming requests to the back-end servers. The LB treats all back-end servers the same regardless of their capacities and locations. When the number of requests is small, the difference in the number of requests being handled by each back-end server is significant. However, when the load balancer receives a large number of requests, the RD algorithm will be able to evenly distribute the requests. The RD algorithm is easy to implement but it only performs well for clusters consisting of servers with a similar configuration (CPU, memory, etc.).

2) RR Algorithm: Similar to the RD load balancing algorithm, the RR load balancing is one of the easiest load balancing algorithms to implement. The RR algorithm distributes incoming requests to the back-end servers in a round-robin manner. Like the RD, the RR works irrespective of the resource capabilities and location of the back-end servers. It is sufficient for clusters consisting of servers with identical specifications.

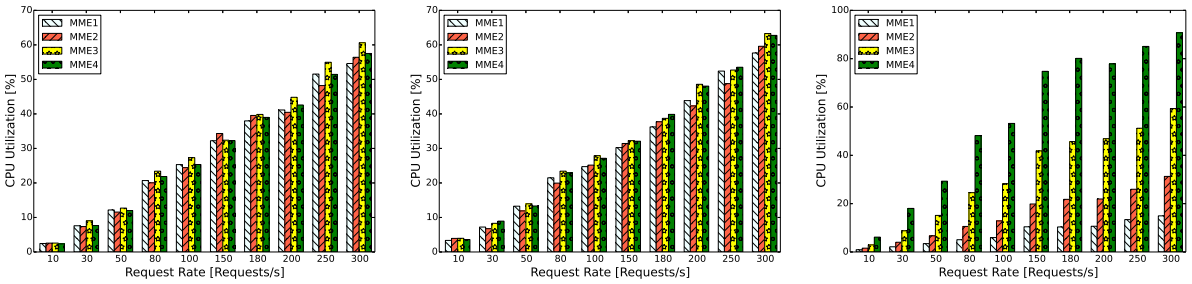
3) WRR Algorithm: In contrast to the RD and RR, in the WRR, each back-end server is assigned a value called "weight" (w), which is in proportional to the actual resource capabilities, i.e., CPU, memory of the server or can be proportional to the speed of the link from the LB to each server. In this way, the incoming requests are proportionally distributed to the weight of each server. For example, the server with a weight of 2 will receive two times more requests than the one with weight of 1. In our current implementation, the weight is statically set according to two factors: the number of CPUs allocated to the vMME servers, and the latency of links between the LB and the vMME servers.

IV. EXPERIMENTAL EVALUATION

This section describes our experiment setup using the Open5GCore platform and results from our experiments.

A. Experiment Setup

The Open5GCore platform [15] is a standards-compliant implementation of the current 4G EPC system. Our experiment setup uses KVM [16] as the virtualization platform. In Open5GCore Release 2, the MME is implemented according to a two-tier approach, and the load balancer randomly distributes the user requests. We extended the original test cases provided by the Open5GCore to be able to run with more than two MMEs. The Benchmarking Tool (BT) VM provides the emulation of the Long Term Evolution (LTE) radio access network including emulated UEs connecting to emulated eNodeBs. We consider two types of scenarios and the settings for these two scenarios are detailed in Table II. In order to have a more realistic scenario, we take into account the heterogeneity of signaling load by considering the mix of three different signaling events namely attachment, detachment, and handover. It has been shown in [17] and [18] that the number of attachment requests is similar to the number of detachment requests. And the number of S1-based handover requests is often a third to a half of the number of attachment requests. Based on this observation on signaling usage pattern in the LTE network, we use the following ratio



(a) Random (b) Round-Robin (c) Weighted Round-Robin

Fig. 6. CPU utilization of each MME instances in Scenario 1.

in our all experiment scenarios: 40% attachment, 40% detachment, and 20% handover. In addition, we start the experiments by pre-attaching 100 UEs to the system.

1) Scenario 1: In this scenario, we compare the performance of three load balancing schemes when the link latency from the LB to the MME instances are different while the resource capacities, i.e., CPU and memory of the MME instances are the same. The detailed configuration of the Experimental Scenario 1 is shown in Fig. 4 a). The link latency is emulated using the Netem traffic control tool in Linux. The weight factor (w) of each MME in the WRR scheme is assigned according to the link latency, as shown in Fig. 4 a). The number of UEs used in this scenario is 10000. We run the experiment with different overall request rates: 10, 30, 50, 80, 100, 150, 180, 200, 250, and 300 requests/s.

2) Scenario 2: In this scenario, we make the cluster of MME instances to be heterogeneous by allocating MMEs with different resources. In particular, the MME instances are allocated with the same amount of memory but different number of CPUs, as shown in Fig. 4 b). In addition, the link latency from the LB to each MME is kept the same. The weight factor (w) of each MME instance in the WRR load balancing scheme is assigned corresponding to their CPU capacity. To make it possible to have an overload situation, the total number of UEs configured in this scenario is 20000. We run the experiment with different overall request rates: 50, 80, 100, 150, 200, 250, 300, 350, 400, and 450 requests/s.

B. Experimental Results

In the following, we present the results obtained from the two experiment scenarios. The attachment delay and CPU utilization of each MME are measured to compare among three load balancing schemes.

1) Scenario 1: The experimental results for this scenario are shown in Fig. 5 and Fig. 6. The bar graphs in Fig. 5 show the mean of the average attachment delay for the three load balancing schemes in Scenario 1; the error bars in the graphs illustrate the 95% confidence intervals. As observed from Fig. 5, the attachment delay increases as the request rate increases for all three load

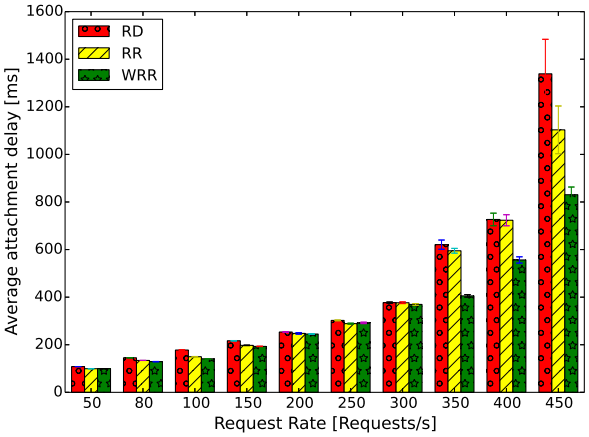
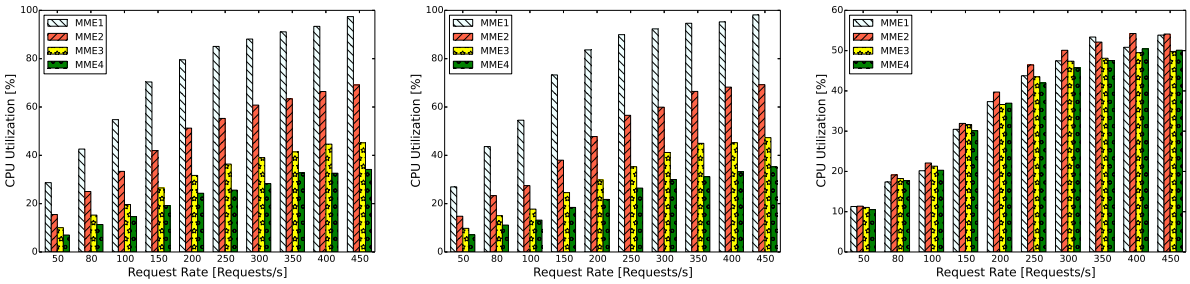


Fig. 7. Comparison of the attachment delay of the RD, RR, and WRR schemes in Scenario 2.

balancing schemes. We observe that when the overall request rate is less than 200 request/s which is equivalent to 80 attachment requests/s, the attachment delay in the WRR scheme is up to 38% lower than that of the RD and the RR schemes. The reason is that, in the WRR scheme, according to the weight assignment, there is more requests (8 times) being handled by the 4th MME which has the lowest link latency to the LB than the 1st MME which has the highest link latency to the LB. Whereas, the requests are distributed evenly (in RR) or almost evenly (in RD) among the four MMEs. Therefore, the number of requests being handled by each MME which has long delay is higher than that of the WRR. When the overall rate is about 250 requests/s and more, the WRR scheme performs worse because with the weight of 8, the 4th MME has to process a large number of UEs at a high rate, the queue is filling up quickly, and the CPU utilization of this MME goes up, thus resulting in a higher attachment delay than that of the RD and the RR schemes.

By observing the CPU utilization of each MME in Fig. 6, it is clear that when the request rate increases, the CPU utilization of each MME also increases in all



(a) Random (b) Round-Robin (c) Weighted Round-Robin

Fig. 8. CPU utilization of each MME instances Scenario 2.

three schemes. The CPU utilization of 4 MMEs in the RD and the RR schemes are almost the same for each request rate. Whereas, the CPU utilization of 4 MMEs in the WRR scheme is proportional to the weight factor assigned to each MME (as shown in Fig. 4 a)).

2) Scenario 2: The experimental results for this scenario are shown in Fig. 7, and Fig. 8. The bar graphs in Fig. 7 show the mean of the average attachment delay under three load balancing schemes in Scenario 2; the error bars in the graphs illustrate the 95% confidence intervals. We observe that when the request rate increases, the attachment delay also increases in all three schemes. When the overall rate is less than 300 request/s which is equivalent to 120 attach requests/s, the attachment delay in the three schemes is almost the same. However, when the overall rate is higher, the WRR starts outperforming the other two schemes. For example, at the overall rate of 450 requests/s which corresponds to 180 attach requests/s, the attachment delay for the WRR scheme is about 40% lower than that for the RD scheme. The reason is that when the rate is that high, in the RD and the RR schemes, the number of requests that needs to be handled by the MME1 which has less CPU resource is higher than its maximum capacity, that leads to the saturation of CPU resources. As a consequence, the attachment delay goes up. However, in the WRR scheme, since we know the amount of CPU resource allocated to each MME and we assign the weights accordingly, the number of requests being handled by each MME will be proportional to the assigned weight. Therefore, contrary to the RD and RR schemes, no overload happens. It implies that MMEs can still receive more requests to process in the WRR scheme.

As seen in Fig. 8, the CPU utilization of each MME also increases when the request rate increases. In the RD and the RR, the distribution of CPU utilization of each MME corresponds to their allocated capacity. When the overall rate is getting higher (from 350 requests/s), we can see that the CPU utilization of the MME1 goes up to approximately 100% utilization. That explains the high attachment delay values in the RD and the RR schemes shown in Fig. 7. Whereas, the CPU utilization of each MME in the WRR scheme are almost the same for each request rate, and is far from the saturation point even at the overall rate of 450 requests/s.

V. Conclusion and Future Works

In this paper, we have discussed the scalability of the MME by presenting two main vMME architectures: a two-tier and a three-tier architecture. In addition, our initial experiment results of the two-tier architecture have illustrated the benefits of having multiple vMME instances in reducing the control plane latency when the system is heavily loaded. However, most of the existing approaches pay less attention on the load balancing function while implementing such a new vMME architecture. In addition, the two currently available load balancing schemes namely the RD and the RR are not sufficient enough since they do not take into account the resource capacities and the locations of the MMEs. It is illustrated by comparing the results of these two algorithms with our implemented WRR algorithm that WRR offers a significantly reduced attachment delay and resource utilization. However, the current implementation of the WRR algorithm uses statically assigned weights of vMME instances. As part of our future work, we intend to enhance our WRR algorithm by letting the weights be dynamically assigned on the basis of information fed back from the vMMEs about their actual loads. We are also taking into account the difference of signaling events while making the balancing decision. In addition, several scaling strategies such as proactive/reactive and hybrid scaling are being investigated in conjunction with the load balancing function in order to create a scalable and efficient vMME solution.

Acknowledgment

The work was carried out in the High Quality Networked Services in a Mobile World (HITS) project, funded by the Knowledge Foundation of Sweden.

REFERENCES

[1] V.-G. Nguyen, A. Brunstrom, K.-J. Grinnemo, and J. Taheri, "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 Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1567–1602, 2017.

[3] 3GPP, "3GPP ts 23.501 technical specification group services and system aspects; system architecture for the 5G system; stage 2 (release 15)," 2017.

[4] ——, "3GPP ts 23.401 technical specification group services and system aspects; general packet radio service (gprs) enhancements for evolved universal terrestrial radio access network (e-utran) access, (release 10)," 2011.

[5] Nokia Siemens Networks, "White paper: Signaling is growing 50% faster than data traffic," http://goo.gl/aUKANS, 2012, [Online; accessed 6-June-2017].

[6] A. S. Rajan, S. Gobriel, C. Maciocco, K. B. Ramia, S. Kapury, A. Singhy, J. Ermanz, V. Gopalakrishnanz, and R. Janaz, "Understanding the bottlenecks in virtualizing cellular core network functions," in *The 21st IEEE LANMAN*. IEEE, 2015, pp. 1–6.

[7] X. An, F. Pianese, I. Widjaja, and U. Günay Acer, "DMME: a distributed LTE mobility management entity," *Bell Labs Technical Journal*, vol. 17, no. 2, pp. 97–120, 2012.

[8] Y. Takano, A. Khan, M. Tamura, S. Iwashina, and T. Shimizu, "Virtualization-based scaling methods for stateful cellular network nodes using elastic core architecture," in *Proc. of the 6th IEEE CloudCom*. IEEE, 2014, pp. 204–209.

[9] G. Premsankar, K. Ahokas, and S. Luukkainen, "Design and implementation of a distributed mobility management entity on OpenStack," in *Proc. of the IEEE CloudCom*. IEEE, 2015, pp. 487–490.

[10] A. Banerjee, R. Mahindra, K. Sundaresan, S. Kasera, K. Van der Merwe, and S. Rangarajan, "Scaling the LTE control-plane for future mobile access," in *Proc. of the 11th ACM CoNEXT*. ACM, 2015.

[11] J. Prados-Garzon, J. J. Ramos-Munoz, P. Ameigeiras, P. Andres-Maldonado, and J. M. Lopez-Soler, "Modeling and dimensioning of a virtualized MME for 5G mobile networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 4383–4395, 2017.

[12] F. Z. Yousaf, P. Loureiro, F. Zdarsky, T. Taleb, and M. Liebsch, "Cost analysis of initial deployment strategies for virtualized mobile core network functions," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 60–66, 2015.

[13] R. Varudu, "Design and implementation of a flexible mme load balancing solution for carrier grade virtual mobile core networks," Master's thesis, The Technical University of Berlin, Berlin, 2016.

[14] D. King, M. Liebsch, P. Willis, and J. dong Ryoo, "Virtualisation of mobile core network use case," Working Draft, IETF Secretariat, Internet-Draft, January 2015. [Online]. Available: http://www.ietf.org/internet-drafts/draft-king-vnfpool-mobile-use-case-02.txt

[15] Fraunhofer FOKUS Research Institute, http://www.open5gcore.org/, [Online; last accessed 18-March-2018].

[16] KVM Virtualization, https://www.linux-kvm.org/, [Online; last accessed 18-March-2018].

[17] B. Hirschman, P. Mehta, K. B. Ramia, A. S. Rajan, E. Dylag, A. Singh, and M. Mcdonald, "High-performance evolved packet core signaling and bearer processing on general-purpose processors," *IEEE Network*, vol. 29, no. 3, pp. 6–14, 2015.

[18] S. Tabbane, "Core network and transmission dimensioning," http://goo.gl/kYyiKe, [Online; last accessed 18-March-2018].