Proactive Live Migration for Virtual Network Functions using Machine Learning

Seyeon Jeong, Nguyen Van Tu, Jae-Hyoung Yoo and James Won-Ki Hong

Department of Computer Science and Engineering, POSTECH, Korea.
{jsy0906, tunguyen, jhyoo78, jwkhong}@postech.ac.kr

***Abstract*—VM (Virtual Machine) live migration is a server virtualization technique for deploying a running VM to another server node while minimizing downtime of service the VM provides. Currently, in cloud data centers, VM live migration is widely used to apply load balancing on CPU workload and network traffic, to reduce electricity consumption, and to provide uninterrupted service during the maintenance of hardware and software updates on servers. It is critical to use VM live migration as a prevention or mitigation measure for possible failure when its indications are detected or predicted. Especially in NFV (Network Function Virtualization) environment, timely use of VNF (Virtual Network Function) live migration can maintain system availability and reduce operator's loss due to service failure. In this paper, we propose a proactive live migration method for vEPC (Virtual Evolved Packet Core) based on failure prediction. A machine learning model learns periodic monitoring data of resource usage and logs from servers and VMs/VNFs to predict future vEPC paging failure probability. We implemented the proposed method in OpenStack-based NFV environment to evaluate the real service performance gains for open source vEPC implementations.**

***Index Terms*—VNF live migration, Machine learning, Virtual EPC**

I. Introduction

With the evolution of server virtualization, service applications running in VMs have less restriction on their physical locations (e.g., edge or central cloud) and their particularity (e.g., available resources). VM live migration is an important operation in server virtualization to move a running VM to another server node with a minimum service downtime using two alternate VMs during the migration process. The memory synchronization between those two VMs and the corresponding network path reconfiguration incur such service downtime [1].

VM live migration is widely used in cloud computing to apply load balancing on CPU workload and network traffic and to reduce electricity consumption by consolidating active VMs into a specific location group of servers in a data center. VM live migration is also used to provide uninterrupted service during the maintenance of a data center, for instance, host kernel update or server/network repair [2]. In addition, VM live migration can be used for a prevention or mitigation measure to faults when their indications are detected or predicted. As one example of fault prevention, operators can live-migrate VMs running on a server that shows anomalously high memory utilization and temperature for a long time, and take an investigation without affecting the service quality. In NFV (Network Function Virtualization) environment, VM live migration has an essential role in maintaining the availability of VNFs (Virtual Network Functions) running on VMs. This VNF live migration can reduce the operator's loss (e.g., service level violation) due to abrupt service down by failure.

In this paper, we propose a proactive live migration method for vEPC (Virtual Evolved Packet Core) based on machine learning prediction on failure. This method collects resource usage (e.g., CPU, memory, disk), system logs (syslog), and application/VNF logs. Such time-series data is fed to a deep learning model to learn patterns of the system's status over time including failure history. Then a migration decision is made to proactively migrate a failure-expected vEPC instance to another (healthy) server, based on the failure score from the prediction model. We implemented a VNF live migration system in our OpenStack-based NFV testbed, and evaluated the effectiveness of our solution when vEPC fails in paging user devices for downlink data transfer in case of the host server failure.

The remainder of this paper is organized as follows. Section II presents related work and Section III describes design and implementation considerations of the proposed system. Then, we provide a case study on a vEPC paging failure and verification on the proactive vEPC live migration method in Section IV. Finally, we conclude our work in Section V including future work.

II. Related Work

The pertinence problem of migrating an abnormal VNF in core cloud to edge cloud is considered in [3]. The migration is made only when the migration cost of reconfiguring service paths to its new location in the edge is lower than the operator's loss of QoS degradation from retaining the abnormal VNF. During this migration decision-making, a deep learning algorithm is used to predict a user mobility pattern in a candidate edge site. This work provides a promising VNF migration scenario in MEC (Multi-access Edge Computing), but its verification is omitted in the paper.

In failure prediction, the authors of [4] propose a server failure prediction model using machine learning. They feed to the proposed model, temporal data (e.g., memory usage) and spatial data (e.g., rack location) collected from half a million physical servers in their data centers. The model then ranks

978-3-903176-36-2 ©2021 IFIP

top-k servers with predicted failure scores and shows that there is a tendency that servers with failure history are more likely to be failed again in the future. The authors prove that the overall VM failure rate decreases by 30% after using the failure prediction score in allocating and migrating VMs. VNF failure also can be predicted by learning system logs [5]. In this work, syslog of telco production vPE (Virtual Provider Edge) router instances is fed into a deep learning model that detects abnormal patterns of syslog and generates early warnings to possible vPE failure. Although it is difficult for individual researchers to access such production data in both [4] and [5], they show the effectiveness of failure prediction using machine learning and the possibility of applying proactive VNF live migration in a real environment.

A VNF backup method can be compared with a VNF live migration method. A generic VNF backup method maintains at least two instances of one master and slaves for one VNF type to ensure one of the slaves immediately takes over the master role when the master becomes failed [6]. This approach has the advantage of minimized service downtime thanks to previously configured backup paths, but it requires more resource allocation and a refined state synchronization mechanism among the group of VNF instances. On the other hand, VNF live migration has longer service downtime but it requires fewer resources and is stateful (i.e., sessions remained after migration). One most dominant clouding platform, OpenStack, provides the stateful VM live migration by making many low-level considerations transparent to users.

III. Design and Implementation

The proposed VNF live migration system consists of the Monitoring module, the Prediction module, and the Migration module (in Fig. 1). This system operates as one modular function in the previously proposed Network Intelligence architecture [7]. The implementation of this architecture in our site references the ETSI NFV-MANO model [8], so the VNF live migration system proposed in this paper is compliant with other NFV management functions in our previous work (e.g., auto-scaling [9]).

In the proposed system, the Monitoring module periodically collects monitoring data of resource usage, system logs (syslog), and application-level VNF logs from servers and VMs/VNFs. We will show how log data is used to predict vEPC failure in Section IV. To implement this module, we deploy collectd as monitoring agents on both servers and VMs to collect their CPU, memory, disk and network I/O. Likewise syslog, log messages of individual VNFs (if generated in a file) are also registered to the collectd log module to be sent towards a central monitoring node. We use influxDB to store time-series data and build Grafana and Graylog for analysis on time-series data and log messages respectively. To integrate with machine learning algorithms in the Prediction module, data can be served as either query-base (Swagger OpenAPI) or Publication/Subscription-base (Apache Kafka).

Next, the Prediction module operates machine learning models that are used to assist VNF live migration decisions.

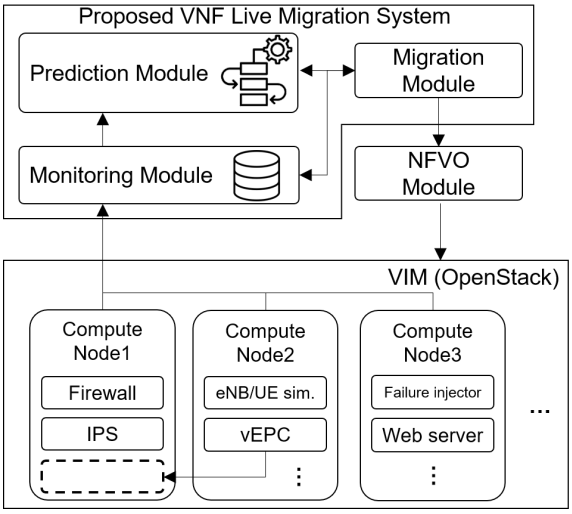


Fig. 1: Proposed VNF live migration system architecture

We discuss in Section IV how this module predicts vEPC failure in detail. By learning monitoring data that can reflect repeated patterns of workload and network traffic in a data center, the model also identifies the best destination server for VNF live migration, based on predicted resource states or capacities of all servers in near future.

Lastly, the Migration module generates a final decision on VNF live migration. This module requests the Prediction module to return the best migration destination for VNFs running in an anomalous node. We note that the best destination server may not be the least load one because such greedy selection does not ensure the optimality of the migration decision if there should be an upcoming heavy workload or traffic on the server. Then this server is likely to be overloaded, and the next migration decision could be made against the previously migrated VNFs again, leading to the cost of service downtime. Whereas, our VNF live migration method uses the prediction about upcoming server states from the machine learning models. Fig. 2 supports the fact and provides average throughput and response time of a web server VM in different destination server selections for live migration. Our method (proposed) shows 13%, 6% and 8% increase in the average number of requests per second, compared to no-mig, random and cpu-least policy respectively.

The NFVO module that is implemented in the previous work [7] takes the VNF migration decision parameters of the target VNF IDs and the destination server ID (OpenStack manages) from the Migration module and calls OpenStack VM live migration APIs to enforce the migration decisions into the testbed.

IV. Case study on vEPC live migration

In this section, we describe the mechanism of paging UE in vEPC and a vulnerability issue of the (consecutive) vEPC paging failure [10]. Then we introduce our approach to tackle the issue using proactive vEPC live migration based on the

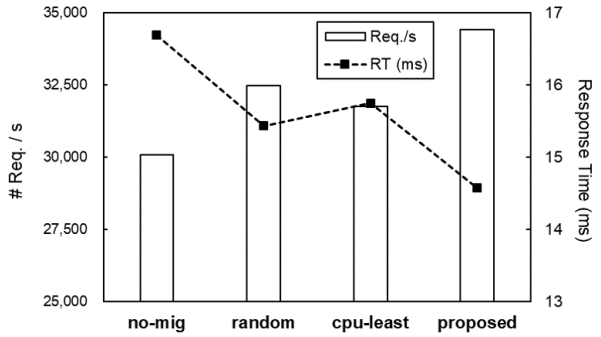


Fig. 2: Average performance of a web server VM in different destination server selections for live migration

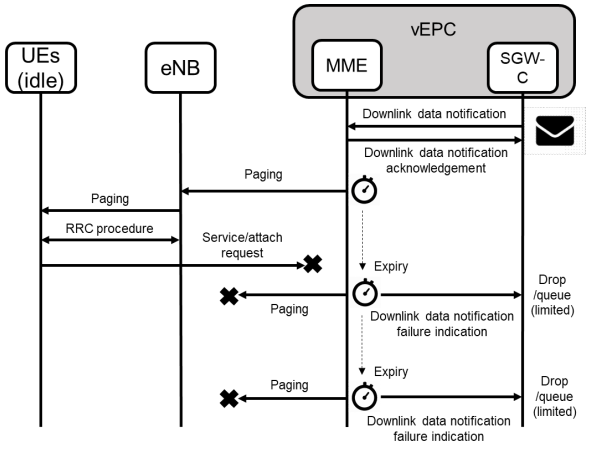


Fig. 3: vEPC paging failure

failure prediction. Finally, we evaluate its effectiveness as a failover solution to cloud-based vEPC deployment from the vEPC failure scenario injected into our testbed (Section III).

A. Paging Mechanism in vEPC

The EPC is the core network architecture for 3GPP LTE (Long Term Evolution) networks. It consists of separated network functions (NFs) of MME (Mobility Management Entity), S-GW (Serving Gateway), P-GW (PDN Gateway), and others. With the evolution of server virtualization, this functional split accelerates the realization of CUPS (Control and User Plane Separation) and the deployment of virtualized EPC in the telco's cloud. These virtualized NFs of vEPC that can operate in VMs are much flexible and scalable (e.g., auto-scaling) from benefits of cloud computing than its deployment in vendor-specific proprietary hardware [11]. However, this vEPC also inherits the weaknesses of server virtualization and cloud computing. One major factor is relatively unreliable host COTS servers that are vulnerable to malfunction or failure due to hardware (e.g., broken disk) or software issues (e.g., security breach).

The MME of EPC is the key control node for LTE access network. It is responsible for session and mobility management of user devices (UEs) via eNB (eNodeB). One major procedure of session management is awakening an idle-state UE to enable it to stop the recess period (e.g., for energy saving) and to reattach the MME (EPC) so that downlink data can be transferred to the UE. In this paging procedure, a paging message is sent from MME to eNB first over the S1AP/SCTP protocol. The idle-state UE who receives the paging message relayed by the eNB sends back the service request message to the MME to resume its session (e.g., data plane tunnel) previously allocated by the EPC during the initial connection establishment. However, in the case of MME in vEPC, if the host server of the vEPC or vEPC VM itself becomes malfunctioned or failed, there will be a possibility of failure in the bidirectional paging mechanism.

According to the 3GPP standard for paging failure (3GPP TS 24.301 [12]), a paging timer, called T3413, is created in MME once a paging message is sent to UE, and then MME retransmits a paging message at the timer expiry with the creation of a new timer (Fig. 3). At the same time of the timer expiry, the ***Downlink_Data_Notification_Failure_Indication*** message is sent to S-GW, who holds the downlink data in its (limited) queue or drops it once receiving the notification signal. If the malfunction of vEPC is temporary (e.g., resource overload), the paging procedure can be recovered by retransmissions of paging messages. However, if the malfunction is long-term due to the host server failure, it costs the following until failover.

- **Loss of downlink data:** the amount of downlink data loss is proportional to the number of UEs whose sessions are managed by the vEPC on the failed host server.
- **Wasted resources:** similarly, MME must hold resources (e.g., session information, timers) allocated to the non-responding UEs because it does not know when attempts to reattachment are resumed.

Failover procedure to this consecutive paging failures is not standardized and depends on vendor-specific implementation. For the three representative open source vEPC implementations (upstream) as of this writing, we analyze their behavior in the case of consecutive paging failure. We observed that those vEPC implementations behave differently, and srsRAN (successor of srsLTE) [13] and NextEPC appear not to tackle the paging failure properly but to pay the costs above. Open5GS (successor of NextEPC) [14] shows an active handling for the consecutive paging failure by freeing the resource wasted by non-responding UEs but the loss of downlink data is unavoidable. To properly manage this vEPC failure, an NFV orchestration-level solution is needed considering the vEPC deployment in a cloud data center. In this context. we propose a proactive vEPC live migration method using deep learning-based vEPC failure prediction.

B. Prediction on vEPC Paging Failure

As mentioned in Section II, VM/VNF live migration in OpenStack is stateful because it ensures memory synchronization between two VMs used in the process and the same IP

address is preserved to the migrated VM with the corresponding network reconfigurations (e.g., changing forwarding rules) to the new location. In order words, most of the changes due to VM live migration is transparent to peers of the existing connections except for the short service downtime. Therefore, VM live migration is a suitable operation to manage and orchestrate vEPC instances. We focus on vEPC containing NFs altogether in a single VM (all-in-one) but it is also possible to deploy EPC NFs in different VMs of the same subnet. In the latter case, our approach proactively migrates an MME VM.

To predict the vEPC paging failure mainly due to the host server failure, we use an LSTM (Long Short-Term Memory) model to learn time-series resource (CPU, memory) usage of the host server and vEPC VM. The resource utilization values can be useful indicators for anomaly (e.g., overload), but they are too generic to be used solely for comprehending the specific condition of the consecutive paging failure. Therefore, we use the vEPC log data of ***Downlink_Data_Notification_Failure_Indication*** which is generated at the time of T3413 paging timer expiry. This log as its name implies is a clear indication for the consecutive paging failure but not always leads to the failure. There would be other useful vEPC logs that can increase the failure prediction accuracy but the logging is dependent on vEPC implementations (i.e., vendor-specific), so we only use the standardized log message. Considering that excessive memory consumption is one major cause of server failure [4] [15], we use the swap disk utilization that indirectly indicates the out of memory state and we also refer to the syslog of ***OOM_killed_process_PID*** which is generated when a process is forcibly terminated by the OOM (Out of Memory) killer in the host OS kernel thread. The overall architecture of the LSTM model to predict the vEPC failure is shown in Fig. 4.

To apply supervised learning to our LSTM model for better prediction accuracy, failure label (ground-truth) data is needed. The vEPC log of ***BROKEN_PIPE*** can be an obvious failure label in that in Linux it represents "connection closed by peer", which means the tunnel for control signal between vEPC and UE (via eNB) is closed by the eNB/UE side first. This happens when eNB finds from its SCTP configurations (e.g., max retransmissions, heartbeat timeout) that vEPC (MME) is unreachable and then closes the socket. The connection can be reopened by an attach-request from the UE side, but vEPC should be still not responding until failover.

Finally, we generate a dataset[1] by repeatedly injecting resource anomaly and vEPC failure scenarios ("broken pipe") of different periods and there premonitory conditions ("downlink data failure indication"). According to [15], flapping network cards and unexpected host memory consumption are major causes of server (node) failure in data centers, and they use VM live migration to maintain the service quality by relocating VMs to healthy servers. To simulate such server failure cases, we implemented a server failure injector that utilizes a resource stress tool (stress-ng) and some Linux commands to control network interfaces of servers in our testbed (Fig. 1).

[1]https://github.com/syjeong96/vEPC_failure_prediction.git

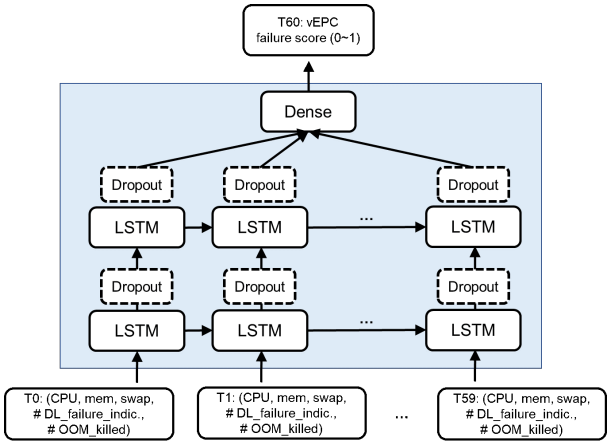
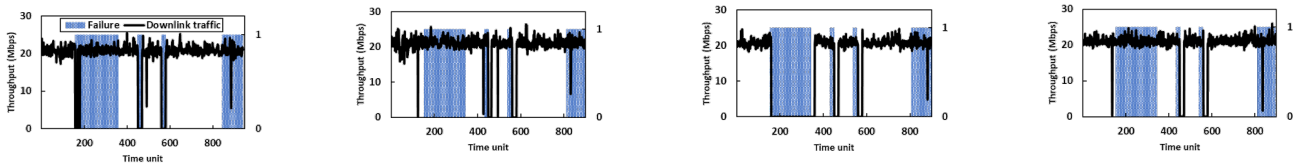


Fig. 4: LSTM model for predicting vEPC paging failure

C. Verification

1) Prediction model: We use the dataset of 11,696 records and each record for a time unit (e.g., 5 seconds monitoring interval) includes CPU/memory/swap space utilization, the number of "vEPC downlink failure indication" logs, the number of "OOM killed process" syslog, and the occurrence of "vEPC broken pipe" log as a failure label (0 or 1). The proportion of failures in the total time units is 2,898 and the total failure duration consists of 95% of short-term (30, 50, 100 time units) and 5% of long-term (500, 1000, 2000 time units). We assume that the failures in short-term periods represent temporary server anomaly (e.g., resource overload) which can be resolved over time and the resulting vEPC paging failures also can be recovered by paging retransmissions. Proactive vEPC migration to avoid those short failure periods may be an excessive measure with no benefit or even degradation in service performance, considering the incurred service downtime and completion time possibly longer than the failure period. We used Keras and TensorFlow to build the LSTM models in the Prediction module (Fig. 1).

2) Downlink throughput: In this section, we compare downlink throughput measurements with and without the proposed proactive vEPC live migration, using the generated vEPC failure prediction models and the testing dataset which is a part of the collected dataset (records) in the previous section. In the testing scenario, the iPerf client VM sends TCP traffic to the eNB/UE simulator (srseNB/srsUE in srsRAN) VM via the vEPC (srsEPC in srsRAN) VM. We can expect the followings: if the vEPC under the test is in a failure period, the downlink throughput is 0, else the vEPC is proactively migrated to one healthy server before entering a failure period, then the downlink traffic is preserved with a temporary loss from the live migration service downtime. The timing of a migration can be tuned by a threshold for the predicted failure score at the moment; a threshold of 0.5 leads to an aggressive



(a) threshold: 0.5, horizon: 5min., throughput: 19.6015Mbps

(b) threshold: 0.5, horizon: 10min., throughput: 20.1964Mbps

(c) threshold: 0.9, horizon: 5min., throughput: 15.5449Mbps

(d) threshold: 0.9, horizon: 10min., throughput: 20.0941Mbps

Fig. 5: Visualization of downlink traffic in different proactive migration policies

policy with more migrations and a threshold of 0.9 leads to a conservative policy with fewer migrations.

The average downlink throughput in different policy parameters are shown in Fig. 5. The blue patterned boxes indicate the actual failure periods (ground-truth) of the vEPC in the testing dataset. The black curved line represents downlink traffic throughput measured at the eNB/UE simulator VM. There are two types of the plunge in downlink throughput: one is from service downtime during the live migration and the other one is from the vEPC failure. For the former, the amount is negligible, compared to the amount of loss when the failure is not handled by proactive migration. For the latter, it occurs when the migration decision is not enough early for the actual migration process to be completed before the failure (i.e., the original vEPC VM still serves the traffic) or the migration decision itself is not made due to the policy parameters (including the model prediction accuracy) being not sufficiently tuned well to catch the failures. For all the policies in Fig. 5, no one properly tackles the two short failures in the 400 to 600 time units. Nevertheless, the proposed proactive live migration effectively prevents the downlink traffic loss during the long-term vEPC failures. The average downlink throughput without any migration policy was measured as 13.2778Mbps which is 65% of the best migration policy.

V. Conclusion

In this paper, we propose the proactive vEPC live migration method based on machine learning-based prediction on vEPC paging failure. The proposed LSTM model learns about resource utilization, OS/vEPC logs and vEPC failure history (as labeled data) so that it can predict the future failure score of the target vEPC. We then verify the method can migrate a failure-expected vEPC to another server proactively enough and compare the downlink traffic throughput for different configurations of the method in our OpenStack-based NFV testbed with an open source vEPC implementation.

Due to the difficulty of accessing open server/service failure data in real cloud data centers, we plan to refine our failure injection strategy to be more specific and useful by inspecting various cloud computing and NFV use cases more. We will also improve the machine learning models for prediction.

Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (2018-0-00749, Development of Virtual Network Management Technology based on Artificial Intelligence) and the ITRC (Information Technology Research Center) support program (IITP-2021-2017-0-01633).

References

[1] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, 2005, pp. 273–286.

[2] G. C. Engine, "Setting instance availability policies." [Online]. Available: https://cloud.google.com/compute/docs/instances/setting-instance-scheduling-options

[3] A. L. Ibrahimpašić, B. Han, and H. D. Schotten, "Ai-empowered vnf migration as a cost-loss-effective solution for network resilience," in *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2021, pp. 1–6.

[4] Q. Lin, K. Hsieh, Y. Dang, H. Zhang, K. Sui, Y. Xu, J.-G. Lou, C. Li, Y. Wu, R. Yao *et al.*, "Predicting node failure in cloud service systems," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 480–490.

[5] Z. Li, Z. Ge, A. Mahimkar, J. Wang, B. Y. Zhao, H. Zheng, J. Emmons, and L. Ogden, "Predictive analysis in network function virtualization," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 161–167.

[6] H. Huang and S. Guo, "Proactive failure recovery for nfv in distributed edge computing," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 131–137, 2019.

[7] S. Lange, N. Van Tu, S.-Y. Jeong, D.-Y. Lee, H.-G. Kim, J. Hong, J.-H. Yoo, and J. W.-K. Hong, "A network intelligence architecture for efficient vnf lifecycle management," *IEEE Transactions on Network and Service Management*, 2020.

[8] ETSI ISG NFV, "Standards for nfv - network functions virtualisation." [Online]. Available: https://www.etsi.org/technologies/nfv

[9] D. Lee, J.-H. Yoo, and J. W.-K. Hong, "Deep q-networks based auto-scaling for service function chaining," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–9.

[10] S. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino, "Lteinspector: A systematic approach for adversarial testing of 4g lte," in *Network and Distributed Systems Security (NDSS) Symposium 2018*, 2018.

[11] M. T. Raza, S. Lu, and M. Gerla, "vepc-sec: Securing lte network functions virtualization on public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3287–3297, 2019.

[12] 3GPP, "Non-access-stratum (nas) protocol for evolved packet system (eps); stage 3." [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1072

[13] srsRAN, "srsran - your own mobile network." [Online]. Available: https://www.srsran.com/

[14] Open5GS, "Open source project of 5gc and epc (release-16)." [Online]. Available: https://open5gs.org/

[15] M. Baker-Harvey, "Google compute engine uses live migration technology to service infrastructure without application downtime." [Online]. Available: https://cloudplatform.googleblog.com/2015/03/Google-Compute-Engine-uses-Live-Migration-technology-to-service-infrastructure-without-application-downtime.html