Optimizing 5G VPN+ Transport Networks with Vector Packet Processing and FPGA Cryptographic Offloading

Bruno Dzogovic[1], Bernardo Santos[1], Boning Feng[1], Van Thuan Do[2,1], Niels Jacot[2], Thanh Van Do[3,1]

[1] Oslo Metropolitan Univeristy, Pilestredet 35, 0167 Oslo, Norway
[2] Wolffia AS, Haugerudvn. 40, 0673 Oslo, Norway
[3] Telenor ASA. Snarøyveien 30 1331 Fornebu, Norway
{bruno.dzogovic, bersan, boning.feng}@oslomet.no
{vt.do,n.jacot}@wolffia.net
thanh-van.do@telenor.com

Abstract. Network slicing is the crucial prerogative that allows end users and industries to thrive from 5G infrastructures, however, such a logical network component can deteriorate from security vulnerabilities that prevail within cloud environments and datacenters. The Quality of Experience in 5G is a metric that takes into consideration sets of factors, which play role in the definition of the end-to-end performance, which is indeed latency, packet processing, utilization of legacy protocols, old hardware, encryption, non-optimized network topologies, routing problems and multitude of other aspects. This research sheds light on the inherent networking stack performance issues that translate into 5G environments, in a use-case where encrypted VPN tunneling is used to secure the backhaul transport network between the 4G/5G cores and the frontend networks.

Keywords: 5G, Enhanced VPN+, Vector Packet Processing, FPGA SoC.

1 Introduction

5G engrosses various sectors of the modern society, including healthcare, transport, smart infrastructure, industrial and other spheres. One of the most important environs that 5G aims to support is the Massive Internet of Things (MIoT). In healthcare, IoT enables providers to assist patients in various ways by providing wearables, sensors and implants for various medical conditions and monitoring [1] etc. The devices require stringent security measures for preserving the confidentiality and data privacy of patient information. Nevertheless, the healthcare sectors are constantly targeted by cyber criminals and additional considerations from that aspect are desirable to prevent private information leakage and mitigate repercussions [2]. The information retained about healthcare services and patients is of substantially sensitive nature and therein the need for additional protection and threat mitigation. An example of eccentric cyber-attacks during the COVID-19 pandemic from 2020, indicates that the adversaries are utilizing APT (Advanced Persistent Threat) attacks and is attributed to the possibility that the intention is to compromise and exfiltrate research data regarding COVID-19 vaccines

This is a post-peer-review, pre-copyedit version of a conference proceeding published in MobiWIS 2021: Mobile Web and Intelligent Information Systems, 17th International Conference, Proceedings, which is part of the Lecture Notes in Computer Science book series (volume 12814). The final authenticated version is available online at DOI: https://doi.org/10.1007/978-3-030-83164-6_7

[2]. The increase of cyber threats on healthcare institutions in Central Europe has increased exponentially in November 2020, namely by 145%, which is an astoundingly elevated figure [3]. The enhanced quality of service is what 5G delivers to the smart healthcare and in parallel security threats that it is accompanied with [4]. Most of the research is focused on the stated Quality of Service and performance, but the security remains a highly misjudged domain, therein the upsurge of cyber-attacks on healthcare institutions in the last two years.

Adversaries are continuously developing new practices to enhance their attack vectors and exploit various vulnerabilities for fulfilling their malicious goals. One rather traditional technique to secure a communication is encrypted tunneling, namely employing VPNs to support the communication between endpoints. In this paper, we examine a previous work on a 5G infrastructure for IoT in healthcare and address a performance issue that arises as a result to the encryption in the transport network between the 5G network core (5GC) and the Centralized Unit (CU) in a cloud environment. This is achieved by combining improvements in the Linux kernel for packet processing and introducing an additional FPGA hardware to offload the cryptographic operations from the 5G core compute node. The improvements of performance in 5G VPN+ network slice encrypted with AES-256 are thereby substantial.

1.1 Motivation and Problem Statement

Smart healthcare involves medical devices that are issued to patients and communicate through a 5G network. In simple terms, the transport network that connects the Core Networks of the service providers with contiguous Centralized Units for baseband processing is as secure as the infrastructure itself, which signifies that threats can emerge when adversaries are able to access the network through any means (either from the internet or on-premises). 5G establishes the concept of network slicing that separates the tenants in the network based on their use case. A typical method to resolve the communication and restrict access to the corresponding actors is by using policy-based networking and selective routing of traffic, VLAN segmentation, compute nodes isolation as well as firewalls and traffic filtering. These techniques are incorporated at the orchestration layer of the SDN controller and the same are insufficient to prevent adversaries from commencing various attacks. To strengthen the security of the transport network in 5G between the CU and Core Networks, we have established an infrastructure that allows provisioning of a custom network slice, crafted for healthcare purposes, and utilizes an enhanced VPN+ tunneling to secure the communication. Nevertheless, the symmetric encryption in the tunnel adversely affects the performance of the communication at scale and requires optimizations to provide satisfactory quality of experience for the increasing number of MIoT devices and the end users in healthcare.

This paper begins with an introduction and explaining background work in the related field. Further, a brief explanation of the methodology and implementation follow and correspondingly, the results of the experiments are represented. To finalize, the paper discusses certain limitations and advantages of the proposed approach and concludes on the lessons learned.

2 Background and related work

2.1 5G architecture

The 5G system consists of User Equipment (UE), Access Network (next-generation Node-B, or gNB) and Next-Generation Core Network (5GC). By combining Software-Defined Networking (SDN) and open Application Programming Interfaces (APIs), users can have virtual Network Functions (vNFs) tailored and customized according to a specific scenario. Figure 1 represents an overview of the 5G architecture, which is separated into two main group functions: Control Plane (CP) core of 5GS and User Plane (UP) functions. The control-plane group has different elements defined in terms of Network Functions (NF). It is comprised of a common framework and offers services to other authorized NFs or users. This Service-Based Architecture (SBA) allows modularity, scalability, and reusability [5].

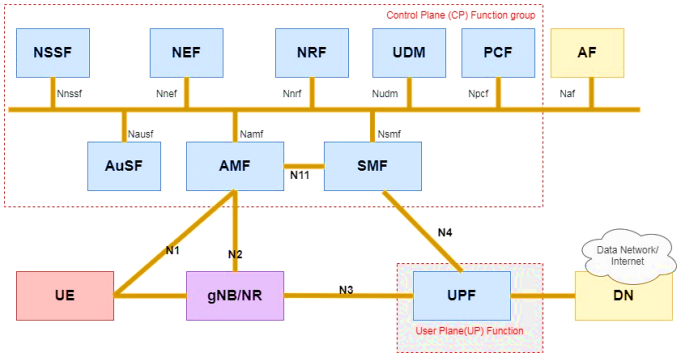

Figure 1. 5G Core Architecture

Network Slicing

The concept behind network slicing is based on logical network components, situated on a common physical infrastructure also dissected into virtual networks. The network softwareization concept [6] enables network slicing through software-based solutions. For that purpose, network slicing in 5G utilizes Software-Defined Networking (SDN), Network Function Virtualization (NFV) in clouds and the Edge for the realization of slices over the same physical infrastructure of the service providers. Each slice is controlled independently and can scale according to requirements.

Compared to 4G LTE, 5G replicates the function of physical hardware in form of software. SDNs can thus be easily adapted to serve the needs of backends (such as SD-WAN) as well as local deployments for customizing network slices in 5G. In a previous work at the Secure 5G4IoT lab by the Oslo Metropolitan University, we underlined the details on network slicing using a slicing controller and an SDN controller to provide virtual network functions for connectivity to the corresponding network slice [7,8]. VNFs can be implemented in different ways. One common method is to separate a physical network interface into manifold virtual functions, assign these virtual

networking endpoints to a VNF/CNF (Container Network Function) and implement policy and routing via the SDN controller through these terminals for the network slice. Figure 2 represents a working example of virtual network function provisioning for 5G cloud radio access network (C-RAN) with a single Centralized Unit (CU) in OpenStack cloud. Another traditional method for separating vNFs is by using VLAN (Virtual LAN) on Layer-2. According to the working example in the Secure 5G4IoT Lab at the Oslo Metropolitan University, the virtual functions are delivered using SR-IOV (Single Root – Input Output Virtualization) of the network. The 5G core functions are containerized and use the Kuryr plugin to interface the containers with the OpenStack Neutron [9].

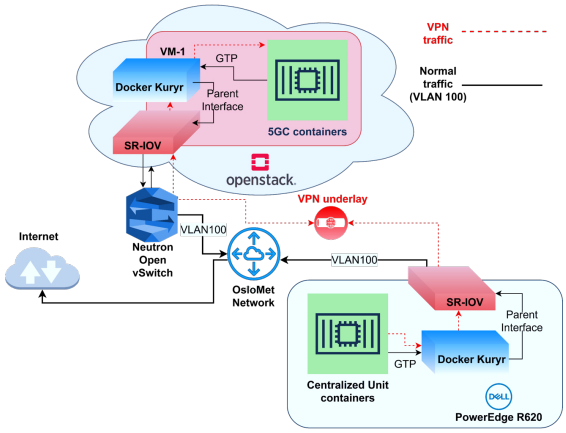

Figure 2. VLAN segmentation using SR-IOV and VPN instance in the transport network between the Centralized Unit containers and the 5G Cloud Core Network

High-performance vector packet processing stack

To handle low-level packet processing issues the VPP (Vector Packet Processing) is used. By enabling a software routing using the VPP driver, we deliver a VPN underlying network for securing the communication between the C-RAN network and the core networks in the cloud. VPP does not process packets on sequential basis as is the case with the scalar model, but instead it processes the entire vector of packets through a graph node before proceeding to the next graph node. There is a support for hardware acceleration through plugins for offloading the packet processing functions to external hardware [10]. VPP uses vector processing as opposed to scalar processing, treats more than one packet at a time. One of the benefits of the vector approach is that it fixes the I-cache thrashing problem. It also mitigates the dependent read latency problem (prefetching eliminates latency). This approach fixes the issues related to stack depth / D-cache misses on stack addresses by improving the cycle of capturing all available packets from the device RX ring, forming a vector that consists of packet indices in RX order, running the packets through a directed graph of nodes, and returning to the RX ring. As processing of packets continues, the circuit time reaches a stable equilibrium based on the offered load. As the vector size increases, processing cost per packet decreases because the I-cache misses over a larger N are being amortized [10]. VPP is

integrated in OpenStack using the Neutron Modular Layer-2 (ML2-VPP) mechanical driver, as shown in Figure 3.

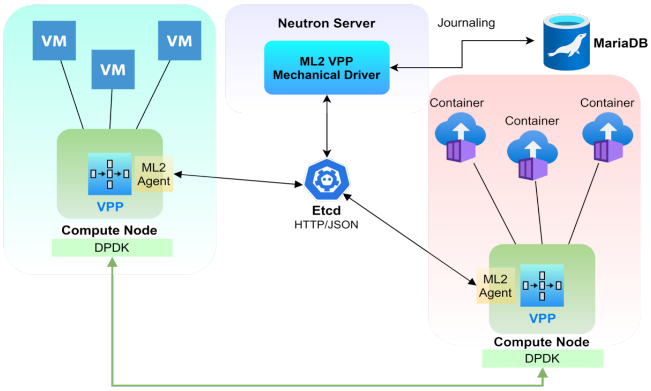

Figure 3. VPP native integration with the OpenStack Neutron Modular Layer 2 (ML2) driver

For the production-scale cloud, the VPP ML2 agents are deployed via the OpenDaylight SDN controller in OpenStack, which are referred to as *"Honeycomb data-plane agents"* where Kubernetes and Ansible are used for orchestration [11,12,13]. VPP is shown to massively improve performance in relatively low-power computing environments [14,15,16].

2.2 Enhanced VPN+ and cryptographic functions offloading to FPGA SoC

One way to enable transport network VPN connection is to bypass application layer VPNs and perform tunneling at transport network layers. In the style of VPN as a Service (VPNaaS) and On-demand network slice provisioning NSaaS (Network Slice as a Service), tenants can request a network slice with VPN tunneling integrated as a custom vNF. The ACTN framework (Abstraction and Control of Traffic-Engineered Networks) is thereby introduced [17] to define the ability for customers to deploy private networks without the understanding of the backend. ACTN defines the Virtual network slicing service function chaining (SFC) model. This same model serves for provisioning 5G transport networks by utilizing the CNC (Customer Network Controller), MDSC (Multi-Domain Service Coordinator) and PNC (Provisioning Network Controller) [18]. CNC is responsible for 5G 3GPP access-network communication with the underlying network of the 5G infrastructure and is also known as Traffic Provisioning Manager (TPM) [19]. The TPM functions as a CNC from ACTN reference point of view and can be deployed in a carrier network as shown in Figure 4, TPM can be deployed in Mobile Network A - Domain 1 and Domain 2, while the CMIs interfaces are connected to SDN controllers (in this case we refer to the PNC from the ACTN framework reference) [19].

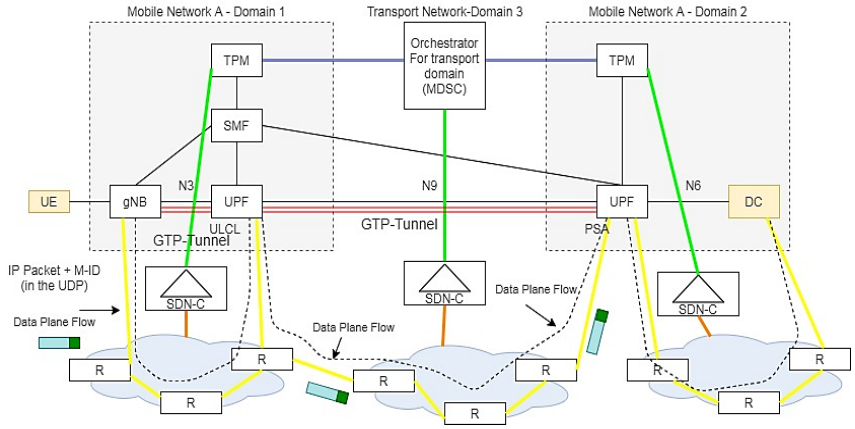

Figure 4. Enhanced 5G Transport Network Architecture in a multi-region cloud model [19]

For the current experiments, we employ a hybrid model, in which the SDN controller imports the information of the underlay network through BGP-LS (BGP-Link State) [20] via RestCONF APIs, and the traffic engineering (TE) information is collected and shared to external components of the network. Therefore, a Traffic Engineering Database (TED) is formed [21], which stores information about the TE information for dynamic Quality of Service parameters regulation as in the case with MPLS and GMPLS networks [20]. This way, the tenants can decide to instantiate an enhanced VPN+ at the network underlay, while avoiding overlay application-layer overheads at scale.

Field Programmable Gate Arrays are used in clouds for various functions. A SoC (Silicon on a Chip) architecture allows the FPGA to be programmable remotely and without direct access. This combination usually is followed by an ARM architecture device tightly integrated with the FPGA, which allows interfacing to the FPGA fabrics. The FPGA can then be used for various applications that require specific and customized computational properties and are task intensive for the general x86 architecture CPUs [22,23], which is programmed to perform cryptographic operations for AES-256 in the VPN tunnel. The FPGA is programmed in Verilog to access the memory directly and interact with the VPP kernel module, which opens a potential to scale the FPGA fabrics into cluster of multiple FPGA SoC units. The cluster can then be served as a service to the specific network slicing virtual functions that request it.

3 Methodology and implementation

The Figure 5 represents a 4G and 5G infrastructure deployed at the Oslo Metropolitan University with three different slices: a 5G New-Radio, 4G standard LTE access and an IoT network slice with IEEE 802.11 Wi-Fi access. The core networks have their own SR-IOV endpoints and is considered a separate virtual cluster sharing the same cloud. The experimentation methodology is based on network testing performed with the iperf tool to measure traffic performance at scale. Before conducting the evaluation of the throughput in the network, the Maximum Transmitted Unit (MTU) is adjusted to 9000

from the default 1500 value to minimize fragmentation incurred variance in the network flows. This is because the traffic between the Centralized Unit and the 5G Core Network is encapsulated to support the transmission of GTP traffic in an extended IP header. With a MTU value of 1500, the traffic between the UE (User Equipment) and the Internet will experience packet drop, jitter and additive error, therefore it is adjusted accordingly on all interfaces, including the SR-IOV physical functions, virtual functions, as well as the container-plane network.

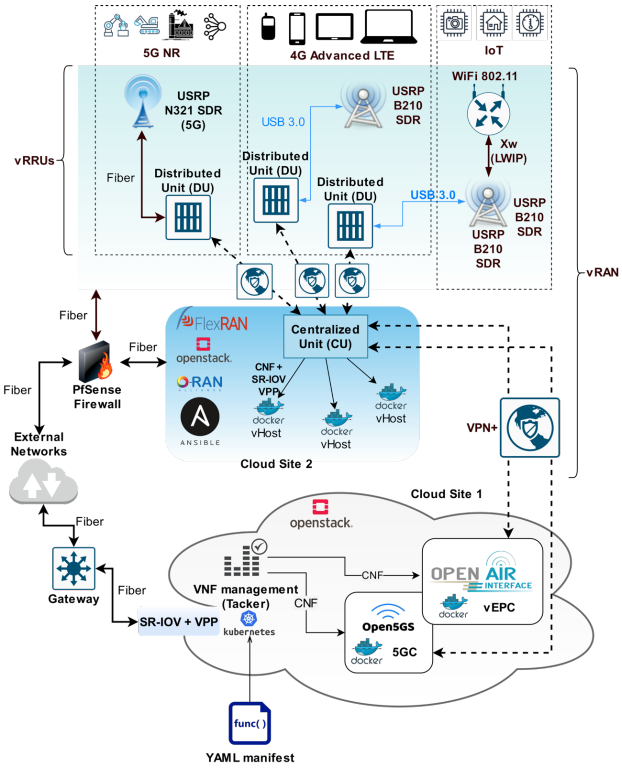

Figure 5. 4G and 5G hybrid infrastructure at the Oslo Metropolitan University

The compute nodes are Dell R620 servers running Linux Ubuntu server 18.04 operating with low-latency kernel version 14.18.0-25. The Core Network vNFs require a stable environment to operate and thus any fluctuations incurred by the CPU on a hardware level in terms of frequency, voltage, heat-related inconsistences and multitenancy can cause unpredictable result. To minimize experimental error, CPU hyperthreading is disabled as well as power states (C-states, P-states) and the unit overclocked to fixed 3.0 GHz. For testing the connectivity between the Centralized Unit and the Core Networks, network overlays are avoided to eliminate overhead and perform the control experiments directly on the 10Gbps optical network fabrics between the physical compute nodes. This experimental data is stored and used to compare to the performance impact virtualization can have on the nodes in terms of bandwidth and CPU resources

utilization combined. Further, we proceed with tests on the VPN+ tunneling in between the compute nodes, which are followed by tests on the VPN+ endpoints on virtualization layer between the Centralized Unit containers and the Core Network containers. This will shed light on the difference between the impact of hardware networking compared to virtualization with SR-IOV and direct VPN tunneling compared to virtualization-layer VPN+ tunneling.

As a final realignment, we change the generic Linux networking kernel modules with Vector Packet Processing and compare these results to the previously obtained data. That will represent the real status of the performance improvements that can be expected from utilizing vector processing compared to scalar processing. Last but not least, the FPGA SoC is introduced at the VPN+ server. Each test is performed in two stages: single-stream network testing and multiple-stream tests in parallel. The latter exaggerates the traffic conditions and simulates a realistic scenario where the transport network is saturated with traffic.

3.1 Evaluation

The obtained data is classified as follows:

- Scenario A: Hardware-level testing of the compute nodes at the optical network fabrics
- Scenario B: Virtualization-plane tests via the corresponding SR-IOV virtual functions translated in the containers that host the Centralized Unit and the Core Networks
- Scenario A1: VPN+ connectivity between the two compute nodes directly
- Scenario B1: Virtualization-plane tests of the VPN+ connectivity through the same SR-IOV virtual functions
- Scenario A2: VPN+ connectivity between the two compute nodes directly, with Vector Packet Processing Linux kernel module
- Scenario B2: Virtualization-plane tests of the VPN+ connectivity with Vector Packet Processing Linux kernel module
- Scenario A3: VPN+ connectivity between the two compute nodes directly with Vector Packet Processing Linux kernel module and FPGA SoC cryptographic offloading
- Scenario B3: Virtualization-plane tests of the VPN+ connectivity with Vector Packet Processing Linux kernel module and FPGA SoC cryptographic offloading

The A and B scenarios are the experimental control group for comparison with the further test scenarios. This will shed light on the performance detriment that is inflicted on the link. For that purpose, the network flows are measured together with CPU utilization and compared. Correspondingly, the A1 and B2 scenarios test the VPN+ connectivity and the impact it has on the deployment compared to the default state from the A and B scenarios. In the A2 and B2 scenarios, the VPP Linux kernel module is introduced, which will represent the mean performance gain that is otherwise lost due to the packet processing issues from the default scalar approach. Finally, the scenarios A3 and B3 implement an FPGA SoC for offloading the encryption from the CPU of the

compute node. These tests are performed only in multiple-stream examinations, in order to assess the performance improvement at scale (in worst-case scenarios).

The traffic of the single-stream and multiple-stream tests is adjusted in such a manner that respects a constant threshold under the total maximum capacity of the node's performance. The experiments do not examine latency-related issues.

4 Results

The results from the experiments are obtained using the iPerf3 network testing tool and the following arguments are passed. The time of execution is 5 minutes (300 seconds), with an interval of transmission each 1 second. The TCP buffer size is set as a constant to 32 Megabytes and the test runs as a client-server model with the server being executed at the 10.0.0.1 host, which is the Core Network and the 10.0.0.2 is the Centralized Unit host. Results are summarized in Table 1 and Table 2, denoting CPU resource utilization and maximum bandwidth, correspondingly.

Table 1. Summary of the CPU utilization at the 5GC core side and the CU side. The values represent total CPU utilization, divided between user resources and system namespace services. Each scenario is characterized with single-stream and multi-stream test results.

Scenario	CPU utilization (%) 5GC_total	CPU utilization (%) 5GC_user	CPU utilization (%) 5GC_system	CPU utilization (%) CU_total	CPU utilization (%) CU_user	CPU utilization (%) CU_system
A / S-S	61.9658	8.23386	53.7319	35.983	5.03469	30.9483
A / M-S	95.9105	18.9873	76.9232	41.7224	8.2758	33.4466
	CPU utilization (%) 5GC-SRIOV_total	**CPU utilization (%) 5GC-SRIOV_user**	**CPU utilization (%) 5GC-SRIOV_system**	**CPU utilization (%) CU-SRIOV_total**	**CPU utilization (%) CU-SRIOV_user**	**CPU utilization (%) CU-SRIOV_system**
B / S-S	63.5708	8.62531	54.9455	21.4606	3.48127	17.9793
B / M-S	96.2447	20.1134	76.1313	27.36	4.43466	22.9254
	CPU utilization (%) 5GC-VPN_total	**CPU utilization (%) 5GC-VPN_user**	**CPU utilization (%) 5GC-VPN_system**	**CPU utilization (%) CU-VPN_total**	**CPU utilization (%) CU-VPN_user**	**CPU utilization (%) CU-VPN_system**
A1 / S-S	37.1798	10.0446	27.1352	3.54544	0.36947	3.17597
A1 / M-S	44.2623	8.84595	35.4163	4.68069	0.944212	3.73648
B1 / S-S	36.9928	10.078	26.9148	1.89289	0.22533	1.66756
B1 / M-S	44.0171	8.80109	35.216	3.11648	0.811096	2.30538
	CPU utilization (%) 5GC-VPN-TN_only	**CPU utilization (%) 5GC-VPN-TN_user**	**CPU utilization (%) 5GC-VPN-TN_system**	**CPU utilization (%) CU-VPN-TN_only**	**CPU utilization (%) CU-VPN-TN_user**	**CPU utilization (%) CU-VPN-TN_system**
A2 / M-S	100.0	49.257	50.743	7.121	0.782	2.51
B2 / M-S	100.0	48.290	51.71	8.234	0.913	2.912
A3 / M-S	59.34	2.173	23.439	7.856	0.828	2.87
B3 / M-S	63.17	3.012	27.126	9.162	0.711	4.451

Table 2. Summary of the bandwidth tests at the sender and receiver side. The results include the total retransmissions and duration of the tests in seconds. Each scenario is represented with single-stream and multi-stream tests

Scenario	Send Duration (s)	Sent Data (GB)	Send Speed (Gbps)	Retran. (total)	Rec. Duration (s)	Rec. Data (GB)	Rec. Speed (Gbps)
A / S-S	300.036	371.04267	9.89328	0	300.036	371.04249	9.89327
A / M-S	300.015	371.16965	9.89736	1	300.015	371.16546	9.89725
B / S-S	300.039	371.18716	9.89704	0	300.039	371.18684	9.89703
B / M-S	300.021	371.17811	9.89739	22	300.021	371.17443	9.89729
A1 / S-S	300.037	19.591886	0.522386	163311	300.037	19.591432	0.522374
A1 / M-S	300.021	20.563469	0.54832	347139	300.021	20.559070	0.548203
B1 / S-S	300.038	19.921978	0.531218	170510	300.038	19.922784	0.531207
B1 / M-S	300.022	21.049824	0.561288	348553	300.022	21.045380	0.56117
A2 / M-S	300.021	223.00132	5.347821	3	300.021	223.00412	5.347733
B2 / M-S	300.023	219.34128	5.330129	2	300.023	219.34115	5.331604
A3 / M-S	300.030	360.21275	8.928763	0	300.030	360.21478	8.972871
B3 / M-S	300.021	343.33713	8.873988	0	300.021	343.33711	8.873569

To determine the impact of the VPN on the backhaul network, we establish causal relationships between the hardware-level tests, virtualization layer tests and compare the results with the improvement from utilizing the VPP module. Finally, the results are also compared with the inclusion of the FPGA SoC offloading. For that purpose, the multiple linear regression analysis is used.

The value of the correlation coefficient ("multiple R") is 0.850086999, indicating a good relationship between the CPU utilization, bandwidth, and the number of occurring retransmissions. The significance F-test of the null-hypothesis is less than 0.05 (i.e., S-F=0.003116316). Figure 6 and Figure 7 show the overall distribution of CPU utilization compared to the speed per each scenario. During the hardware-level tests, the overall utilization of the CPU in the node was 41.7224%, whereas during the SR-IOV parallel stream tests, the total usage amounts for 27.36%. Nevertheless, there is an unavoidable variation in the system processes at the different times of testing, for which if we calculate the offset, we get 33.4466% during hardware-level tests and 22.9254% during SR-IOV tests, results in $\Delta_{SR-IOV(m)} = 10.5212\%$. This value is subtracted from the overall performance in order to isolate the effects of virtualization during the multiple stream tests, that is 41.7224% during the hardware-level tests and 27.36% during the SR-IOV tests. The difference of 14.3624% subtracts the system resources difference by 10.5212%, obtaining 3.8412% impact on the virtualization plane during the SR-IOV tests with multiple streams. The overhead increases when the traffic scales over the same VNF (Figure 8 and Figure 9).

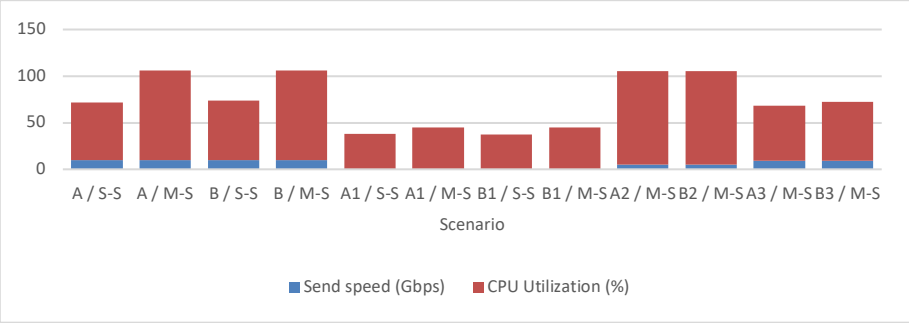

Figure 6. Distribution of the multiple linear regression analysis of the correlation between relative bandwidth (in Gbps) and CPU utilization in percentage (%) by scenario

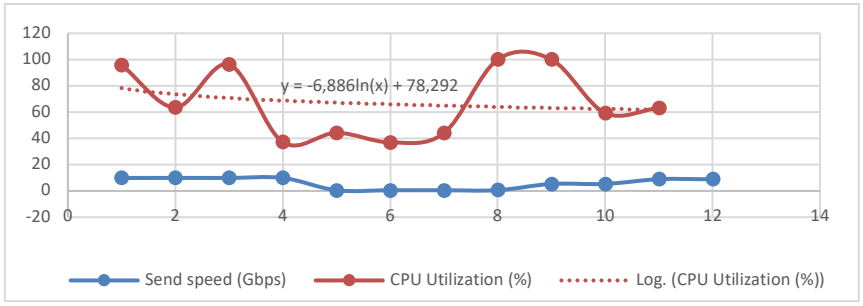

Figure 7. Multiple linear regression analysis of the relative bandwidth in Gbps and total CPU utilization in percentage (logarithmic distribution)

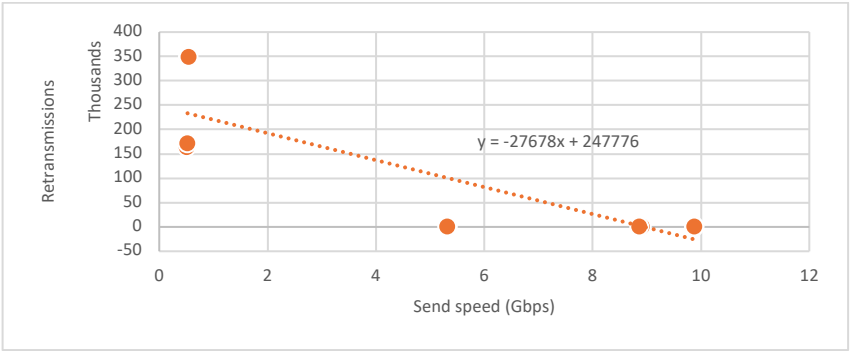

Figure 8. Correlation between the relative bandwidth and the number of retransmissions

Without provisioning additional VNFs, it is likely that the impact would be more than linear over a threshold of difference between the kernel's scalar packet processing capability and the absolute number of scaled VNF functions for the SR-IOV drivers, accounting for the traffic generated also in the 5G network on top of the diverse network slices. Furthermore, as the number of retransmissions increase, the level of computational resources required for retaining maximal bandwidth will increase,

adding on the bottleneck of the operating system's kernel at the physical nodes and is represented through the CPU underutilization. This is further amplified through an encrypted VPN tunnel, accentuating the I-cache thrashing problem with the scalar packet processing. The VPP allows the CPU to have time-series workloads allocated for submitting each encrypted packet to the network interface, thereby increasing the bandwidth of the tunnel and showing 100% CPU utilization. Additional rectification is achieved when the FPGA fabric is introduced, further restoring the traffic performance (~8.87 Gbps).

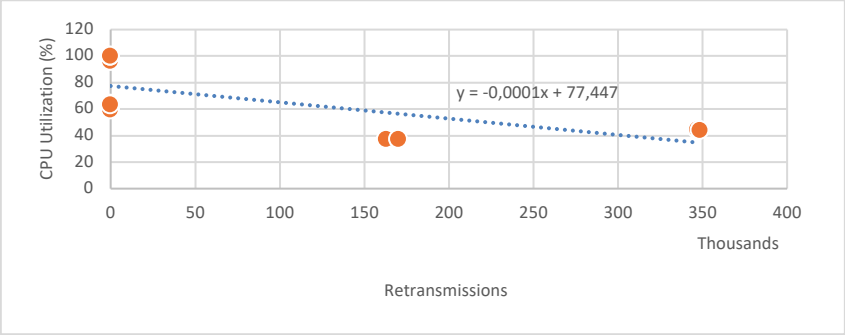

Figure 9. Dependency between the number of retransmissions and total CPU utilization (system namespace and user running the 5GC core network and VPN)

5 Discussion

One of the main hindrances to performance loss in a software-defined 5G backhaul networking is the utilization of legacy hardware in datacenters. The modern workloads require improved hardware, especially CPU processing units and in general new architectures in order to run with higher efficiency. Datacenters are usually equipped with specialized processing hardware for particular workloads, such as GPUs or FPGAs, which at scale can have detrimental impact on the execution of required tasks. In the case with VPN, the technique that combines Vector Packet Processing and FPGA offloading can be scaled to accommodate bigger infrastructures and negate the requirement of horizontally scaling the same.

Using FPGA SoC requires tedious implementation methodology and incurs latency penalties on the network. Despite that this work is not focused on examining the latency related impacts in the network, an external SoC managed FPGA forwards packets from the SDN controller that directs the network flows through the FPGA and back to the CPU of the compute nodes. A better performing FPGA layer combined with tight integration with the SDN networking stack of the backhaul, can resolve the said issues.

6 Conclusion

Utilizing a vNF passthrough, minimizes the impact of virtualization on the performance of 5G backhaul transport networks at scale. A rather simpler approach that attains flat

infrastructure, easier to automate and self-organize, the virtualization layer can be supplemented with vector packet processing to annul some ramifications that arise due to inherent networking kernel limitations in Linux. This implication becomes intensified when the transport network is tunneled via VPN, which can encrypt the communication with various algorithms. The symmetric encryption needs to scramble every packet that traverses the transport network endpoints, which is highly taxing for the CPU and the available resources at the compute nodes in the cloud. The communication becomes severely bottlenecked and the overall potential of the 5G transport network substantially diminished. A prodigious and cost-efficient solution is to integrate FPGA fabrics to handle the encryption of the tunneling and use vector packet processing to increase the flow rates and minimize packet retransmissions, which can drastically improve the performance.

References

1. M. Zhong, Y. Yang, H. Yao, X. Fu, O. A. Dobre and O. Postolache, 5G and IoT: Towards a new era of communications and measurements. IEEE Instrumentation & Measurement Magazine, vol. 22, no. 6, pp. 18-26, (2019), doi: 10.1109/MIM.2019.8917899.
2. Ravie Lakshmanan: Healthcare Industry Witnessed 45% Spike in Cyber Attacks Since November 2020. The Hacker News, URL: https://thehackernews.com/2021/01/healthcare-industry-witnessed-45-spike.html, last accessed 2021/03/26.
3. Muthuppalaniappan, Menaka, and Kerrie Stevenson: Healthcare cyber-attacks and the COVID-19 pandemic- an urgent threat to global health. International journal for quality in healthcare: journal of the International Society for Quality in Health Care vol. 33,1 (2021). Doi:10.1093/intqhc/mzaa117.
4. A. Ahad, M. Tahir and K. A. Yau: 5G-Based Smart Healthcare Network: Architecture, Taxonomy, Challenges and Future Research Directions. IEEE Access, vol. 7, pp. 100747-100762. (2019). Doi: 10.1109/ACCESS.2019.2930628.
5. M. Q. Khan: Signaling Storm Problems in 3GPP Mobile Broadband Networks, Causes and Possible Solutions - A Review. 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), pp. 183-188. Southend, UK. (2018). Doi: 10.1109/iCCECOME.2018.8658708.
6. I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini and H. Flinck: Network Slicing and Softwarization - A Survey on Principles, Enabling Technologies, and Solutions. IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2429-2453. (2018). Doi: 10.1109/COMST.2018.2815638.
7. B. Dzogovic, T. van Do, B. Santos, D. Van Thuan, B. Feng and N. Jacot: Thunderbolt-3 Backbone for Augmented 5G Network Slicing in Cloud-Radio Access Networks. 2019 IEEE 2nd 5G World Forum (5GWF), Dresden, Germany, pp. 415-420. (2019). Doi: 10.1109/5GWF.2019.8911710.
8. F. Z. Yousaf, M. Bredel, S. Schaller and F. Schneider: NFV and SDN—Key Technology Enablers for 5G Networks. IEEE Journal on Selected Areas in Communications, vol. 35, no. 11, pp. 2468-2478. (2017). Doi: 10.1109/JSAC.2017.2760418.
9. OpenStack cloud software: Kuryr plugin official documentation, URL: https://docs.openstack.org/kuryr/latest/ last accessed 2021/03/29.
10. Fd.io: Vector Packet Processing. URL: https://fd.io/vppproject/vpptech/ last accessed 2021/01/16.

11. RedHat OpenShift: About Single Root I/O Virtualization (SR-IOV) hardware networks. URL: https://docs.openshift.com/container-platform/4.4/networking/hardware_networks/about-sriov.html last accessed 2021/01/16.
12. RedHat: Ansible automation tool. URL: https://docs.ansible.com/ansible/latest/index.html last accessed 2021/01/16.
13. Fd.io: Vector Packet Processing – RDMA ibverb Ethernet driver. Version 19.08-27-gf4dcae4. URL: https://docs.fd.io/vpp/19.08/rdma_doc.html last accessed 2021/01/16.
14. Nikolai Pitaev, Matthias Falkner, Aris Leivadeas, and Ioannis Lambadaris: Characterizing the Performance of Concurrent Virtualized Network Functions with OVS-DPDK, FD.IO VPP and SR-IOV. Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE '18), pp. 285-292. Association for Computing Machinery, New York, NY, USA, (2018). Doi: 10.1145/3184407.3184437.
15. Liren Miao, Hongchao Hu, and Guozhen Cheng: The Design and Implementation of a Dynamic IP defense System Accelerated by Vector Packet Processing. Proceedings of the International Conference on Industrial Control Network and System Engineering Research (ICNSER2019), pp. 64-69. Association for Computing Machinery, New York, NY, USA, (2019). Doi: 10.1145/3333581.3333588.
16. J. Yan, T. Li, S. Wang, G. Lv and Z. Sun: Demonstration of Path-Based Packet Batcher for Accelerating Vectorized Packet Processing. 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 1-3, Hong Kong, China (2018). Doi: 10.1109/SAHCN.2018.8397154.
17. IETF: RFC 8453 – Framework for Abstraction and Control of Traffic Engineered Networks (ACTN). URL: https://datatracker.ietf.org/doc/rfc8453/?include_text=1 last accessed 2021/03/27.
18. Metro-haul project: What is ACTN framework? URL: https://metro-haul.eu/2018/08/30virtu/what-is-actn/ last accessed 2021/27/03.
19. Y. Lee, J. K: Applicability of ACTN to Support 5G Transport, TEAS Working Group IETF (2019). URL: https://tools.ietf.org/pdf/draft-lee-teas-actn-5g-transport-00.pdf.
20. V. Roux: Path Computation Element (PCE) Communication Protocol (PCEP). IETF, (2008). URL: https://tools.ietf.org/html/rfc5440 last accessed 2021/27/03.
21. IETF: Traffic Engineering Database Management Informormation Base in Support of MPLS-TE/GMPLS. URL: https://tools.ietf.org/html/rfc6825 last accessed 2021/27/03.
22. Ke Zhang, Yisong Chang, Mingyu Chen, Yungang Bao, and Zhiwei Xu: Engaging Heterogeneous FPGAs in the Cloud. Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '19). Association for Computing Machinery, New York, NY, USA, (2019). Doi: 10.1145/3289602.3294001.
23. Xilinx Zynq-7000 SoC: Datasheet – Overview. Version 1.11.1. URL: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf last accessed 2021/03/27.

Acknowledgement

This paper is a result of the H2020 Concordia project (https://www.concordia-h2020.eu) which has received funding from the EU H2020 programme under grant agreement No 830927. The CONCORDIA consortium includes 23 partners from industry and other organizations such as Telenor, Telefonica, Telecom Italia, Ericsson, Siemens, Airbus, etc. and 23 partners from academia such as CODE, university of Twente, OsloMet, etc.