An In-Kernel Solution Based on XDP for 5G UPF: Design, Prototype and Performance Evaluation

Thiago A. Navarro do Amaral*, Raphael V. Rosa†, David F. Cruz Moura†, Christian E. Rothenberg†
School of Electrical and Computer Engineering (FEEC)
University of Campinas (UNICAMP)
Campinas, Brazil
*t159121@dac.unicamp.br, †{raphaelvrosa, dfcmoura, chesteve}@dca.fee.unicamp.br

***Abstract*—The edge computing infrastructure can scale from datacenters to single device. The well-known technology for fast packet processing is DPDK, which has outstanding performance regarding the throughput and latency. However, there are some drawbacks when the usage is done in the edge: *(i)* the polling mechanism for packet processing keeps the CPU exclusively occupied even if there is no traffic, leading to wasted resources; and *(ii)* DPDK interface becomes unavailable for the applications inside the host, so the integration between a non-DPDK application and a DPDK application becomes a hard task. In this paper, we propose an open-source in-kernel 5G UPF solution based on 3GPP Release 16 to be deployed in a restrictive environment like MEC, where MEC host and UPF are collocated with the Base Station, sharing the same computational and network resources. The solution leverages the eBPF/XDP, a novel Linux kernel technology for fast packet processing. We show it can scale and achieve 10 Mpps using only 60% of the CPU with 6 cores.**

***Index Terms*—5G networks, fast packet processing, XDP, eBPF, MEC, UPF.**

I. Introduction

Nowadays, three paradigms are changing the design of the network infrastructure: SDN [1], NFV [2], and cloud computing. These models make 5G networks more flexible, scalable, open, and programmable. Though, one of the main challenges remains meeting uRLLC (ultra-reliable low-latency communications) requirements. To meet these specific requirements, the MEC (Multi-Access Edge Computing) [3] was introduced to reduce this gap.

There are two key components in MEC: MEC host and UPF (User Plane Function). The former one is a general purpose computing facility that provides computing and storage resources to applications [3], while the latter is mainly responsible for handling user traffic to the appropriate DN (Data Network). These components could be collocated within the Base Station (edge), sharing the same network and computational resources. The edge computing infrastructure can scale from datacenters to single devices [4], depending on the use case requirements. So, because of this diversity, portable solutions for fast packet processing might be considered.

DPDK is a well-known fast packet processing technology. Although it has an outstanding performance regarding the throughput and latency [5], there are some drawbacks for a typical MEC deployment scenario: *(i)* DPDK support by the NICs; *(ii)* the polling mechanism for packet processing (DPDK Poll Mode Drivers) keeps the CPU exclusively occupied even if there is no traffic, leading to wasted resources; and *(iii)* DPDK interface becomes unavailable for the applications inside the host, so the integration between a non-DPDK application and a DPDK application becomes a hard task [6].

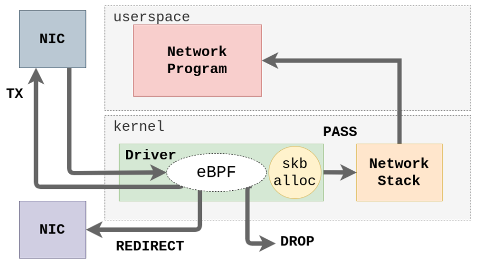


Fig. 1: Simplified XDP architecture diagram with the possible actions that can be applied to the received packet before the *socket buffer* allocation by the operating system. Source: adapted from [7].

In this paper, we propose an open-source[1] in-kernel solution based on 3GPP Release 16 as an alternative for 5G UPF (User Plane Function). The solution leverages the eBPF/XDP (eXpress Data Path), a novel built-in Linux kernel technology for fast packet processing, which has advantages over DPDK [6], i.e. *(i)* integrates cooperatively with the regular networking stack; *(ii)* does not require dedicating full CPU cores to packet processing. We show the proposed solution can scale and achieve 10Mpps using only 85% of the CPU with 6 cores.

The rest of the paper is organized as follows. The section II describes the in-kernel technology for fast packet processing (eBPF/XDP) and the 5G System (5GS) architecture. The section III elaborates the design and explains the key components used in the implementation. The section IV describes the setup and the test cases for the performance evaluation. The section V shows the results of the proposed solution and the analyses. The section VI describes the related works. The conclusions are given in the last section.

II. Background

A. eBPF

It is a general purpose engine that allows you to execute instructions in the Linux kernel itself [8]. It was built around

[1] https://github.com/navarrothiago/upf-bpf

two goals: (i) to have minimal overhead when mapping these instructions to native instructions and (ii) to be able to ensure that the program is safe at load time. This technology has been available since kernel version 3.14. The program can be hooked in specific areas (e.g. network stack) in the Linux kernel and its execution is triggered by an event (e.g. a received packet). The eBPF program portability to different Linux kernel versions has been addressed by the CO-RE (Compile Once, Run Everywhere) technology [9]. Our proposal is based on this technology in order to be compatible with the diversity of the edge computing infrastructure.

B. XDP

It is a hook point located in the reception chain of a network device driver before memory (i.e. socket buffer) allocation by the operating system. As a result, XDP enables fast packet processing within the Linux kernel itself. When the packet is received by the network device driver, the eBPF program is executed and various actions can be taken, such as dropping the packet, redirecting to another interface, or continuing to be processed in the protocol stack (see in Figure 1). The network device driver must support XDP to take full advantage of technology benefits. If not supported, the program can be run in generic mode, but with a degraded performance. More information about XDP is available at [10].

C. 5G Network Architecture

The architecture of 5G networks is represented in the Figure 2. The colored regions represent the 5G network core, called New Generation Core (NGC), where the green block represents the UP (User Plane) component, while the blue one stands for CP (Control Plane) component. One of the main changes regarding LTE networks is to provide a service-oriented architecture (SOA), i.e., composed of virtual network functions with well-defined interfaces. These functions can be performed on commodity hardware equipment and accessed through a communication protocol (e.g. HTTP). Furthermore, the 5G architecture follows the CUPS (Control and User Plane Separation) model. This paradigm enables the deployment of UP closer to applications and services, thus reducing latency and traffic in the core of these networks.

In the next sections, two components of the NGC will be detailed: SMF (Session Management Function) and UPF.

1) SMF: SMF is part of the 5G CP. One of the most relevant SMF functions is to manage the sessions used for data traffic between the UE (User Equipment) and the DN, namely Session PDU (Protocol Data Units). Each session is represented by a logical tunnel passing through the UE, RAN, and UPF. When a PDU session is established, a context is created in each of these components. A context contains a set of specific rules that will be applied in the data packet in order to guarantee QoS (Quality of Service) and to forward the packet to the next hop. In the case of UPF, this context is represented by the PFCP Session.

PFCP (Packet Forwarding Control Protocol) is the signaling protocol used for communication between SMF and UPF

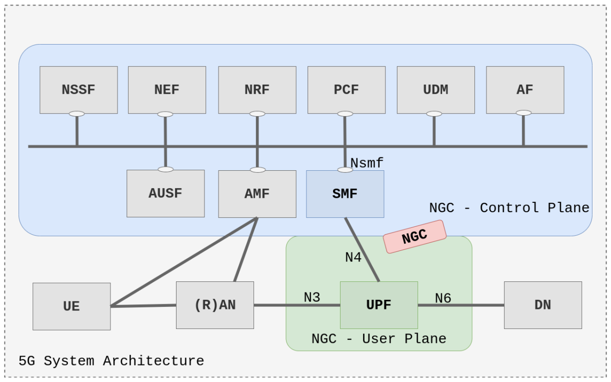


Fig. 2: Simplified 5G System architecture. Highlighted, the SMF and UPF components are addressed in the next sessions. The colored region represents the core of the 5G network.

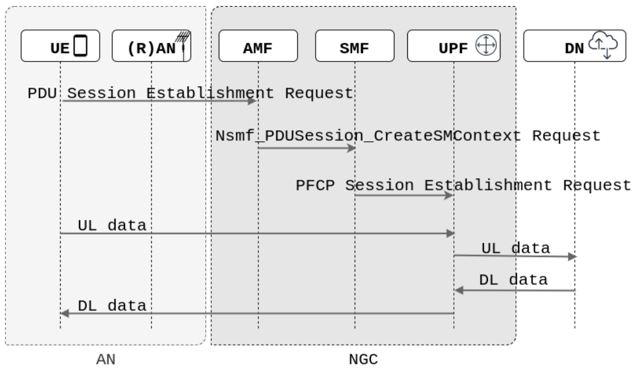

UE
(R)AN
AMF
SMF
UPF
DN
PDU Session Establishment Request
Nsmf_PDUSession_CreateSMContext Request
PFCP Session Establishment Request
UL data
UL data
DL data
DL data
AN
NGC

Fig. 3: Simplified procedure for requesting session establishment.

components. This communication involves PFCP session management procedures. The communication interface between the two components is represented by reference point N4 in Figure 2.

CP components communicate with SMF through the service provided by Nsmf_PDUSession [11]. This service involves PDU session creation, update, and removal procedures. One session establishment use case example is when the UE is registered at the core of the network and has data to send to the DN. If there is no session established, the procedure for requesting session establishment will be carried out as shown in the Figure 3. After performing the session establishment, the UE will be able to send/receive data in the sense uplink/downlink.

Compared with LTE networks, the SMF encompasses a set of functionalities from the MME (Mobility Management Entity) component within EPC (Evolved Packet Core). Additionally, the N4 reference point encompasses a set of functionalities defined for the Sxa/Sxb/Sxc interfaces, which are also used for signaling between the CP and the UP [12].

2) UPF: One of the main user plane components for NGC is the UPF. It is responsible for several functionalities related to user data traffic, such as forwarding and routing packets, applying rules to ensure quality of service, generating traffic

Rules	Interfaces			
	Sxa	Sxb	Sxc	N4
PDR	x	x	x	x
FAR	x	x	x	x
URR	x	x	x	x
QER	-	x	x	x
BAR	x	-	-	x
MAR	-	-	-	x

TABLE I: Relation between the signaling interfaces (5G and LTE) between the CP and UP and the packet rules applied.

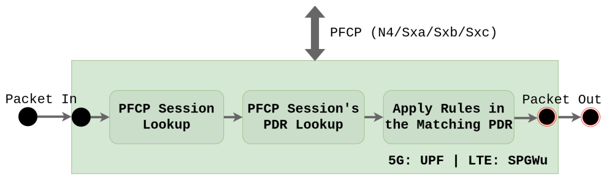


Fig. 4: Flow of packet processing in UP function defined by UPF and SPGWu, components of NGC and EPC respectively.

usage reports, and inspecting packets [11]. It works like a gateway between the RAN and the external DN (e.g. Internet, IP Multimedia System, local data network, etc). Regarding 4G/LTE networks, the UPF encompasses the functionalities of SGW (Servicing Gateway) and PGW (Packet Data Network Gateway)[2].

Data traffic takes place through a PDU session that is represented by contexts stored in the UE, RAN, and UPF components. In the case of the UPF, the context is represented by the PFCP Session, which contains the following rules: PDRs (Packet Detection Rules), FARs (Forwarding Action Rules), QERs (QoS Enforcement Rules), URRs (Usage Reporting Rules), BARs (Buffer Action Rules), and MARs (Multi-Access Rules).

Table I shows the relationship between the signaling interfaces (5G and LTE) used between the CP and UP of 5G and LTE networks and the rules applied in the packet. Our proposal focuses on the PDRs and FARs.

The flow for processing packets in the UP is represented in Figure 4. When the packet is received, its header is analyzed to find the PFCP Session to which the packet belongs. Once the session is found, the highest priority PDR is selected. The PDR contains all the FARs, QERs, URRs, BARs, MARs that are applied in the package in the next step. Finally, the packet is forwarded to the network interface as defined in the FAR. It is important to highlight that this flow of UP activities fits both 5G networks (UPF) and LTE networks (SPGWu) [12].

One of the protocols used to load user data packets is GTPu (GPRS Tunneling Protocol User Plane) [14]. The original

[2]In this work, the term SPGW will be used to refer to the combination of the SGW and PGW components as specified in [13]. SPGWu represents the component responsible for implementing the user plan functionalities for LTE networks.

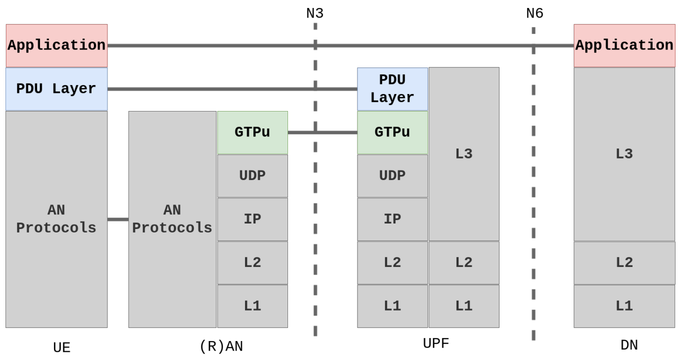


Fig. 5: Protocols stack used in communication between UP elements.

packet (IP datagram) is called a T-PDU (Transport Protocol Data Unit). When combined with the GTPu header, the packet is called a G-PDU (GTP encapsulated user Protocol Data Unit). The diagram representing the protocol stack used in the communication between the UP elements is represented in Figure 5.

III. Design and Implementation

This section describes our proposed architecture and implementation for fast packet processing using eBPF/XDP for user plane on the mobile core network (5G/LTE).

A. Features

Routing and forwarding packets are amongst the most relevant dataplane features in 5GS [11]. We have defined these actions as the core functionalities of the proposed solution, as part of the PFCP session context, which is created by sending PFCP Session Establishment Request message from the control plane (i.e. SMF for 5G network or SPGWc for LTE networks) to the dataplane (i.e. UPF for 5G and SPGWu for LTE). This request is sent when the UE has data to transmit to the network, for instance. The diagram presented in Figure 6 shows the PFCP session context data model of the proposed solution based on [12]. It does not support all IEs, but only the PDRs and FARs to forward packets in the core network user plane. The main features supported are:

i PFCP session management: create, read, update, and remove PFCP sessions, PDRs and FARs;
ii Fast packet processing for uplink and downlink user data traffic: classify and forward UDP and GTP traffic based on PDR and FAR, respectively.

B. Design

The library is divided in two layers: Management Layer, and Datapath Layer. The high level library design is shown in Figure 7.

1) Management Layer: It is an user space layer to manage PFCP sessions and eBPF programs lifecycle. A client can create/read/update/delete PFCP sessions through API. When a PFCP session establishment request message is received by the user plane component, the message is parsed and a call

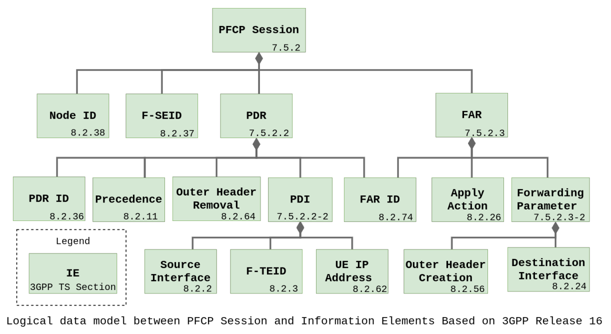


Fig. 6: The PFCP context data model with the IEs of the proposed solution.

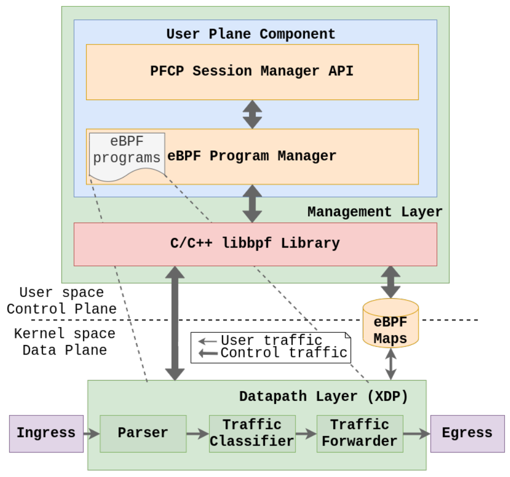


Fig. 7: High level design of the user plane library using eBPF/XDP.

is made to the library via PFCP Session Manager API. The PFCP Session Manager calls the eBPF Program Manager to load dynamically an eBPF bytecode, which represents the new PFCP session context, i.e., there is an eBPF program running in kernel space for each PFCP session. The program contains the eBPF maps used to store the PDRs and FARs IEs. All the communication between the user space and the kernel space is through the libbpf library [15], which is maintained by the Linux kernel source tree. The PFCP Session Manager parses the structures received to an eBPF map entries and updates the maps with them. The PFCP session context is created in Datapath Layer, where the user traffic will be handled.

2) *Datapath Layer:* It is a kernel space layer to process the user traffic inside the XDP. A service chain function was created with three main components: the Parser, the Traffic Classifier and the Traffic Forwarder. The Parser parses the ingress traffic to check if it is a uplink (GTPu) or a downlink (UDP) flow. If it is an uplink/downlink flow, the TEID (Traffic Endpoint Identifier)/UE (User Equipment) IP address key is used to get the PFCP session context. Then, the packet is passed to the PFCP session context represented by an eBPF program via tail calls. Here, the Traffic Classifier accesses the eBPF hash maps in order to find a PDR associated to

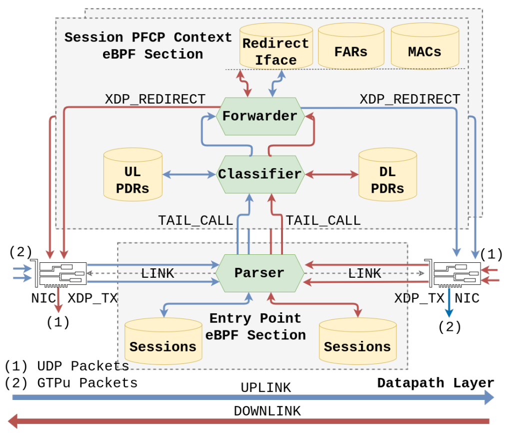


Fig. 8: Workflow in Datapath Layer.

the packet. If there is a PDR stored, the packet passes to the Forwarder. Finally, the Forwarder uses the FAR ID from the PDR to find the FAR, which is stored in an eBPF hash. The FAR contains the action (e.g. forward) that will be applied, the outer header creation and the destination interface. Besides, the Forwarder accesses other eBPF maps to check the MAC address of the next hop and the index of the destination interface where the packet will be redirected. The datapath workflow is shown in Figure 8.

C. Implementation

The components were developed in C++ (Management Layer) and restrict C (Datapath Layer). The source code is available through Apache-2.0 License. The PFCP Session Manager component UTs were implemented using the GoogleTest framework. Besides, a HTTP API was developed for end-to-end tests. This API simulates the PFCP control plane and wrappers the proposed solution. So, when a HTTP request is received, the JSON message is converted to the library structures and is passed through via function call to Session Manager API. It is important to highlight that the same structures used in the Management Layer are also used in the Datapath Layer and were based on the data model as described in Figure 6. The end-to-end performance evaluation was developed in python using Trex Stateless API [16] and will be described in the next section. All the tests were automated and the report of the test was generated automatically. Some of the challenges and limitations faced during the development of the eBPF/XDP program are well addressed in [17].

IV. Performance Evaluation

A. Setup

We have tested the proposed solution taking RFC2544-like measurements. The testbed is composed by two server Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz, 32GiB of the DRAM, 15M of L3 cache, 6 cores (hyper-threading disabled), dual-port 82599ES 10-Gigabit SFI/SFP+ NIC. Both machines have Ubuntu 20.04.02 LTS installed with Linux kernel v5.4.0-72-generic compiled with BTF flags enabled. One machine is used to generate user traffic with TRex Traffic Generator [18] and the other is the DUT (Device Under Test) where is deployed the proposed solution. The NICs were configured with toeplitz hash algorithm. The setup is shown in Figure 9.

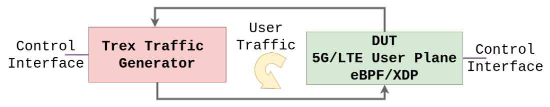


Fig. 9: Testbed setup.

B. Test case

We created a test case to evaluate the scalability and the workload when the solution achieves the maximum throughput for a specific traffic generation flow. Both uplink and downlink scenarios were tested. The tests consisted in *(i)* run Trex Traffic Generator server, *(ii)* run HTTP API (DUT), *(iii)* send a PFCP session establishment request via HTTP API to DUT, *(iv)* configure the number of Rx queue [19] in DUT, *(v)* generate traffic (GTP or UDP) using Trex Traffic Stateless API and, finally, *(vi)* collect workload per core (DUT) and throughput (Trex Traffic Generator) metrics. We create a JSON message to define the PFCP session establishment request to be sent to DUT. This message contains the PDRs (uplink and downlink) and FARs IEs and the static ARP table of the next hop. The PFCP session context message is shown in Figure 10.

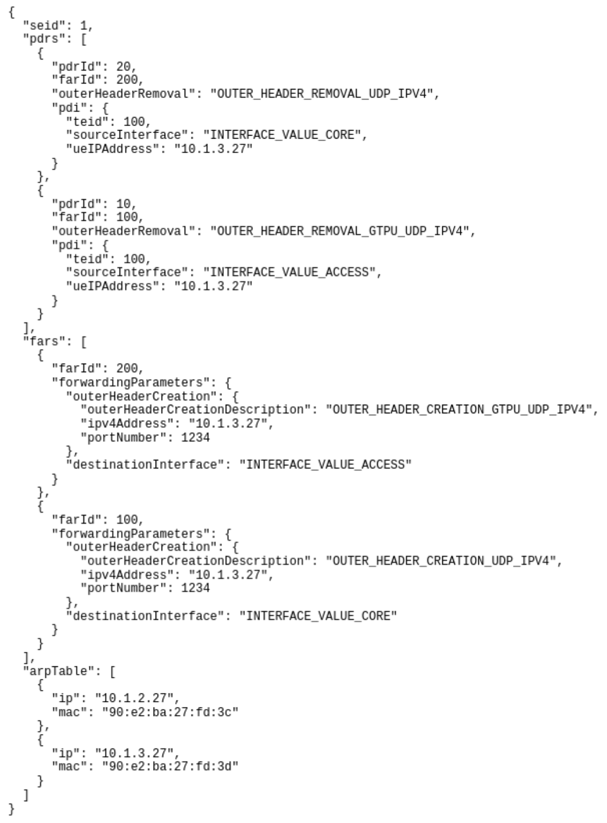
C. Data Traffic Generation

The traffic was generated using the Field Engine modules available in Trex Stateless API. This engine generates 1000 flows varying the source IP address randomly. This technique is used to avoid the assignment to a specific receive queue, distributing the traffic between the receive queues in the DUT. So, this improves the throughput due to the load balance between the cores. The GTP or UDP headers are generated depending on the test (uplink or downlink) based on the PFCP Session Establishment Request message. In both cases, the frame size is 64 bytes.

V. Results

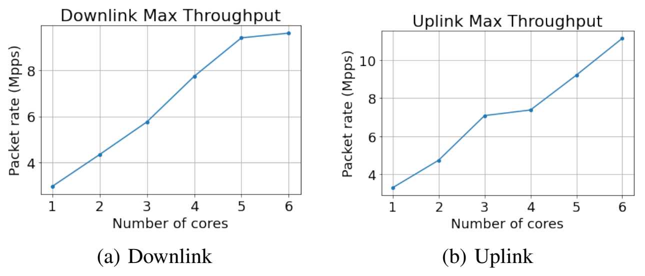
Figures 11 and 12 show the throughput per number of cores and the distribution of the CPU load for the cases when the number of cores is 4, 5 and 6 cores, respectively. There is a linearity between the number of cores and the throughput. So, the solution scales with the number of the cores, achieving almost 10 Mpps for downlink and greater than 11 Mpps for uplink. The numbers reflect the packet forwarding performance with packet loss less than 0.7%. Besides, almost 40% of the CPU is idle, which could be used by other tasks.

Figures 12a and 12c show a higher packet loss. The main reason for that is due to load balancing. As we can see, the core #1 load is 100% for both. So, the core was not handle all received packets and some them were overwritten from its rx queue. Basically, four factors contribute to the load balancing: NIC hash algorithm, NIC hash key (tuple), NIC indirection table, and the traffic. So, the load balancing could be optimized



```
{
  "seid": 1,
  "pdrs": [
    {
      "pdrId": 20,
      "farId": 200,
      "outerHeaderRemoval": "OUTER_HEADER_REMOVAL_UDP_IPV4",
      "pdi": {
        "teid": 100,
        "sourceInterface": "INTERFACE_VALUE_CORE",
        "ueIPAddress": "10.1.3.27"
      }
    },
    {
      "pdrId": 10,
      "farId": 100,
      "outerHeaderRemoval": "OUTER_HEADER_REMOVAL_GTPU_UDP_IPV4",
      "pdi": {
        "teid": 100,
        "sourceInterface": "INTERFACE_VALUE_ACCESS",
        "ueIPAddress": "10.1.3.27"
      }
    }
  ],
  "fars": [
    {
      "farId": 200,
      "forwardingParameters": {
        "outerHeaderCreation": {
          "outerHeaderCreationDescription": "OUTER_HEADER_CREATION_GTPU_UDP_IPV4",
          "ipv4Address": "10.1.3.27",
          "portNumber": 1234
        },
        "destinationInterface": "INTERFACE_VALUE_ACCESS"
      }
    },
    {
      "farId": 100,
      "forwardingParameters": {
        "outerHeaderCreation": {
          "outerHeaderCreationDescription": "OUTER_HEADER_CREATION_UDP_IPV4",
          "ipv4Address": "10.1.3.27",
          "portNumber": 1234
        },
        "destinationInterface": "INTERFACE_VALUE_CORE"
      }
    }
  ],
  "arpTable": [
    {
      "ip": "10.1.2.27",
      "mac": "90:e2:ba:27:fd:3c"
    },
    {
      "ip": "10.1.3.27",
      "mac": "90:e2:ba:27:fd:3d"
    }
  ]
}
```

Fig. 10: The JSON message representing the PFCP Session Establishment Request used in the tests.



(a) Downlink (b) Uplink

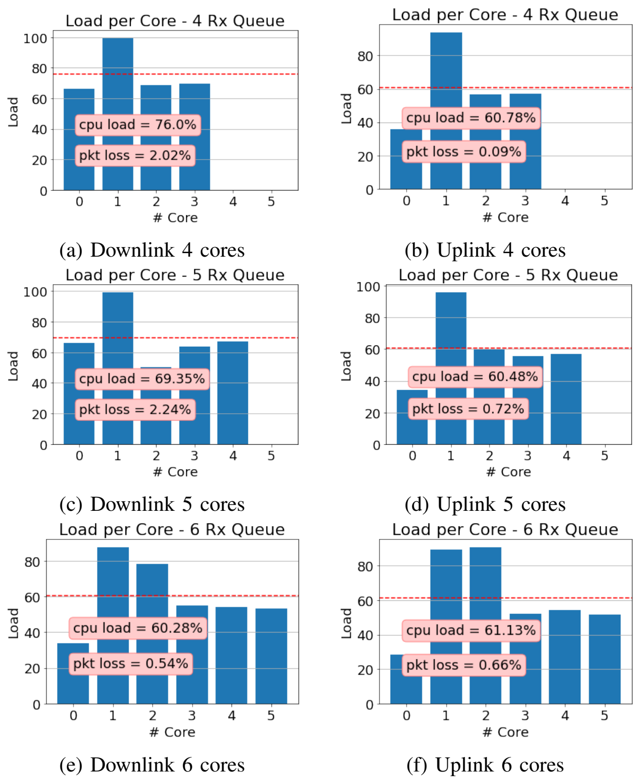
Fig. 11: Scalability of the proposed solution. The solution achieves almost 10 Mpps for downlink and 11 Mpps for uplink with packet loss less than 0.7%.

if we created a flow-based indirection table. This analysis is not the scope of this paper. It is important to highlight that our results are similar to [20], although their work has not presented a CPU usage for the scalability test.

VI. Related Work

Since the introduction of concepts such as NFV [2], several works have been carried out addressing the issue of fast packet processing [21] in general and tailored to 5G [22].

The authors in [23] present a prototype for packet processing based on hardware using the programmable platform NetFPGA with the programmable data plane in the P4 language.



(a) Downlink 4 cores
(b) Uplink 4 cores
(c) Downlink 5 cores
(d) Uplink 5 cores
(e) Downlink 6 cores
(f) Uplink 6 cores

Fig. 12: Workload distribution for varying number of cores.

The work consists of developing a firewall located between the core and the edge of 5G networks. The firewall is responsible for analyzing the internal IP headers of each packet, unlike the traditional firewall, which only analyzes the external header. In parallel, [24] also proposes a similar solution, but with a focus on multi-tenancy scenarios to ensure quality of service for applications with strict latency requirements, with tests performed with OpenAirInterface (OAI) [25]. Although FPGA based solutions perform well, they are considered expensive compared to general purpose CPU as well as complex to develop due to their low-level hardware description languages (low-level hardware description languages - HDLs), such as the VHDL language [21]. It is important to point out that these presented solutions do not seem to be ideal to be implemented in datacenters infrastructures located at the edge, which have restrictions regarding the implementation cost [4].

Regarding software-based packet processing solutions, [26] presents a prototype using the modular router, Click [27], integrated with the framework Netmap [28] for transferring session context between base stations. In addition, [29] evaluates a DPDK-based [30] prototype for media gateway located in the IP multimedia subsystem (Multimedia Subsystem - IMS). DPDK-based solution can increase performance 10 times for packet processing [5], however it is surpassed by XDP technology in scenarios when packet forwarding happens through the same interface [6].

Finally, to the best of our knowledge, the only work found in the literature that comes closest to our proposal is [20], which presents a prototype of a gateway based in eBPF using TC and XDP technology for fast packet processing. It focuses on developing a component that can be deployed on the edge. The main features presented are packet forwarding, classification and QoS policies. It was developed inside the Polycube Framework [31]. The main advantages of our work comparing with [20] are *(i)* it is decoupled to a specific frameworks, *(ii)* it is aligned with 3GPP specifications and *(iii)* it is based on *libbpf* instead of BCC [32], which does not belong to Linux source tree and depends on the *clang* run-time compiler. So, we argue that our solution can be easily integrated with different software-based 5G uses plane solutions.

The approach presented in this paper may leverage open source projects for telecommunication networks core, such as srsLTE [33], OAI [25], Open5GS [34], UPF-EPC [35], Magma Facebook [36], and Free5Gc [37]. Only srsLTE [33] does not provide support for NCG functionalities. The OAI, srsLTE, and Open5GS solutions do not have specific technologies for fast processing in the UP, which has been developed in the operating system's user space. Already UPF-EPC, Magma Facebook, and Free5Gc present kernel-level UP implementations using the kernel module gtp5g [38], BESS (Berkeley Extensible Software Switch) [39] and OvS (Open vSwitch) [40], respectively. With regard to CUPS support, we can highlight OAI, UPF-EPC, Open5GS, Free5Gc, and Magma Facebook. Although the Magma Facebook is based on an older version of the OAI without CUPS support, the solution was built using the SDN Ryu controller [41] and OvS. We believe all these projects could benefits from our proposed solution, especially those that do not support fast packet processing, like OAI, srsLTE, and Open5GS.

VII. Conclusion

This work addressed an alternative solution for 5G UPF to be deployed in a restrictive environment like MEC, where MEC host and UPF are collocated with the Base Station, sharing the same computational and network resources.

We have presented an open source C++ library, which implements the routing and forwarding user plane features based on 3GPP Release 16. Our evaluation has shown that the solution achieves 10Mpps with only 85% of CPU usage and 6 cores. Because it is an in-kernel solution based on eBPF/XDP, we believe it could be easily integrated with other solutions which run on Linux.

As future work, we intend to include QER in order to demonstrate user plan flexibility when new rules are created or removed and evaluate the behavior when varying the number of PFCP sessions and packet size. Furthermore, we plan to use real data traffic for tests and conduct a proof of concept to demonstrate an integration with an open-source 5G UPF.

Acknowledgments

This work was supported by grants 2018/23101-0 and 2020/05182-3 from the São Paulo Research Foundation (FAPESP). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] N. W. Paper, "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. issue 1," Oct. 2012.

[3] "Mec in 5g networks." [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[4] "Cloud edge computing: Beyond the data center - openstack open source cloud computing software." [Online]. Available: https://www.openstack.org/use-cases/edge-computing/cloud-edge-computing-beyond-the-data-center

[5] "Building enterprise-level cloud solutions with outscale." [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/case-studies/xeon-e5-2660-family-ssd-s3700-series-dpdk-case-study.pdf

[6] T. Høiland-Jørgensen, J. D. Brouer, D. Borkmann, J. Fastabend, T. Herbert, D. Ahern, and D. Miller, "The express data path: Fast programmable packet processing in the operating system kernel," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 54–66. [Online]. Available: https://doi.org/10.1145/3281411.3281443

[7] "Suse - introduction to ebpf and xdp." [Online]. Available: https://www2.slideshare.net/lcplcp1/introduction-to-ebpf-and-xdp

[8] "A thorough introduction to ebpf." [Online]. Available: https://lwn.net/Articles/740157

[9] "Bpf co-re (compile once - run everywhere." [Online]. Available: https://nakryiko.com/posts/bpf-portability-and-co-re/

[10] M. Vieira, M. Castanho, R. Pacífico, E. Santos, E. Pinto, and L. Vieira, "Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications," *ACM Computing Surveys (CSUR)*, vol. 53, pp. 1–36, 02 2020.

[11] "5g; system architecture for the 5g system (5gs) (3gpp ts 23.501 version 16.6.0 release 16)," 10 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

[12] "Lte; 5g; interface between the control plane and the user plane nodes (3gpp ts 29.244 version 16.5.0 release 16)," 11 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

[13] "Universal mobile telecommunications system (umts); lte; architecture enhancements for control and user plane separation of epc nodes (3gpp ts 23.214 version 16.2.0 release 16)," 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

[14] "Digital cellular telecommunications system (phase 2+) (gsm); universal mobile telecommunications system (umts); general packet radio service (gprs); gprs tunnelling protocol (gtp) across the gn and gp interface gprs tunnelling protocol (gtp) across the gn and gp interface (3gpp ts 29.060 version 16.0.0 release)," 10 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

[15] "Libbpf linux kernel library." [Online]. Available: https://github.com/libbpf/libbpf

[16] "Trex traffic generator stateless api documentation." [Online]. Available: https://trex-tgn.cisco.com/trex/doc/cp_stl_docs/api/index.html

[17] S. Miano, M. Bertrone, F. Risso, M. Tumolo, and M. V. Bernal, "Creating complex network services with ebpf: Experience and lessons learned," in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, 2018, pp. 1–8.

[18] "Trex - realistic traffic generator." [Online]. Available: https://trex-tgn.cisco.com/

[19] "Linux kernel network - receive side scaling." [Online]. Available: https://github.com/torvalds/linux/blob/master/Documentation/networking/scaling.rst#rss-receive-side-scaling

[20] F. Parola, S. Miano, and F. Risso, "A proof-of-concept 5g mobile gateway with ebpf," in *Proceedings of the ACM SIGCOMM 2020 Conference on Posters and Demos*, ser. SIGCOMM '20. Association for Computing Machinery, 2020.

[21] X. Fei, F. Liu, Q. Zhang, H. Jin, and H. Hu, "Paving the way for nfv acceleration: A taxonomy, survey and future directions," vol. 53, no. 4. New York, NY, USA: Association for Computing Machinery, Aug. 2020. [Online]. Available: https://doi.org/10.1145/3397022

[22] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128620311786

[23] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero, and Q. Wang, "Hardware-accelerated firewall for 5g mobile networks," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, 2018, pp. 446–447.

[24] R. Ricart-Sanchez, P. Malagon, P. Salva-Garcia, E. C. Perez, Q. Wang, and J. M. Alcaraz Calero, "Towards an fpga-accelerated programmable data path for edge-to-core communications in 5g networks," vol. 124, 2018, pp. 80 – 93. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804518302923

[25] "Openair-cn: Evolved core network implementation of openairinterface." [Online]. Available: https://github.com/OPENAIRINTERFACE/openair-cn

[26] B. Pinczel, D. Géhberger, Z. Turányi, and B. Formanek, "Towards high performance packet processing for 5g," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015, pp. 67–73.

[27] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, p. 263–297, Aug. 2000. [Online]. Available: https://doi.org/10.1145/354871.354874

[28] L. Rizzo, "Netmap: A novel framework for fast packet i/o," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'12. USA: USENIX Association, 2012, p. 9.

[29] W. Chen and C. H. Liu, "Performance enhancement of virtualized media gateway with dpdk for 5g multimedia communications," in *2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA)*, 2019, pp. 156–161.

[30] "Data plane development kit (dpdk)." [Online]. Available: https://dpdk.org

[31] "About ebpf/xdp-based software framework for fast network services running in the linux kernel." [Online]. Available: https://github.com/polycube-network/polycube

[32] "bpf compiler collection (bcc)." [Online]. Available: https://github.com/iovisor/bcc

[33] "srslte - open-source 4g and 5g software radio suite developed by software radio systems (srs)." [Online]. Available: https://github.com/srsLTE/srsLTE

[34] "Open5gs - open source project of 5gc and epc (release-16)." [Online]. Available: https://github.com/open5gs/open5gs

[35] "Upf epc - 4g/5g mobile core user plane." [Online]. Available: https://github.com/omec-project/upf-epc

[36] "Magma - facebook connectivity," c2020. [Online]. Available: https://connectivity.fb.com/magma/

[37] "free5gc - open-source project for 5th generation (5g) mobile core networks." [Online]. Available: https://www.free5gc.org/

[38] "Kernel module for gtp protocol." [Online]. Available: https://github.com/PrinzOwO/gtp5g

[39] S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy, "Softnic: A software nic to augment hardware," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-155, May 2015. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-155.html

[40] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vswitch," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'15. USA: USENIX Association, 2015, p. 117–130.

[41] "Ryu - sdn framework." [Online]. Available: https://ryu-sdn.org/