# Design and Implementations of non-3GPP Wireline Access Gateway for 5G Wireless and Wireline Convergence

Pai-Hui Wang, Hung-Chang Tsao, Li-Hsing Yen, and Chien-Chao Tseng
Dept. of Computer Science, College of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.

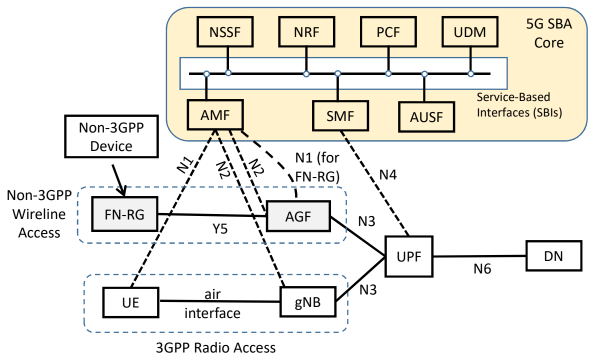


Fig. 1: Architecture of 5G wireless and wireline convergence

***Abstract*—To support the 5G Wireless and Wireline Convergence (5G-WWC) architecture, the Access Gateway Function (AGF) is defined to serve as an interworking function between the 5G core (5GC) and fixed network residential gateways (FN-RG) with a non-3GPP wireline access network. AGF acts as a proxy to communicate with 5GC on behalf of FN-RG. This paper presents a design for AGF that realizes control and user plane separation. Three different implementations based on user-space, kernel-space, and kernel-bypass (DPDK) softwares are reported and tested. The results confirm the superiority of the kernel-bypass approach in terms of latency and throughput of user-plane traffic.**

## I. Introduction

It has been an emerging trend to converge the fifth-generation (5G) system and non-5G access networks so that user equipment (UE) can connect to a 5G system through some access technology not specified in 5G. To this end, the 3rd Generation Partnership Project (3GPP) has specified the support for three types of non-3GPP access networks: untrusted, trusted, and wireline [1]. The first two deals with untrusted and trusted wireless access systems such as IEEE 802.11 (Wi-Fi), respectively. The last is for the 5G wireless and wireline convergence (5G-WWC). The 5G-WWC not only allows network operators to expand their service coverage but also eliminates the need to deploy a dedicated core network for the wireline network.

3GPP has collaborated with the Broadband Forum (BBF) to develop 5G-WWC-related specifications. A joint effort is specifying a new network function named the Access Gateway Function (AGF). AGF enables connectivity to 5G core (5GC) from the 5G-capable residential gateway (5G-RG) or the fixed network residential gateway (FN-RG). 5G-RG is capable of exchanging control-plane messages with 5GC through gNodeB (gNB). On the other hand, FN-RGs are legacy gateways (such as cable modem) that are not 5G capable, so AGF should act as a UE proxy for FN-RGs (Fig. 1). In the control plane, AGF plays the role of a gNB, which communicates with the Access and Mobility Management Function (AMF) in 5GC through the N2 interface. AGF also plays the role of a UE for each FN-RG and communicates with AMF through the N1 interface. AGF delivers packets between FN-RG and 5GC in the user plane through the N3 interface.

In this paper, we report on the design and implementations of an AGF. To provide scalability and align with the 5GC design principle, we divide AGF into the AGF Control Plane (AGF-CP) and the AGF User Plane (AGF-UP). Our focus is twofold:

- Design and implementation of AGF to provide a wired connection point for FN-RG and act as an agent for FN-RG to communicate with 5GC.
- Providing line-rate data flow transmission between FN-RG and 5GC.

We detail three implementations that share most AGF-CP codes but differ in AGF-UP. The first implementation uses UERANSIM [2], an open-source software, to implement AGF in the user space. The second implementation further uses a kernel module gtp5g [3] to speed up packet encapsulation and forwarding functions in the user plane. The last implementation uses DPDK [4] to bypass kernel processing of incoming packets so that the DPDK AGF-UP application can directly access and process these packets. We set up an experimental environment to test the latency and throughput between FN-RG and the data network (DN). The results demonstrated the effectiveness of gtp5g and DPDK in accelerating packet processing in AGF-UP.

The remainder of this paper is structured as follows. Sec. II briefs the backgrounds and related work. Sec. III presents the design and requirements. The following three sessions present three different implementations. Sec. VII shows the experimental results, and the last section concludes the paper.

## II. Background and Related Work

### A. Related Network Functions in 5G Core

5G-WWC benefits from the Service Based Architecture (SBA) [5] of the 5GC, which turns applications and services of the 5GC into fine-grained network functions. This restructuring leverages Network Function Virtualization (NFV) to enable flexible and agile network deployment [6].

In addition to SBA, 5GC also performs Control and User Plane Separation (CUPS), which separates the control plane from the user plane. In the control plane, network functions communicate with each other by a common service interface (Service Based Interface; SBI). Among these network functions, the AMF plays a crucial role. It is in charge of the initial registration, authentication, location tracking, and UE paging. It also manages the sessions between the UEs and DN. In particular, the AMF interworks with the AGF to establish the PDU session for an FN-RG on the wireline network. In the user plane, the User Plane Function (UPF) processes and forwards user packets. Our previous work applied CUPS to UPF to accelerate data plane performance [7].

## B. Related Work

Lemes et al. [8] summarized the non-3GPP trusted, untrusted, and wireline access systems in the 3GPP standards, provided analysis and implementation descriptions for various access systems, and compared the differences among all access systems. The paper also presented a untrusted non-3GPP access using Wi-Fi that allows user devices to connect to DN through 5GC. Their experiments measured the number of control messages, packet size, and processing time required for registration when establishing a connection. The part of non-3GPP wireline connection that was not implemented in this study will be the focus of our work. Also, unlike this study, our work will focus on data layer transmission performance in the testing part, including bandwidth and delay.

The study in [9] focused on the authentication mechanism of UE in the 5GC and the specifications for the authentication of user devices in the 5G-WWC standard for trusted and untrusted non-3GPP connections, as well as wireline connections. For the wireline connection part, this study explained the authentication for 5G-RG and FN-RG separately.

## III. Design of non-3GPP Wireline Access Gateway

## A. Functional Requirements

Any implementation of AGF should provide the following functionalities:

- *Enabling FN-RG attachment*. AGF needs to provide IP connectivity for FN-RGs that dynamically attach to it. This can be achieved via the Dynamic Host Configuration Protocol (DHCP).
- *Performing UE registration and authentication*. AGF should perform UE registration and authentication on behalf of FN-RGs. After completing these procedures, 5GC treats each FN-RG as a registered UE.
- *Establishing PDU session*. Since the registered UE uses Protocol Data Unit (PDU) session for data exchange between the UE and the UPF, AGF should request PDU session establishment on behalf of FN-RGs.
- *Exchanging control messages with 5GC via the N2 interface*. This is needed because AGF itself should act as a gNB.
- *Encapsulatiing/decapsulating data stream in N3*. For data delivery through the N3 interface, a tunnel between gNB and UPF should be established using the GPRS Tunneling Protocol-User Plane (GTP-U) [10]. Since AGF acts as a gNB, it should encapsulate user data from FN-RG and decapsulate user data from UPF using GTP-U.

Except for the last functionality, which is the main task of AGF-UP, all the functionalities mentioned above are handled by AGF-CP.

## B. AGF Control Plane (AGF-CP)

Our design further divides AGF-CP into DHCP module, Proxy UE, and N2 module. *DHCP module* provides the parsing of DHCPv4 Discover and Request messages according to the RFC 2152 standard and offers the FN-RG IP address through Offer and Acknowledge messages, completing the four-way DHCP handshake, so that FN-RG can attach to AGF. The DHCP module also tracks the connection status of FN-RG and determines whether the device is connected or disconnected through periodic updates of the DHCP lease. *Proxy UE* performs UE registration, authentication, and PDU session establishment request on behalf of FN-RG. Since FN-RG does not have 5G capability, AGF will launch a Proxy UE instance for each FN-RG attached. *N2 module* is responsible for the exchange of control messages with AMF through the N2 interface.

## C. AGF User Plane (AGF-UP)

AGF-UP provides the encapsulation and decapsulation of the GTP-U tunneling protocol. The processing speed of encapsulations and decapsulations significantly affects the service quality of user traffic such as throughput and delay. The main objective of this study is to implement AGF-UP using various approaches and see how their performance differs.

The endpoints of a GTP-U tunnel are uniquely identified by tunnel endpoint identifications (TEIDs), which are contained in the PDU session information obtained when Proxy UE establishes the PDU session. Therefore, there should be an interface between AGF-CP and AGF-UP for AGF-UP to access such information.

## IV. User Space Approach: UERANSIM AGF

Our first approach to AGF implementation takes advantage of UERANSIM [2], an open-source software that provides developers with tools for verifying the 5GC functions. The software consists of UE and gNB simulators. The UE simulator implements most of the messages used by the N1 interface. The gNB simulator implements most of the messages used by the N2 interface and the GTP-U tunneling protocol of the N3 interface, providing both a message exchange mechanism for the control plane and encapsulation and forwarding of user plane data.

The architecture of UERANSIM AGF is shown in Fig. 2. AGF-CP consists of the following modules:

- *DHCP Module*. We implemented a DHCP server using the Scapy library to parse and reply to DHCP packets. Furthermore, when receiving a DHCP Discover from FN-RG, the DHCP module will notify AGF Core (discussed below) of the device connection information, so that AGF Core can coordinate other modules to establish

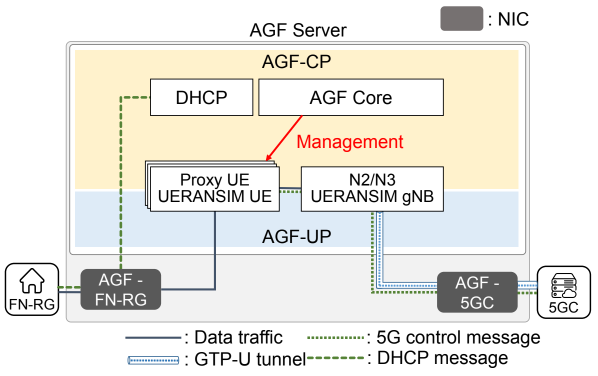


Fig. 2: Architecture of UERANSIM AGF

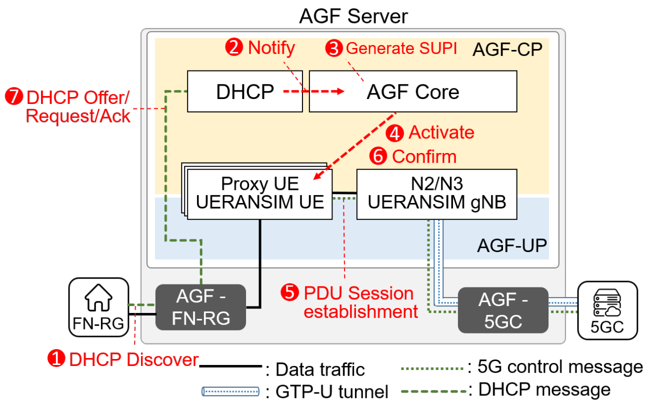


Fig. 3: UERANSIM AGF workflow

a PDU session between AGF (on behalf of an FN-RG) and UPF.

- *AGF Core*. We implemented this module to coordinate other AGF modules to help FN-RG establish a connection to 5GC. When receiving an FN-RG connection notification from the DHCP module, AGF Core will generate a Subscription Permanent Identification (SUPI) for the FN-RG based on the connection information. SUPI is assigned by 5G operators to uniquely identify a subscriber upon UE registration. AGF Core then uses that SUPI to create a corresponding proxy UE for FN-RG to establish a PDU session with the 5GC. Regarding the generation of the SUPI of FN-RG, the standard [11] demands the use of GLI (Global Line Identifier) as part of SUPI. Because GLI contains the Line ID source and the Line ID value, which represent the machine and the interface of the machine, respectively, that requirement allows 5GC to use the SUPI to identify the location of the wired connection point and provide the corresponding services based on the location. In our current implementation, AGF Core uses the SHA256 hash algorithm to obtain the hash value of the MAC address of FN-RG, and takes the first 10 digits as its SUPI. This shall be corrected in a later version to comply with the specification.
- *Proxy UE*. This module uses the UERANSIM UE simulator to implement the registration and setup of PDU sessions with 5GC. Since the UE simulator also contains the data stream transmission and reception functions, it also participates in implementing the user plane in this AGF implementation, which we will explain when describing the user plane module.
- *N2/N3 Module*. This module uses the UERANSIM gNB simulator to encapsulate Proxy UE control messages according to the N2 interface standard and send them to AMF. It also decapsulates the control messages from AMF through the N2 interface and sends them to the corresponding Proxy UE. Since the gNB simulator also contains the data stream transmission and reception functions, it also participates in the implementation of AGF-UP.

AGF-UP consists of the following parts:

- *Proxy UE* (shared with AGP-CP). After the PDU session is established, Proxy UE will use the UE IP address obtained during the establishment of the PDU session to create a TUN/TAP virtual device connected to the N2/N3 module. Afterwards, all IP packets from FN-RG will be directed to the corresponding Proxy UE TUN/TAP virtual device and will then be forwarded by network address translation (NAT) to the N2/N3 module.
- *N2/N3 Module* (shared with AGF-CP). It performs GTP-U encapsulation-related actions based on the IP address and TEID obtained by Proxy UE when establishing the PDU session. This module encapsulates data streams from Proxy UE as GTP-U packets, which are sent to UPF via the N3 interface. It also decapsulates UPF GTP-U packets and sends the data to the corresponding FN-RG. Due to the limitation imposed by the UE simulator, the UE IP address assigned by 5GC to Proxy UE in the creation of the PDU session cannot be assigned directly to FN-RG. Therefore, we need to assign an extra IP address to FN-RG (via DHCP), and use NAT to change the source IP address of the packets from FN-RG (which is the IP address assigned to FN-RG) to the UE IP address before forwarding these packets to the N2 / N3 module.

The connection process consists of several steps (see Fig. 3).

1) *Detecting FN-RG attachment*. The DHCP module receives the DHCP Discover message initiated by FN-RG (Step 1), indicating an attachment of a new FN-RG. The DHCP module notifies AGF Core of the attachment and sends the device information of the FN-RG to AGF Core (Step 2).
2) *Deploying Proxy UE*. AGF Core hashes the MAC address of FN-RG to generate an SUPI (Step 3). After generating the SUPI, AGF creates a corresponding Proxy UE instance for the FN-RG and transfers the SUPI to the Proxy UE to perform the registration (Step 4).
3) *Establishing PDU Session*. The Proxy UE uses the SUPI obtained from AGF Core to initiate UE registration and authentication with 5GC. When the Proxy UE completes the establishment of the PDU session, it obtains a UE IP address and TEID (Step 5). During the session establishment process, AGF Core will periodically check

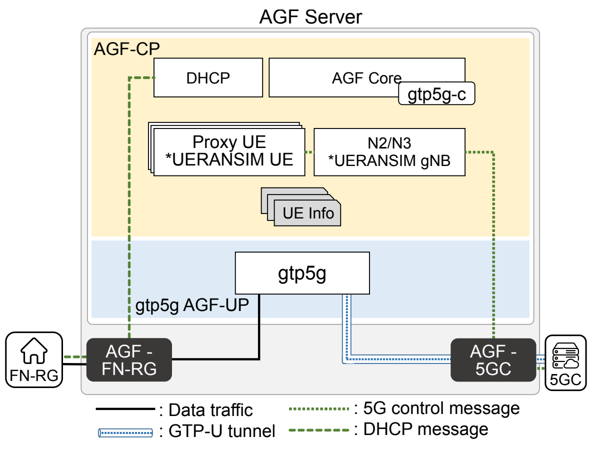


Fig. 4: Architecture of gtp5g AGF

the log of the Proxy UE to confirm the establishment of the connection (Step 6).

4) *Assigning FN-RG IP address*. After the PDU session is established, the DHCP module assigns a valid IP address to the FN-RG (Step 7). Meanwhile, AGF Core installs the corresponding IP rules and NAT rules, so that the data flow from FN-RG can be correctly forwarded to the N2/N3 module through the TAP/TUN virtual device established by the corresponding Proxy UE.

## V. Kernel Space Approach: gtp5g AGF

We improved the performance of UERANSIM AGF by replacing the user-plane packet encapsulation and forwarding functions in the Proxy UE and N2/N3 modules with gtp5g [3]. gtp5g is a core module (kernel module) developed by the free5GC team [12] which provides GTP-U encapsulation and decapsulation in the kernel space. Since the N3 interface is now implemented by gtp5g, we renamed the N2/N3 module in UERANSIM AGF as the N2 module in gtp5g AGF. The architecture of gtp5g AGF is shown in Fig. 4.

The following are the new modules or modifications to the original modules in the gtp5g AGF.

- *DHCP module*. The DHCP module no longer assigns an independent IP address to the FN-RG, but directly assigns the UE IP address obtained from 5GC to the FN-RG. This modification eliminates the need to implement NAT in the AGF. The reason for the change is that gtp5g AGF no longer uses the user plane function of the UERANSIM UE simulator, so Proxy UE no longer needs to create a TUN/TAP virtual device.
- *AGF Core*. AGF Core now reads the PDU session information stored in AGF-CP by Proxy UE and N2 module, converts the PDU session information into packet processing rules of gtp5g, and installs the rules to the gtp5g core module of AGF-UP using libgtp5gnl. We have named this feature the gtp5g controller (gtp5g-c). In addition, AGF Core now also passes the UE IP address allocated by 5GC to the DHCP module, which further assigns the IP address to the corresponding FN-RG.
- *Proxy UE*. Proxy UE follows the implementation of UERANSIM AGF, using the UERANSIM UE simulator. We removed the user plane function from the UERANSIM UE simulator, and only keep the control plane part of the function, including UE registration and PDU session establishment. We also added a function to Proxy UE, which allows it to store the UE IP address allocated by 5GC in a text file, UE Info, in the system. This file serves as an interface between AGF-CP and AGF-UP and is shared among the relevant modules.
- *N2 module*. The N2 module still uses the gNB simulator. However, all user-plane functions, including the encapsulation and decapsulation of GTP-U packets, are now removed. Only the 5G control message forwarding function is preserved. We also added a new function to the N2 module, which allows it to store the TEID in UE Info.
- *gtp5g*. gtp5g provides the core module of GTP-U encapsulation and decapsulation for the user-plane data flow between FN-RG and UPF. We use it directly without any modification.

The connection process of FN-RG is basically the same as in the UERANSIM AGF except for the following two points. First, Proxy UE and the N2 module write the UE IP address and TEID, respectively, in UE Info after establishing the PDU session. AGF Core then reads out the information, converts it into gtp5g rules, and installs the rules into the gtp5g core module of AGF-UP based on libgtp5gnl. Second, AGF Core passes the UE IP address to the DHCP module, which then allocates the address to the corresponding FN-RG, completing the four-way DHCP handshake.

## VI. Kernel-Bypass Approach: DPDK AGF

DPDK [4] is a development platform and interface that provides fast packet processing. User programs can access a network interface card (NIC) in the system through the application programming interface (API) provided by DPDK. The following two techniques are taken by DPDK to speed up packet processing:

- *Kernel Bypass*. Traditionally, any operation pertaining to NICs must be processed by NIC driver and TCP/IP protocol stack in the system kernel, which causes significant processing delays to applications. The API provided by DPDK allows a direct exchange of messages between applications and NICs, reducing processing delay by bypassing the kernel process.
- *Poll Mode Driver*. Traditionally, an NIC triggers an interrupt to the NIC driver located in the kernel when the NIC receives an incoming packet. By contrast, DPDK uses a polling mechanism to continuously query the NIC to retrieve and process packets in real time, significantly reducing the time required for packets to be processed by the application. Although the polling mechanism significantly consumes the CPU power, it is not a serious problem because CPUs nowadays are powerful enough to bear the processing pressure caused by the polling mechanism. For example, a CPU core can be dedicated to the polling task.

DPDK also proposes the kernel NIC interface (KNI) technology [13] to allow the coexistence with applications that interact with NICs through traditional Linux drivers

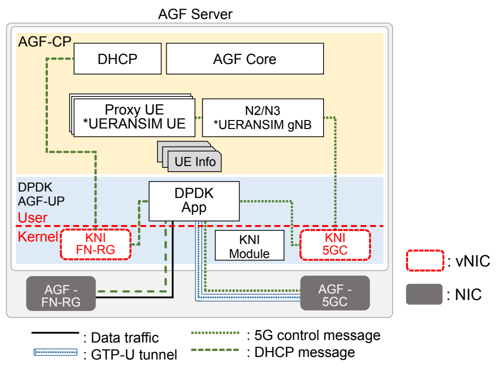


Fig. 5: Architecture of DPDK AGF

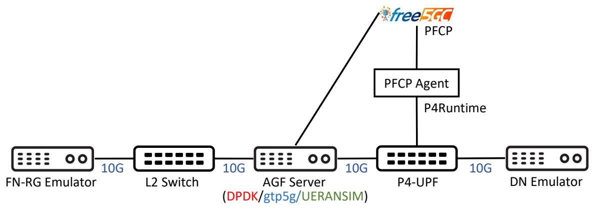


Fig. 6: AGP experiment environment

and TCP/IP protocol stack. This is achieved by creating a virtual NIC (vNIC) that is managed by a Linux NIC driver. Thus, traditional applications can access a vNIC through the corresponding NIC driver.

We used DPDK to implement AGF-UP, which achieves faster packet processing and provides development flexibility. The implementation basically follows the gtp5g AGF but replaces the gtp5g module with a DPDK application, as shown in Fig. 5. We disconnected the two physical NICs in AGF (one to FN-RG and the other to 5GC) from the Linux core network drivers and reconnected them to the DPDK polling driver. To reuse the AGF-CP modules, we created two vNICs in AGF-UP so that the AGF-CP modules can continue to run without modification.

In AGF-CP, we removed gtp5g-c from AGF Core. When detecting an update to the UE Info file, AGF Core notifies the DPDK application located in AGF-UP to adjust the packet processing rules. Since the adjustment is now handled by the DPDK application itself, the time from the establishment of the PDU session to the completion of AGF-UP rule adjustments is further reduced.

The DPDK application consists of five functions. The first is DPDK Init, which is used for program initialization, and its main functions include NIC configuration and memory allocation, etc. The second is KNI Manager, which controls the KNI core module through the DPDK KNI API, helping to adjust the KNI virtual device. The third is the Packet Handler, which handles all received packets. This function is needed because all NICs are no longer managed by the Linux core due to kernel bypass, so the application must handle incoming packets by itself. The fourth is the GTP-U Module, which provides the encapsulation and decapsulation capabilities of the GTP-U tunneling protocol. It is invoked when the Packet Handler receives packets that need to be encapsulated or decapsulated. When performing encapsulation and decapsulation, the GTP-U module operates according to the information provided by its own UE Info Storage and configuration file. UE Info Storage contains all UE IP addresses and TEIDs, and the configuration file contains information such as the MAC addresses of the two physical NICs in AGF. The last function is the UE Information Loader. When receiving a notification of an AGF Core update of UE Info, the UE Information Loader reads the UE Info file and stores the UE IP address and TEID in the DPDK application database.

The FN-RG connection process is basically the same as in the gtp5g AGF except for the following point. After AGF Core detects and reads out the UE IP address and TEID in the UE Info file, it notifies the DPDK application of the information (instead of converting it into gtp5g rules and installing the rules to the gtp5g core module). The DPDK application then updates its own database to perform GTP-U encapsulation and decapsulation for the new connection.

## VII. Performance Evaluation

We conducted performance comparisons for the three AGF implementations. We used free5GC as the 5GC, and paired it with P4-UPF to ensure the bandwidth of the user plane in 5GC. We used a server to emulate FN-RG, and connected it to the AGF server through an L2 switch. On the AGF server, we ran each AGF implementation for performance comparison. All connections used an SFP+10G optical fiber cable. Finally, we used a server to emulate DN and connected it with the FN-RG emulator. The test environment is shown in Fig. 6. Tables I and II show the server and switch specifications, respectively, used in the experiments.

### A. *End-to-End Latency*

We first measured the latency caused by packet forwarding and GTP-U encapsulation/decapsulation for each AGF implementation. We pinged the DN server from the FN-RG server, sending 100 ICMP ping requests at an interval of 0.2 seconds, and recorded all round-trip times (RTTs). The results are shown in Table III. We can observe that the UERANSIM AGF, which is a user-space implementation, exhibited the highest latency. On the other hand, the gtp5g AGF implementation, which is based on kernel space, achieved a latency of approximately 0.3 ms. The DPDK version relies on its kernel bypass mechanism and achieved the lowest latency among all three implementations.

### B. *Throughput of TCP Flows*

We ran the iPerf3 [14] client and server on FN-RG and DN, respectively, and sent TCP messages from FN-RG to DN. The test lasted 30 seconds, with throughput measured every second. In addition to this single-flow setting, we also set up a total of five pairs of iPerf3 client and server between FN-RG and DN, and established two TCP connections for each pair, creating a total of 10 TCP connections for a 30-second test. Fig. 7 shows the throughput of each implementation per second for both single-flow and ten-flow settings.

As can be seen in the figure, UERANSIM AGF did not perform well in terms of throughput in both settings. Regardless of the number of flows, the throughput of DPDK

TABLE I: Server Specification

| Machine | CPU | Memory | OS | Note |
|---|---|---|---|---|
| FN-RG Emulator, AGF Server | Intel Xeon E5-2630 v4 2.2~3.1Ghz 10C20T | 128 GB DDR4-2133 | Ubuntu 18.04.1 LTS (Kernel 5.4.0-87-generic) | NIC: Intel X710 10Gbps SFP+ |
| DN Emulator | Intel Xeon E5-2630 v4 2.2~3.1Ghz 10C20T | 128 GB DDR4-2133 | Ubuntu 16.04.1 LTS (Kernel 4.15.0-142-generic) | NIC: Intel X710 10Gbps SFP+ |
| free5GC Server | Intel Xeon E5-2620 v4 2.1~3.0Ghz 8C16T | 128 GB DDR4-2133 | Ubuntu 18.04.4 LTS (Kernel 5.0.0-52-generic) | |
| PFCP Server | Intel Core i5-8500 2.1~3.0Ghz 8C16T | 32 GB DDR4-2133 | Ubuntu 18.04.4 LTS (Kernel 4.15.0-144-generic) | ONOS Ver. 2.2.0 |

TABLE II: Switch Specification

| Switch | Hardware | Software |
|---|---|---|
| L2 Switch | Inventec D10056 | Barefoot SDE Ver. 9.3.0 |
| P4 UPF | Edgecore Wedge 100BF32X | Barefoot SDE Ver. 8.9.1 |

TABLE III: End-to-End Latency Comparison

| | UERANSIM | gtp5g | DPDK |
|---|---|---|---|
| Max. | 1.640 ms | 0.406 ms | 0.205 ms |
| Min. | 1.130 ms | 0.213 ms | 0.122 ms |
| Avg. | 1.376 ms | 0.309 ms | 0.169 ms |
| Mean Deviation | 0.074 ms | 0.035 ms | 0.018 ms |

AGF was estimated to be 9.92 Gbps when counting the header length of each packet (116 bytes, which iPerf3 did not count). This result can be considered reaching the link rate. gtp5g AGF reached almost half of the link rate (10 Gbps) with one flow and achieved a throughput of approximately 1 Gbps lower than that with ten flows.

To see why gtp5g AGF had a lower throughput with ten TCP flows, we studied the dynamics of CPU usage during the test for the ten-flow TCP setting and compared it with another setting with ten UDP flows. Fig. 8 shows the result. We can see that gtp5g well utilized the multicore CPU architecture for UDP flows, but did not do the same to TCP flows. This could justify the performance degradation observed when gtp5g AGF handled ten TCP flows.

## VIII. Conclusions

This article reports on a design for AGF with three possible implementations. The first implementation is based on the open-source user-space software UERANSIM. The second implementation uses a kernel module, gtp5g, to handle the encapsulation and forwarding functions of the user plane packets. The third implementation bypasses kernel interrupts to further speed up user-plane data processing. The results exhibit the benefits of all acceleration techniques.

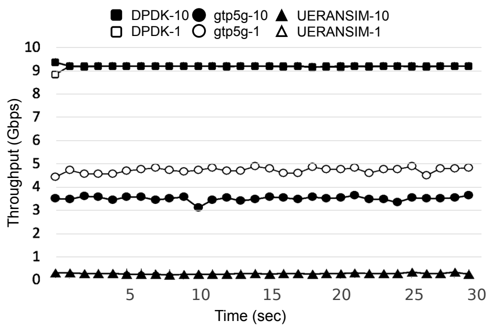
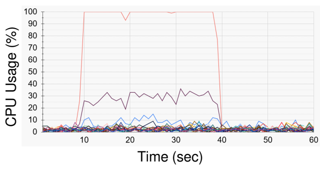
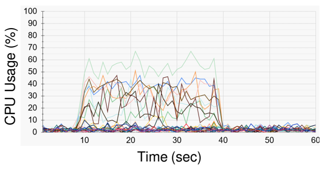


Fig. 7: TCP throughputs in single-flow and multi-flow settings



(a) Multi-flow TCP



(b) multi-flow UDP

Fig. 8: CPU usage in (a) multi-flow TPC and (b) multi-flow UDP during the test time. Each line represents a logical core (40 cores in total).

## Acknowledgement

This work was supported in part by the National Science and Technology Council, Taiwan, under Grants MOST 110-2221-E-A49-044-MY3, MOST 110-2221-E-A49-064-MY3, NSTC 112-2218-E-011-004 and NSTC 112-2218-E-011-006.

## References

[1] "Access to the 3GPP 5G core network via non-3GPP access networks 3rd generation partnership project-3GPP," 3GPP, TS 124.502, V16.7.0, Release 16, Apr. 2021.

[2] "GitHub: aligungr/UERANSIM: Open source 5G UE and RAN (gNodeB) implementation," https://github.com/aligungr/UERANSIM, accessed: 2023-05-10.

[3] "gtp5g: 5G compatible GTP kernel module," https://github.com/free5gc/gtp5g, accessed: 2023-09-16.

[4] "Data plane development kit," https://www.dpdk.org/, accessed: 2023-09-17.

[5] "System architecture for the 5G system (5GS)," 3GPP, TS 23.501, V16.6.0, Release 16, Sep. 2020.

[6] P. Rost, A. Banchs, I. Berberana, M. Breitbach, M. Doll, H. Droste, C. Mannweiler, M. A. Puente, K. Samdanis, and B. Sayadi, "Mobile network architecture evolution toward 5G," *IEEE Commun. Mag.*, pp. 84–91, May 2016.

[7] T.-H. Wang, M.-C. Hu, L.-H. Yen, and C.-C. Tseng, "Heterogeneous UPF integration framework and 5G user plane acceleration," in *Proc. 24th Asia-Pacific Network Operations and Management Symp.*, Sejong, Korea, Sep. 2023.

[8] M. T. Lemes, A. M. Alberti, C. B. Both, A. C. De Oliveira Junior, and K. V. Cardoso, "A tutorial on trusted and untrusted non-3GPP accesses in 5G systems—first steps toward a unified communications infrastructure," *IEEE Access*, pp. 116 662–116 685, 2022.

[9] T. Wan, Y. Sahin, and M. Pala, "Authentication in 5G wireline and wireless convergence," in *SCTE-ISBE Cable-Tec Expo*, Oct. 2019.

[10] "Universal mobile telecommunications system (UMTS); LTE; 5G; general packet radio system (GPRS) tunnelling protocol user plane (GTPv1-U)," 3GPP, TS 29.281, V15.7.0, Release 15, Jan. 2020.

[11] "Numbering, addressing and identification," 3GPP, TS 23.003, V16.3.0, Release 16, Jun. 2020.

[12] "free5gc," https://www.free5gc.org, accessed: 2023-09-17.

[13] "DPDK Kernel NIC Interface (KNI)," https://doc.dpdk.org/guides/prog_guide/kernel_nic_interface.html, accessed: 2023-09-17.

[14] "iPerf: The ultimate speed test tool for TCP, UDP and SCTP," https://iperf.fr/, accessed: 2023-09-17.