Extended Coverage without Roaming for beyond 5G Non-Public Networks

Marius Corici, Fabian Eichhorn, Bjoern Riemer, Thomas Magedanz
Software Defined Networking NGNI
Fraunhofer FOKUS Institute
Berlin, Germany
{marius-iulian.corici, fabian.eichhorn, bjoern.riemer, thomas.magedanz}@fokus.fraunhofer.de

***Abstract*—With the accelerating development and deployment of 5G Non-Public Networks (NPNs), many customized and private networks, providing services to local users and devices, are being installed across the world. To fully use the capacity of these networks, their services should also be reachable from other locations outside their limited coverage area. The only solution provided by 5G and beyond-5G 3GPP standardization at the current moment is based on roaming interfaces, which presumes that the NPN operator makes costly and complex peering roaming agreements with many different large scale and NPN operators. This article proposes an alternative to the roaming mechanism based on an Over-The-Top (OTT) peering approach. Similar to the non-3GPP integration, the connectivity in the visited network is transparently used to establish a binding with the home domain. Furthermore, the proposed architecture is implemented and evaluated in a basic form using Fraunhofer 5G Playground, a comprehensive 5G testbed network using commercial base stations and the Fraunhofer Open5GCore in conjunction with standard Android OS devices. Showing that such a solution represents a cost-effective, reduced functionality and relatively easy to implement solution outperforming the roaming approach in both dynamicity and ease of management.**

***Index Terms*—Mobile Networks, Non-Public Networks, NPN, national roaming, NPN extended coverage, 6G**

I. Introduction

5G enables the deployment of small size Non-Public Networks (NPN) providing a dedicated service for the specific use cases. In comparison to the public operator networks, NPNs are characterized by geographically limited coverage areas and local administration with its own specific policies. It is foreseen that in the near future a large number of NPNs will be deployed for the different use cases each with its own limited number of subscribers.

The NPNs give the possibility to locally administer the network with own policies and privacy as expected, and to further customize them for the specific requirements. However, in many situations the devices have to leave the NPN coverage area and still use the local services.

This type of situation was traditionally solved for large scale operators by developing specific international roaming mechanisms. These include the establishment of end-to-end peering between the networks, requiring the exposure of a specific set of roaming APIs as well as the negotiation of interchange policies in the form of roaming agreements.

978-1-6654-3540-6/22

However, with the deployment of NPNs, the need for these roaming mechanisms will significantly increase. Specifically, in this national roaming context, NPN devices pertaining to local networks are aiming to use the ubiquitous coverage of a wide area network (WAN) operator to reach their home network services. Although some advancements were made into NPN-NPN peering [1], the national roaming is stagnating mainly because of the potential business cannibalization threat in which an NPN could potentially provide full country coverage while only deploying a minimal network, and thus by roaming agreements, leaving the network deployment and management costs to the WAN operator.

This article presents an innovative solution to provide home services in the visited networks which does not require roaming mechanisms. Specifically, it is assumed that devices have two separate connectivity services: one in the home network and one in the visited WAN. While in the visited network, it uses the visited network subscription to establish a connection to the home network as to any public internet service. On top of this connection, a secure connectivity to the home network is established similar to the non-3GPP inter-working connections enabling the entry into the home network domain and access to the home network services.

Our new OTT interoperability concept is exemplified as an extension to the current 5G 3GPP standard, showcasing the easy integration with the existing features. Furthermore, we have implemented the concept using the Fraunhofer FOKUS Open5GCore toolkit and a regular Android OS App, enabling its assessment in a comprehensive realistic network environment.

The measured results from the test bed showcase the viability of the proposed solution, underlining the minimal overhead introduced against the significant reduction of the roaming functionality. With this we have proven that our solution overcomes the major functional limitations of roaming while at the same time provides the expected services for visiting devices making it a high-potential alternative to be considered for the further beyond-5G and 6G standardization.

The rest of the paper is organized as follows. In Section II we provide an overview of the 3GPP standard. In Section III we describe the architecture and the functionality of our OTT concept, followed in Section IV by our practical implementation of the concept and in Section V its evaluation. Finally,

in Section VI we provide the conclusions and an outlook into further work.

II. Background

Since the third generation (3G), 3GPP is the global standards body in charge of specifying wireless mobile networks. 3GPP first specified the 5G system (5GS) in its release 15 and later augmented it to support NPNs in release 16 [2]. In the 5GS the core network (CN) manages network access for user terminals (UTs) connected via radio access networks (RANs). How this works is presented briefly in this section.

A. Core Network

The CN comprises the services responsible for access control, authorization, accounting, mobility and session management of a mobile network. It is split into multiple network functions (NFs) that serve dedicated purposes. There are NFs for accounting and mobility (AMF), session management (SMF), user data management (UDM, UDR), network slicing support (NSSF), policy control (PCF), user authentication (AUSF), user plane traffic steering and forwarding (UPF) and more. These NFs belong to the so-called control plane, with the exception of the UPF, which exists in the user plane. An overview of the basic 5GS architecture is shown in Fig. 1. The CN NFs communicate over HTTP/2 in a service based

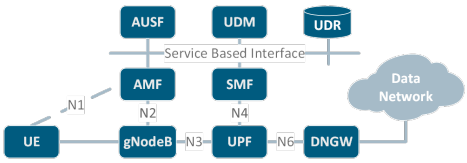

Fig. 1. Simplified 3GPP 5GS Architecture [3]

architecture (SBA). The RAN does not participate in the SBA and instead uses the N2 reference point towards the CN for signalling. To communicate with RAN devices e.g., eNodeB or gNodeB, the AMF supports the NG Application Protocol (NGAP) [4]. NGAP also allows the AMF to exchange signalling messages with UEs using the Non-Access Stratum (NAS) via the N1 reference point [5]. The NAS messages are forwarded by the gNodeB wrapped in NGAP packets. User plane packets from the UE are transmitted from the RAN to the UPF via the N3 reference point, using the GPRS Tunneling Protocol (GTP).

In the case of non-3GPP radio access technologies (RAT), a connection to the CN can be established using the Non-3GPP inter-working Function (N3IWF) for untrusted access networks or the Trusted Network Gateway Function (TNGF) for trusted ones [3]. These two NFs will serve as N2/N3 endpoints. Prior to any signalling, a UE needs to establish an IPSec security association and tunnel with the N3IWF/TNGF to enable a trusted connection. The authentication and security association for the IPSec connection, is performed using extensible authentication protocol (EAP) with the EAP-5G method and Internet key exchange version 2 (IKE) [6]. During this authentication the UE registers with the AMF using NAS, like during a regular attachment. The 3GPP also defines the trusted WLAN inter-working function TWIF, to allow devices which do not support NAS over non-3GPP registration to register with a 5G CN. In this case, the TWIF performs the registration on behalf of the UE and proxies an EAP-AKA' procedure for its authentication.

Since the specifications for 5G non-3GPP access of release 16 are still new and undergoing changes, compatible commercial devices were not known to the authors at the time of this writing. Therefore custom user terminal and gateway functions needed to be implemented.

The Fraunhofer Institute for Open Communication Systems (FOKUS) develops the Open5GCore, a 3GPP aligned 5G mobile CN for research and development [7]. Open5GCore provides software implementations of the main core NFs, as well as a virtual UE and gNodeB for RAN simulation. The Open5GCore was chosen as basis for the implementation.

B. User Terminal

Future 6G mobile networks will connect various end devices with different hard- and software capabilities. Smartphones and tablet devices, for example, are widespread and employed in many scenarios. They can feature user friendly interfaces, while also supporting wireless connectivity and mobility. For the sake of user privacy and security, their operating systems are often more restrictive than those of general purpose personal computers. One example is the Linux based Android operating system developed by Google Inc.. It is open source and supported by many device manufacturers. An easily accessible development environment, comprehensive documentation and global community of developers, make it a good platform for R&D endeavors. This work focuses on commercial off-the-shelf Android devices with support for WiFi access. No custom modifications to hardware or OS were made.

C. Problem Statement

5G NPNs are private networks that provide their connectivity services via the 5G new radio (NR) RAT and 5G CNs. They are also referred to as campus networks [8], mobile private networks (MPN) or plainly private 5G networks [9]. The operators of NPNs are often referred to as micro operators (μO) [10]. The potential benefits of NPNs have been discussed in the literature [11] [12] and above. Faced with potentially large financial investments for new hardware, an aspiring NPN operator may want to provide network services outside of the coverage area of their base stations, or to devices not equipped with the required networking hardware. Backwards compatibility and support for alternative access technologies have been identified as key requirements and challenges of 5G NPNs [11]. While a Public Network Integrated NPN (PNI-NPN) can take advantage of roaming to alleviate coverage issues, a standalone NPN (SNPN) cannot support roaming as of 3GPP's release 16.9.0 [1]. Neither does roaming allow connecting incompatible and legacy devices. The 3GPP also allows OTT connections via the Internet to an SNPN, but this has to be

supported by the devices and requires a publicly reachable N3IWF [3]. How can SNPN operators provide access outside the coverage area and to non-3GPP devices? They need a secure, standard-compatible means of accomplishing this. The N3IWF, TNGF and TWIF are intended to enable access to 5G networks via alternative access technologies. However, these NFs were specified recently and have not been implemented in networks, yet. Neither do many existing devices support the required functionality. NPN operators require an OTT solution, avoiding the need for hardware and OS level support. How can such an OTT solution be implemented?

III. Concept

To solve the issue of accessing (S)NPNs from outside and legacy (non-3GPP) devices, the core network architecture needs to be extended. The approach proposed by this work is shown in Fig. 2, where new elements are highlighted in green.

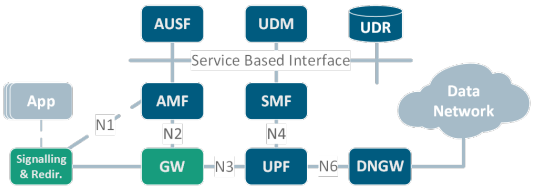

Fig. 2. Proposed Architecture Extension

It introduces a software component implementing the UE signalling to be executed on devices. This software also needs to redirect traffic of other apps and services of the device towards the CN. A gateway function replacing the gNodeB, to serve as an intermediary between this software and the CN, accompanies the UE side software. Together these components can emulate the RAN signalling and allow establishing a connection to the NPN. The existing core NFs do not have to be altered and can treat the new components like they would other RAN components and devices.

IV. Implementation

To implement the presented concept and enable non-standard roaming access to an SNPN, the following steps needed to be taken:

1) Implementation of a new CN function supporting signalling over a non-3GPP access network (AN)
2) Implementation of the signalling logic to perform standard UE procedures over a non-3GPP AN on the UE
3) Establishment of a virtual communications tunnel between CN and UE

How these steps were realized is further discussed in the following. The proposed architecture was implemented in the form of a gateway NF and an Android app called 5GConnect, as seen in Fig. 3.

A. Core Network Extension

As per the 3GPP architecture, the NAS signalling between UE and core network is normally proxied by a gNodeB. In the discussed scenario, the UE does not connect to a physical RAN and gNodeB with it's modem. A gateway NF, as indicated in Fig. 3, is required to fill the gap in this situation and the signalling packets need to be transmitted via the non-3GPP AN. Open5GCore provides both a N3IWF and virtual gNodeB to fill this gap. For the presented prototype, the virtual gNodeB was used. It can receive NAS requests via UDP and TCP and forward them to an AMF, using NGAP, to facilitate the signalling like it's physical counterpart.

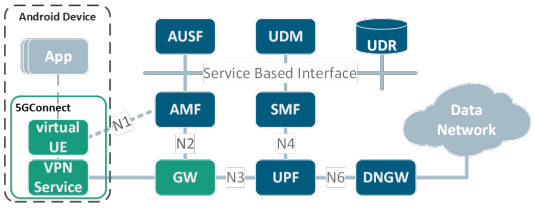

Fig. 3. Implemented Architecture

B. User Terminal Endpoint

To implement the user terminal, a prototype application for the Android operating system was developed. The application comprises three main parts, a simple user interface, implementations of the signaling state machines and the logic for tunneling data to the core network. In Fig. 3, the signaling is represented by the virtual UE and the tunnel logic by the VPN Service.

The user interface was implemented in Java/Kotlin. It was based on standard Android jet pack libraries and Material design guidelines. The implementation was focused on simplicity and straight-forwardness.

The signaling state machines were derived from the Open5GCore virtual UE implementation. It supports sending the NAS requests via a non-standard UDP messages. It was written in native C code interacting with the rest of the application through the Java Native Interface (JNI).

To create a virtual network tunnel, the app needs to be able to receive and forward the traffic of other software running on the device. On Linux, so-called TUN interfaces provide a virtual network interface for IP tunneling. The Android SDK provides the VPNService class, which can create such a TUN interface that intercepts traffic. It is intended for the creation of virtual private network (VPN) clients. Extending this class allowed the implementation of a service that receives a reference to a TUN interface during initialization. The same service was also provided with a UDP socket pointed towards the gateway, so it could pass back and forth packets. The VPNService class also facilitates requesting permission for intercepting traffic from the user in the same way other permissions are acquired on Android.

C. Tunnel

Virtual network connections between two endpoints are commonly created using tunneling protocols. A common example is the Generic Routing Encapsulation (GRE) protocol

[13]. GRE is a transport layer protocol, encapsulating data for transmission over an IP connection. The 3GPP specifies the use of GRE for non-3GPP access tunneling [3]. On the android operating system, regular user applications have limited options for communications and a standard GRE implementation was, at the time of implementation, not part of the framework. Therefore, a custom solution needed to be implemented. This was further inhibited, because non-root applications cannot create raw IP sockets, required to implement a transport layer protocol. The best alternative is transmitting the GRE packets over one of the common transport layer protocol implementations provided by the framework, i.e., TCP or UDP. An existing solution for UDP is "GRE-in-UDP Encapsulation" [14], which fit the requirements. It is straightforward and entails wrapping packets with an additional GRE header and sending them via UDP. The resulting protocol stack of the forwarding tunnel is depicted in Fig. 4.

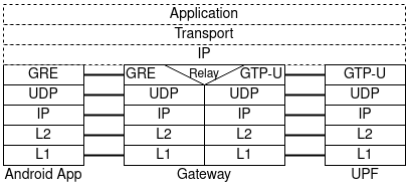

Fig. 4. Overlay tunnel protocol stack

On the CN's end a GRE in UDP endpoint can be implemented using Linux's foo over UDP [15] implementation with GRE encapsulation on top. The virtual gNodeB employs the underlying Linux system's IP tools to create the tunnel endpoint. It then listens for incoming packets on the associated socket. Any packets received are decapsulated by removing the tunnel headers. The packets are subsequently forwarded to the UPF using GTP-U, like regular user plane traffic. Conversely, packets received from the UPF are encapsulated with GRE and UDP headers and forwarded back to the android UE. On the Android device, received packets can be stripped of the GRE header and forwarded back through the TUN interface. With this the user plane tunnel between UE and CN gateway could be established.

However, there was another issue preventing end-to-end connectivity beyond the CN. Normally, the CN assigns a UE connected to its mobile network an IP address. The CN takes care of routing traffic to and from the UE based on this assignment. But, in the presented architecture, the UE's IP address cannot be controlled by the application, due to security restrictions of the OS, and because the UE is virtual and the underlying physical network access needs to remain intact. Therefore, the Android application must ensure that outgoing IP packets contain the source address assigned by the CN and incoming packets use a valid destination address of the end device. The presented solution handles this by rewriting the IP addresses in the IP packet headers and recomputing the checksums. In the classes receiving and sending data, packets are accessed and manipulated at byte level, using the Java NIO Buffer class and its sub-classes. The bytes of the source and destination addresses inside the IP headers are rewritten given their fixed offset within the buffer. Afterwards, the checksum of each packet is recomputed according to the Internet checksum algorithm [16] and written to the Java NIO buffer at the appropriate offset. Finally, the packets can be forwarded to the gateway or the TUN interface respectively. Thus the tunnel is completed and packets can be transmitted end-to-end.

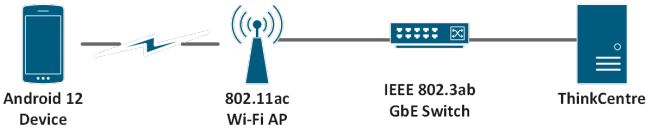

Fig. 5. Overview of the evaluation environment

V. Evaluation

To identify the performance impact of the tunneling, throughput and round trip time were identified as key performance indicators. They were measured in a controlled test environment. Within the test environment, the Open5GCore was run on a Lenovo ThinkCentre small form factor PC with an Intel® Core™ i7-6700T CPU @ 2.80GHz, 16GB of RAM and a standard Intel® on-board 1Gbps Ethernet card. The computer was running a standard Ubuntu Linux OS version 20.04. The Android devices were connected to the network using a Ruckus® T710 802.11ac WiFi access point. An 80MHz wide channel in a 5GHz band was used for the radio link. The Connection between ThinkCentre and access point passed through a data center grade manage GbE switch. The setup is visualized in Fig. 5. The measurements were conducted using two devices, a Samsung Galaxy Tab S7+ 5G and a Samsung Galaxy S21+ 5G, both running Android 12. Both devices can be counted towards early 5G capable devices supporting NSA and SA. One should note, that the devices, made by the same manufacturer, have different CPUs. The relevance of which will reveal itself in the following. Table I provides details on their CPUs, the Snapdragon 865 and the Exynos 2100 respectively.

TABLE I
CPU Specifications of Mobile devices used during evaluation; Source: GSMArena.com

Device	Year	CPU	Lith.	Core Frequencies
Galaxy Tab S7+ 5G	2020	Qualcomm SM8250 Snapdragon 865 5G+	7nm+	1x3.09 GHz Kryo 585, 3x2.42 GHz Kryo 585, 4x1.8 GHz Kryo 585
Galaxy S21+ 5G	2021	Exynos 2100	5nm	1x2.9 GHz Cortex-X1, 3x2.80 GHz Cortex-A78, 4x2.2 GHz Cortex-A55

The reader should note, that these measurements are from a limited campaign and potential errors cannot be ruled out.

A. TCP Throughput

TCP throughput was evaluated using the iperf3 tool. A cross-compiled iperf3 executable from an app available in the play store was used, on the smart devices. First the uninhibited throughput from the devices was measured in up and down link as presented in Fig. 6. With an average of around 650-700Mbps, the performance was good for an 802.11ac connection. Both devices reached close to identical rates.

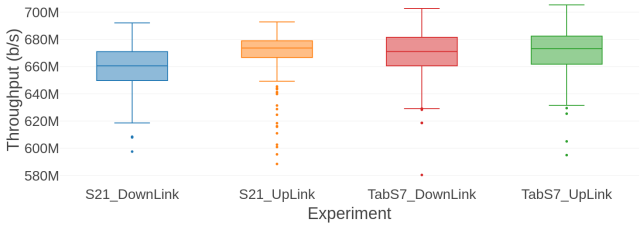

Fig. 6. iperf3 TCP Throughput: Galaxy Tab S7+ vs. S21+

For both devices, the TCP throughput was also measured, while the GRE tunnel towards the core network was active. To visualize the difference, Fig. 7 and 8 show both up and down link throughput with and without tunneling for the S21 and the Tab S7 respectively. It is clear that the tunneling causes a significant decrease in throughput. For both devices, a drop of circa 60-75%, down to around 160Mbps on average, can be observed. The S21 performed slightly better, as will be discussed later on.

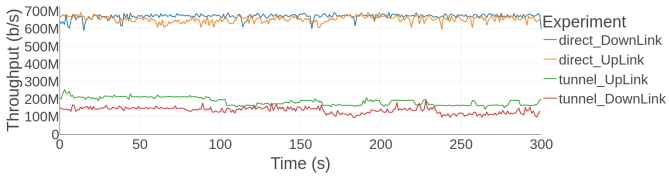

Fig. 7. iperf3 TCP Throughput from S21+ direct vs. tunnel

The tunneling affects throughput in two ways. First, there is an overhead introduced by adding tunnel headers to each packet. The minimal overhead is given by the sum of the bytes in the tunneling headers, the outer IP, outer UDP and GRE headers respectively: $overhead_{tunnel} = IP_{header} + UDP_{header} + GRE_{header}$. For IPv4 the minimum overhead amounts to 32 byte, which given a standard MTU of 1500 byte amounts to circa 2%. $MIN(overhead)_{IPv4} = 16byte + 8byte + 8byte = 32byte$. The overhead can be considered a minor factor and by itself this overhead cannot explain the large drop in throughput.

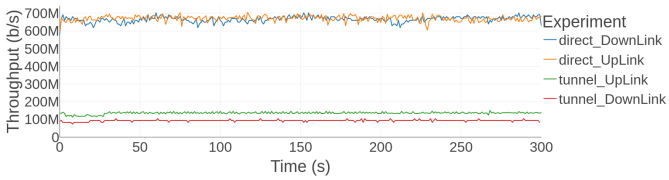

Fig. 8. iperf3 TCP Throughput from Tab S7+ direct vs. tunnel

Second, the prototypical tunneling implementation in a user space JVM application can only provide limited performance. Because the forwarding is executed on the CPU, the processing power of the underlying hardware limits potential throughput. Specifically, the tunneling is not taking advantage of dedicated networking hardware. When comparing tunnel throughput between the S21 and Tab S7, one can observe this quite well. Figure 9 compares the up and down link throughput

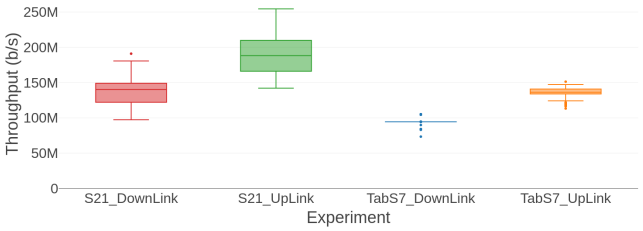

Fig. 9. Comparison of tunneled iperf3 TCP Throughput: S21+ vs. Tab S7+

with active tunneling for both devices. The red and green plots represent the down and up link throughput for the S21 and the blue and orange plots down and up link throughput for the Tab S7 respectively. Down link throughput was less for both devices, despite the fact that no notable difference could be observed over the direct WiFi connection. This has to be attributed to the forwarding implementation which handles the incoming and outgoing traffic differently. With the test environment practically identical, the S21 shows higher throughput in both directions, as well as a higher variance. Considering the CPU specifications in table I, reveals that both S21 and Tab S7 are equipped with quite similar 8 core CPUs. For both, the cores are split into three same size groups of similar clock frequencies. The most significant difference then stems from the lithography of the processors. The 5nm process of the Exynos 2100 appears to give it a slight edge in these measurements.

B. Round Trip Time

The effects of the tunnel on RTT were measured, by comparing the results for connections from the Galaxy Tab S7 towards public hosts on the Internet over 802.11ac and the laboratory internet access. The results for the www.fraunhofer.de and Quad9 (9.9.9.9) hosts, are presented in the following. For the comparison, a box and whiskers plot of all four measurements can be seen in Figure 10. The blue and orange graphs represent the RTT values measured towards fraunhofer.de and the green and red graphs represent those measured towards Quad9. One can only observe a minor impact on RTT from the tunneling. This conforms with expectations, since there was little to no background traffic and the ICMP packets of 64 Byte size should not stress the tunnel significantly.

VI. Conclusions and Further Work

This work presented a non-roaming, OTT solution to the problem of accessing 5G NPNs from incompatible devices or

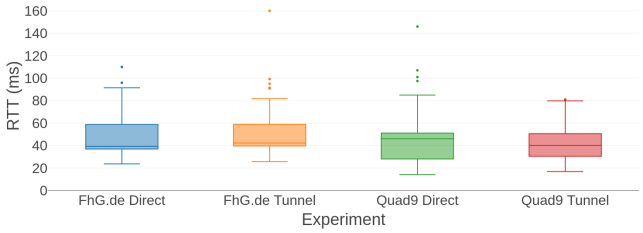

Fig. 10. RTT from test environment towards fraunhofer.de & 9.9.9.9: direct vs. tunneled

from outside of the coverage area. A problem which has, to the best the authors' knowledge, not been addressed adequately in literature and standards, yet. The solution employs a non-disruptive extension to the existing 3GPP architecture for 5G mobile networks, similar to the newly standardized non-3GPP access but foregoing device support. An implementation using the Open5GCore virtual NFs and the Android OS was detailed and evaluated using COTS devices. The presented approach was shown to be viable in R&D environments and less traffic-intensive use cases. While we don't expect this solution to outperform a PNI-NPN with roaming, with regards to throughput, for μOs who demand complete control over the end to end connection, the presented SNPN approach is preferable.

In the future, the android application can be extended to support standard non-3GPP access, which requires an IPSec implementation supporting EAP-5G. Such an implementation was not available on current Android OS versions to the best of the authors knowledge. Furthermore, multi-homing can be introduced as an extension for better performance [17].

Additionally, we consider adding a policy based decision layer on top of this implementation, to be able to determine and properly select the WAN operator transporting the communication in the ubiquitous environment, having additional data layer with acquisition and exchange capabilities [18].

To improve performance, alternative tunneling implementations, e.g., Android NDK C/C++ based, could be evaluated. Further evaluation on additional Android devices of differing specification and from additional manufacturers, could clarify the impact of the underlying hardware. The energy consumption and impact on battery life should be evaluated and optimized. Support of multiple device connections needs to be implemented and evaluated.

In future 6G networks the end devices are expected to become part of or collaborate closely with the RAN and CN control plane [19], [20], [21], [22]. The software presented can serve as a prototype to experiment with this collaboration and support initial evaluations. With its OTT approach, it will allow bypassing deeper system access, normally required for this kind of R&D work.

Acknowledgment

The authors acknowledge the financial support by the German Federal Ministry for Education and Research (BMBF) within the project "Open6GHub" {16KISK003K}. We also thank the Open5GCore [19] development team for providing continuous feedback from the software development perspective on how to simplify the implementation.

References

[1] M. Corici, P. Chakraborty, T. Magedanz, A. S. Gomes, L. Cordeiro, and K. Mahmood, "5g non-public-networks (npn) roaming architecture," in *2021 12th International Conference on Network of the Future (NoF)*, 2021, pp. 1–5.

[2] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5g evolution: A view on 5g cellular technology beyond 3gpp release 15," *IEEE Access*, vol. 7, pp. 127 639–127 651, 2019.

[3] *TS 23.501: System Architecture for the 5G System (5GS)*, 3GPP Std.

[4] *TS 38.413: 5G NG-RAN, NG Application Protocol (NGAP)/Non-Access Stratum (NAS)*, 3GPP Std.

[5] *TS 24.501: Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3*, 3GPP Std.

[6] *TS 33.501: Security architecture and procedures for 5G System*, 3GPP Std.

[7] Fraunhofer FOKUS Institute. (2022) Open5GCore - the next mobile core network testbed platform. [Online]. Available: www.open5gcore.org

[8] J. Rischke, P. Sossalla, S. Itting, F. H. P. Fitzek, and M. Reisslein, "5g campus networks: A first measurement study," *IEEE Access*, vol. 9, pp. 121 786–121 803, 2021.

[9] E. O'Connell, D. Moore, and T. Newe, "Challenges associated with implementing 5g in manufacturing," *Telecom*, vol. 1, no. 1, pp. 48–67, 2020. [Online]. Available: https://www.mdpi.com/2673-4001/1/1/5

[10] M. Matinmikko-Blue and M. Latva-aho, "Micro operators accelerating 5g deployment," in *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, 2017, pp. 1–5.

[11] J. Prados-Garzon, P. Ameigeiras, J. Ordonez-Lucena, P. Muñoz, O. Adamuz-Hinojosa, and D. Camps-Mur, "5g non-public networks: Standardization, architectures and challenges," *IEEE Access*, vol. 9, pp. 153 893–153 908, 2021.

[12] C. Guimarães, X. Li, C. Papagianni, J. Mangues-Bafalluy, L. M. Contreras, A. Garcia-Saavedra, J. Brenes, D. S. Cristobal, J. Alonso, A. Zabala, J.-P. Kainulainen, A. Mourad, M. Lorenzo, and C. J. Bernardos, "Public and non-public network integration for 5growth industry 4.0 use cases," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 108–114, 2021.

[13] T. Li, D. Farinacci, S. P. Hanks, D. Meyer, and P. S. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784, Mar. 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc2784.txt

[14] L. Yong, E. Crabbe, X. Xu, and T. Herbert, "GRE-in-UDP Encapsulation," RFC 8086, Mar. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8086.txt

[15] T. Herbert. (2014) [patch v4 net-next 0/7] net: foo-over-udp (fou). [Online]. Available: https://lwn.net/Articles/614433/

[16] R. Braden, D. Borman, and C. Partridge, "Computing the Internet checksum," RFC 1071, Sep. 1988. [Online]. Available: https://www.rfc-editor.org/info/rfc1071

[17] K.-F. Krentz and M.-I. Corici, "Poster: multipath extensions for wireguard," in *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 2021, pp. 1–3.

[18] M. Corici and T. Magedanz, ""one layer to rule them all" data layer-oriented 6g networks," *Shaping Future 6G Networks: Needs, Impacts, and Technologies*, pp. 221–233, 2021.

[19] M. Corici, E. Troudt, P. Chakraborty, and T. Magedanz, "An ultra-flexible software architecture concept for 6g core networks," in *2021 IEEE 4th 5G World Forum (5GWF)*, 2021, pp. 400–405.

[20] M. Corici, E. Troudt, and T. Magedanz, "An organic 6g core network architecture," in *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*. IEEE, 2022, pp. 1–7.

[21] M. Corici, E. Troudt, T. Magedanz, and H. Schotten, "Organic 6g networks: Decomplexification of software-based core networks," in *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 2022, pp. 541–546.

[22] The 5G Infrastructure Association. (2021, 06) 5G IA White Paper – European Vision for the 6G Network Ecosystem. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2021/06/WhitePaper-6G-Europe.pdf