Wireless Networks manuscript No.
(will be inserted by the editor)

User plane acceleration service for next-generation cellular networks

Engin Zeydan[a,1], **Yekta Turk**[2]

[1]Centre Technologic de Telecomunicacions de Catalunya,
Castelldefels, Barcelona, Spain, 08860

[2]Mobile Network Architect, Istanbul, Turkey, 34889

Received: date / Accepted: date

Abstract Reducing end-to-end latency is a key requirement for efficient and reliable new services offered by next-generation mobile networks. In this context, it is critical for Mobile Network Operators (MNOs) to enable faster communications over backhaul transport networks between next-generation base stations and core networks. However, MNOs will need to make new investments and optimize many points of their current transport infrastructure to serve next-generation services well. In addition, even if MNOs make these investments, there may always be faults and performance degradation in transport networks. This paper presents a new approach to reduce the dependence of MNOs services on the quality of transport networks and rely on software updates on Radio Access Network (RAN) and Core Network (CN) components. A Hyper Text Transfer Protocol (HTTP)-based User Plane (UP) that can be cached and accelerated is proposed, making it an ideal solution to combat transport problems in next-generation mobile networks. Numerical tests validate our proposed approach and underscore the significant improvements in transfer time, throughput, and overall performance achieved by leveraging HTTP caching and acceleration techniques. More specifically, GPRS Tunneling Protocol-User Plane (GTP-U) is, on average, 35% slower than HTTP, with the performance difference increasing as the data size grows, primarily due to additional overhead and GTP-U encapsulation time. Additionally, HTTP caching with a size of 20MB provides a 9.5% acceleration in data transfer time, with an average increase of approximately 9% when the data size exceeds 20MB.

Keywords User Plane · Caching · Acceleration · REST · HTTP

1 Introduction

In 5G deployments, the current quality level and latency status of their existing transport network infrastructure is a major concern for Mobile Network Operators (MNOs). In MNO backhaul networks, there could be different types of transmission technologies such as fiber, Radio Link (R/L), Dense wavelength division multiplexing (DWDM), legacy Internet Protocol (IP)/Multiprotocol Label Switching

[a]Correspondent author, e-mail: engin.zeydan@cttc.cat

(MPLS), or Software-Defined Networking (SDN)-based [1] transport which make the operation and maintenance of the network very complex. Some parts of the network can be operated under highly qualified and efficient conditions, but some parts of the current transport networks may need performance optimization. In particular, the quality of links and the condition of network equipment may be worse in rural areas than in urban areas. Especially, most Fiber to the x (FTTx) networks are not yet ready for next-generation cellular technologies [2]. Various factors such as physical damage, equipment malfunction, signal interference, or environmental conditions can still lead to faults in transmission links. Installing new fiber, repairing existing fiber, moving to a converged fixed-mobile architecture (e.g. as in [3]) or investing in new equipment for the large portions of the network can have huge costs. Moreover, even if the network is upgraded, these faults can still result in packet loss, delays, or increased latency [4]. Therefore, in this paper we investigate solutions that can minimize the transmission time over the backhaul link, based on the assumption that faults in the transmission links lead to additional latency.

End-to-End (E2E) delay tolerance times and technical expectations for 5G vertical services set by The 3rd Generation Partnership Project (3GPP) can be found in Table 1 [5]. Mobile network architectures are expected to evolve to serve these services while maintaining these defined latency values [6]. The latency requirements for different QoS Class Identifier (QCI) values of the different services in Long Term Evolution (LTE) can be found in the Table 2. Note that these defined values are higher than the requirements of 5G services. For LTE networks, User Plane (UP) latency was not an issue, but now 5G services are on the market, E2E delay in the UP has become a major issue. Because of low E2E latency requirements of 5G, existing terrestrial backhaul network of MNOs may have latency issues. Implementing caching-based acceleration techniques such as Performance-enhancing proxys (PEPs) executed on UP can be a beneficial approach to combat latency. This can certainly help meet 5G data E2E transmission requirements for backhaul links.

Table 1: 3GPP Defined Latency Needs for Vertical Industries in 5G.

Service	**E2E Latency Requirement**
Real-time control for discrete automation	$\leq$ 1 ms
Live Streaming	20 ms
Time-critical sensing	30 ms
Remote drone operation & Cooperative farm machinery	$\leq$ 30 ms
Smart grid	50 ms
Real-time video	<100 ms

Table 2: 3GPP Standardized QCI characteristics for LTE.

Service	**E2E Latency Requirement**
Video (Live Streaming) & Interactive Gaming	100 ms
Conversational Video (Live Streaming	150 ms
Video (Buffered Streaming) & TCP-based ftp, p2p, etc.	300 ms

1.1 Internet Protocol Security (IPSec) fragmentation

In mobile backhaul networks, IPSec tunnels are implemented between the Base Station (BS) and a Security Gateway (SecGW) located before the Core Network (CN), as defined in the 3GPP standard [7]. An end-to-end security distribution problem is illustrated in Fig. 1 when UP data acceleration is applied on routers which need to first detect the traffic. Since encrypted traffic over routers cannot be inspected, the IPSec tunnel must first be terminated for acceleration. Then, the UP packet is decapsulated and the payload is extracted from the packet. The acceleration process is applied and then the router at the other end receives the packet, encapsulate it with the UP header and then sends it to the CN. To secure this process, a second IPSec tunnel is established that terminates in the router at the other end. The router at the other end completes the acceleration process by encapsulating the payload and transmitting it to the CN. Finally, a third IPSec tunnel is established between this router at the other end and the SecGW. This three IPSec tunnels structure inevitably compromises the end-to-end security assurance. Furthermore, the establishment of many IPSec tunnels leads to an important IPSec tunnel management problem. In fact, multiple challenges may be associated with multiple tunnels. These include aspects such as key management, configuration complexity, scalability, and overhead. **In terms of key management,** managing encryption keys for multiple IPSec tunnels can become complex and resource-intensive. Each tunnel requires its own set of keys, and ensuring proper key distribution, rotation, and synchronization across all tunnels can be challenging. **In terms of configuration complexity,** as the number of IPSec tunnels increases, so does the complexity of the configuration. It becomes increasingly difficult to maintain consistent and error-free configurations for each tunnel, especially for large-scale deployments. Any misconfiguration can potentially compromise security or cause connectivity problems. **In terms of scalability,** scaling the number of IPSec tunnels can be challenging. Each tunnel consumes system resources, including memory and processing power, on both endpoints and network devices. As the number of tunnels increases, the scalability and performance of the overall system can be compromised. **In terms of overhead,** each IPSec tunnel introduces additional overhead in the form of packet encapsulation, encryption, and decryption processes. This overhead can affect network performance and throughput, especially when there are a large number of tunnels. It can also increase latency, especially for time-critical applications. **In terms of tunnel management and monitoring,** with multiple IPSec tunnels, monitoring and managing their status, traffic flow, and performance can become complex. Administrators need effective tools and mechanisms to ensure the proper functioning of each tunnel, detect potential problems or security breaches, and troubleshoot

issues efficiently. **In terms of interoperability and compatibility,** when multiple IPSec tunnels are established between different network elements or vendors, ensuring interoperability and compatibility can be challenging. Different implementations and configurations may have subtle differences that can lead to connectivity issues or reduced security. **Overall**, addressing these challenges requires robust management practices, automation tools, effective monitoring systems, and a thorough understanding of the underlying network infrastructure and tunneling technologies.

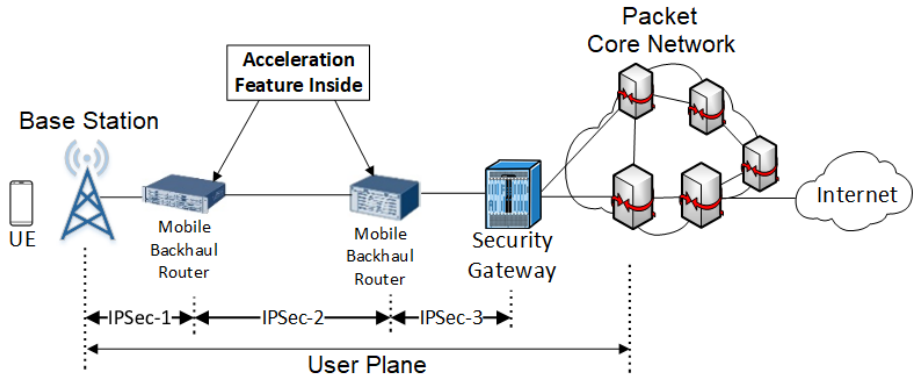

Fig. 1: The IPSec fragmentation problem when UP acceleration is applied to transport network elements.

To eliminate the delay effect in communication networks, the use of PEPs has long been practiced in networks [8]. PEPs uses caching of various types of network traffic for acceleration. The caching features of PEPs accelerate the transmission speed of protocols such as Transmission control protocol (TCP) in the case of broadband networks or IP traffic in satellite networks [9] and GPRS Tunneling Protocol-User Plane (GTP-U) traffic in UP of mobile networks [10]. Unfortunately, installing transport network elements with PEPs functionality is unnecessary and inefficient for the mobile networking case, as it leads to major security problems (E2E security cannot be guaranteed) which is an undesirable concern for both MNOs and regulators. The problem is the fragmentation of IPSec tunnels as described above. When PEPs are introduced into the network architecture, this can lead to additional complexity and fragmentation of IPSec tunnels, which in turn raises security concerns (see [11] and references therein). **First of all**, PEPs are primarily designed to improve network performance by accelerating traffic, performing optimizations, or caching. In the context of mobile networks, PEPs can intercept and modify network traffic to improve efficiency. However, this interception and modification can lead to fragmentation of IPSec tunnels [12]. The original IPSec tunnel is terminated at PEP, and a new IPSec tunnel is established between PEP and the destination. This creates multiple segments of the original tunnel, each terminated and reestablished at different points in the network. **Second**, the fragmentation of IPSec tunnels caused by PEPs can lead to security risks and compromise end-to-end security. The original IPSec tunnel was established to provide

secure communication between the source and destination endpoints. However, when PEPs intercept and modify traffic, the original IPSec tunnel is divided into segments, each of which is terminated and restored separately. This creates potential security vulnerabilities because each segment may have different security parameters, keys, or trust relationships. As a result, end-to-end security cannot be guaranteed for all fragmented segments of the tunnel [13]. **Third,** when the IPSec tunnel is fragmented, the trust and integrity of the original secure end-to-end channel are compromised. Each segment of the tunnel, including PEPs, becomes an additional point of potential vulnerability or attack. Any compromise or unauthorized access to any of these segments could compromise the overall security of the communications. **Finally,** the introduction of PEPs and fragmentation of IPSec tunnels may lead to regulatory compliance concerns.

1.2 Motivation

In this paper, we consider a Hypertext Transfer Protocol (HTTP)-based UP transmission scheme which improves throughput and transfer time under high latency backhaul scenarios. The design of the method is presented as a separate service for the Service-Based Architecture (SBA) of 5G to reduce the implementation complexity. The reasons for developing a separate service for the SBA of 5G, rather than upgrading existing services to a different transport protocol, are as follows: In terms of compatibility and interoperability, upgrading existing services to use a different transport protocol can introduce compatibility and interoperability challenges. Existing services may be tightly coupled with the current transport protocol and may not easily adapt to a new protocol. By proposing a separate service, it allows for the development and implementation of a new transport protocol without disrupting or impacting the existing services. This approach ensures backward compatibility and enables smooth integration with the existing ecosystem. The proposed method leads MNOs to provide next-generation mobile services with their existing infrastructures without investing more in their own transport networks. In terms of incremental deployment, introducing a separate service with a new transport protocol allows for incremental deployment and adoption. It provides the flexibility to pilot and test the new protocol in specific use cases or network segments without requiring a complete overhaul of the entire system. This approach enables controlled and phased implementation, allowing for fine-tuning, optimization, and gathering feedback before full-scale deployment. In terms of modularity and flexibility, by designing a separate service, the architecture can be modular, adaptable (evolve over time) and flexible. The new service can be developed independently, focusing on specific requirements and objectives, allowing for the incorporation of new technologies, protocols, and optimizations without affecting the existing services. Since the proposal in this study is defined as a service, smooth and easy integration with next-generation Radio Access Network (RAN) and CN can be achieved, especially in cloud environments, and a software-based system can be integrated to next-generation mobile network elements with software-based updates, enabling a plug-and-play architecture. In terms of performance and efficiency, the new service can be specifically designed to leverage the advantages of the proposed transport protocol. It can be optimized and fine-tuned to deliver superior performance, reduced latency, improved scalability, or

Table 3
Acronyms and their corresponding definitions

Acronym	Definition	Acronym	Definition
3GPP	The 3rd Generation Partnership Project	MNO	Mobile Network Operator
AP	Access Point	MPLS	Multiprotocol Label Switching
API	Application Programming Interface	MQTT	MQ Telemetry Transport
BS	Base Station	MSISDN	Mobile Station International Subscriber Directory Number
CA	Carrier Aggregation	MANO	Management and Orchestration
CN	Core Network	NC	Network Coding
CNF	Cloud Native Function	NFV	Network Function Virtualization
CPU	Central Processing Unit	NGC	Next Generation Core
CP	Control Plane	NR	New Radio
D2D	Device-to-Device	QoS	Quality of Service
DL	Download	QoE	Quality of Experience
DPDK	Data Plane Development Kit	OS	Operating System
DWDM	Dense Wavelength Division Multiplexing	OSPF	Open Shortest Path First
EPC	Evolved Packet Core	PDCP	Packet Data Convergence Protocol
E2E	End-to-End	PEP	Performance-Enhancing Proxy
EPS	Evolved Packet System	PS	Packet Switched
GTP	GPRS Tunneling Protocol	SDO	Standard Development Organization
GTP-U	GPRS Tunneling Protocol-User Plane	QCI	QoS Class Identifier
GNS3	Graphical Network Simulator-3	RAN	Radio Access Network
HO	Handover	RAM	Random Access Memory
CoMP	Coordinated Multipoint	REST	Representational State Transfer
HARQ	Hybrid Automatic Repeat Request	RF	Radio Frequency
HTTP	Hypertext Transfer Protocol	R/L	Radio Link
ETSI	European Telecommunications Standards Institute	QUIC	Quick UDP Internet Connections
FTTx	Fiber to the x	RU	Radio Unit
IAB	Integrated Access Backhaul	SBA	Service-Based Architecture
IMAP	Internet Message Access Protocol	SDN	Software-Defined Networking
IoT	Internet of Things	SLA	Service Level Agreement
I/O	Input/Output	SecGW	Security Gateway
IPSec	Internet Protocol Security	TB	Transport Block
IEEE	Institute of Electrical and Electronics Engineers	TEID	Tunnel Endpoint Identifier
IP	Internet Protocol	TSN	Time-Sensitive Networking
ITU	International Telecommunication Union	TTI	Transmission Time Interval
JSON	JavaScript Object Notation	TWAMP	Two-Way Active Measurement Protocol
KPI	Key Parameter Indicator	URL	Uniform Resource Locator
LDP	Label Distribution Protocol	UDP	User Datagram Protocol
LTE	Long Term Evolution	UP	User Plane
LTE-A	Long Term Evolution Advanced	UPF	User Plane Function
MAC	Media Access Control	UE	User Equipment
MIMO	Multiple Input Multiple Output	UL	Upload

other desired characteristics. Upgrading existing services to use a different transport protocol may involve significant modifications and adaptations, which can be complex and time-consuming. By developing a new service, the focus can be on harnessing the full potential of the new protocol without the constraints of legacy systems. In terms of future-proofing and innovation, introducing a separate service allows for future-proofing and innovation, providing the opportunity to explore and emulate with new concepts, protocols, and technologies that may not be feasible to integrate directly into existing services. By creating a dedicated service, it becomes a sandbox for innovation and enables the rapid development and deployment of novel ideas without disrupting existing operations. Table 3 contains acronyms and the corresponding definitions used in the paper.

The method proposed in this paper envisions accelerating all UP traffic between the gNodeB and the CN, regardless of whether inspecting each users' traffic flows. The UP transmission of 5G uses the N3 interface to transfer data between the base station and the CN. Therefore, it is ideal to deploy the proposed technique where the user traffic is aggregated and sent to the CN. On the core network side, the proposal can be located in the cloud-based architecture of User Plane Function (UPF), which is responsible for generating UP traffic. A summary of the comparisons between the existing studied with the proposed approach is summarized in Table 6. The contributions in this paper are summarized as follows:

– A HTTP-based UP transmission is recommended to be implemented for next generation mobile networks to achieve better performance (higher through-

put and lower data transfer time) than traditional GPRS Tunneling Protocol (GTP)-based systems, and this approach reduce the number of IPSec tunnels required between the gNodeBs and the core network by design.
– The high-level implementation of the cache-aided UP acceleration method is defined in detail and the advantages obtained are presented via numerical studies.
– Numerical results show that acceleration works well for UP transmission of base stations in high latency backhaul scenarios and different cache sizes.

2 Related Work

To meet delay requirements within 5G mobile networks, the RAN between gNodeB, and user equipment (UE) and the mobile backhaul between gNodeB and 5G Next Generation Core (NGC) must operate with minimal delay. There are also some studies in the literature that address acceleration of the RAN interface, but to the best of the author's knowledge, there are no proposed solutions to date that address acceleration of the UP. On the RAN side, there are ongoing studies on flexible transmission time interval (TTI) selection, delay-sensitive Hybrid automatic repeat request (HARQ) operation and effective Media Access Control (MAC) scheduling methods that are being investigated for improvements [14], [15], [16], [17]. One of the problems with the GTP-U is that it carries too much protocol overhead [18]. To achieve faster UP transmission between gNodeB and CN, researchers are investigating more effective and faster UP protocols to replace the existing GTP-U protocol [19], and Quick UDP Internet Connections (QUIC) protocol can be evaluated as an ideal one [20]. The draft document in [21] focuses on proposing some optimization solutions such as. reducing/eliminating encapsulation, using native routing mechanisms, efficient routing during and between mobility events, supporting anchorless mobility management & offloading local traffic, reducing session state and signaling associated with mobility management, and converging to a flatter architecture consistent with other mobility proposals for 5G UP. These are discussed within relevant scenarios like roaming, support for multiple PDU sessions, etc. Different solutions are also introduced using several techniques, such as Segment Routing v6 (SRv6), Locator Identifier Separation Protocol (LISP), Identifier Locator Addressing (ILA), and Hybrid Information-Centric Networking (hICN), and their use as replacements for GTP. Note that comparing such solutions based on different transport protocols can be difficult in general, as the underlying principles, mechanisms, and performance characteristics may differ significantly. The UP protocol proposed in this paper is based on HTTP which may bring too little overhead compared to GTP-U [22].

Time-Sensitive Networking (TSN) is a recommendation of the Institute of Electrical and Electronics Engineers (IEEE) work group that addresses providing faster transmission for transport networks in terms of flow sensitivity and faster switching. While TSN may provide guidelines and specifications for advanced and time-sensitive networking capabilities in the transport network, the gNodeBs that are part of the 5G infrastructure must still operate within the constraints and limitations imposed by the existing transport network domain [23]. The article in [24] investigates the characterization and performance of general-purpose processors, particularly x86 architecture processors, in both signaling and bearer processing

Table 4: A summary comparison of existing studies with the proposed approach.

Different Approaches		Proposed Approach	
Characteristics	**Limitations**	**Advantages**	**Differences**
Enhancement in the transmission speed and quality between UEs and BS [12], [24], [25], [26].	Has focus only on the RAN side.	Mobile networks will work independent from transport network performance issues.	Focus on accelerating the UP.
Investigate performance of various protocols [15], [16], [17], [18].	Focus only on protocol aspects.	A solution enhancing UP transmission is proposed.	Micro-services constituting the accelerator are presented.
Accelerating traffic in transport network elements [5], [6], [7], [20], [27].	Acceleration in transport network disrupts E2E security.	Keeps and satisfies mobile networks E2E security requirements.	UP acceleration is proposed as a service in mobile nodes.
Solutions on optimizing the GTP-U UP transmission [10], [22], [23].	Improvements are limited with the capabilities of GTP-U protocol.	Overcome the limitations of GTP-U (e.g. caching, compressing, etc.)	HTTP for UP is presented alternatively.

of Evolved Packet System (EPS). The advantage of this approach is that the UP packets are kept away from the kernel, allowing processing with fewer central processing unit (CPU) cycles. The acceleration technique described here is hardware based and can be applied with specialized hardware. The GTP-U acceleration method used in satellite networks is described in the patents of the Very small aperture terminal (VSAT) modem vendors [25], [26]. Although the proposed methods are specific to satellite networks, these studies have identified techniques for accelerating mobile UP data traffic. A method in [27] focuses on accelerating traffic in end-user flows, but does not provide intelligence for 5G base stations. Edge caching [28–30] can be considered as another technique to improve user traffic. However, for a user-centric network, it is often necessary to determine the user traffic in edge caching. This can add additional complexity to the base station architecture.

The rest of the paper is organized as follows: Section 3 presents the delay model and concepts related to UP acceleration in mobile network elements. Section 4 describes the architectural aspects of the proposed system. The emulation setup and analysis of the results are presented in Section 5. Concluding remarks and future research directions are presented in Section 6. Table 5 provides a list of symbols used in this paper.

3 Concepts and Delay Model

The E2E delay is measured as the delay incurred by the packets between the mobile end nodes, i.e. the New Radio (NR) and the UPF. The E2E delay can be expressed as

Table 5: Symbols used throughout the paper

Symbol	**Meaning**
D_{proc}	Processing time in CPU of a node
D_{proc}	Time spent in a queue at node
D_{trans}	transmission time of a packet
D_{prop}	Propagation time for a signal
L	Length of a packet
R_t	Transmission rate
c	Allocated size of cache
$\mathcal{R}$	set of dataset to be downloaded
D_{e2e}	End-to-end delay
$D_{acc-e2e}$	End-to-end delay after UP acceleration
τ	Transport network delay
$f(\tau)$	Function that maps transport network delay to cache size

$$D_{e2e} = D_{proc} + D_{queue} + D_{trans} + D_{prop}, \tag{1}$$

where D_{proc} is the time spent processing a packet in the CPU of a node (e.g., error checking, reading the header of the packet, etc.) D_{queue} is the time a packet spends in a queue at a node while waiting for other packets. D_{prop} is the time it takes for a signal to propagate through the communication media from one node to the next. Note that the proposed system does not affect the propagation delay. D_{trans} is the time required to transmit an entire packet into the communication medium and can be expressed as

$$D_{trans} = \frac{L}{R_t} \tag{2}$$

where L is the length of a packet in bits and R_t is the transmission rate in bits per time unit. Let D_{node} be the total delay, which includes the operations performed within the node and can be expressed as follows

$$D_{node} = D_{proc} + D_{queue} \tag{3}$$

Therefore, the sum of (2) and (3) gives the end-to-end delay in the normal operation of a node, which is

$$D_{e2e} = D_{node} + D_{trans} + D_{prop} \tag{4}$$

Let c represent the allocated size of the cache and $\mathcal{R}$ represent the dataset to be downloaded in one unit of time t_{unit}. Then $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ be the set of this dataset. Since the acceleration process speeds up the CPU processing operation within the node, the accelerated node processing delay can be expressed as

$$D_{acc-proc} = D_{proc} \times (1 - \sum_{i=1}^{n} \frac{P_c(r_i)}{P_c(\mathcal{R})}) \tag{5}$$

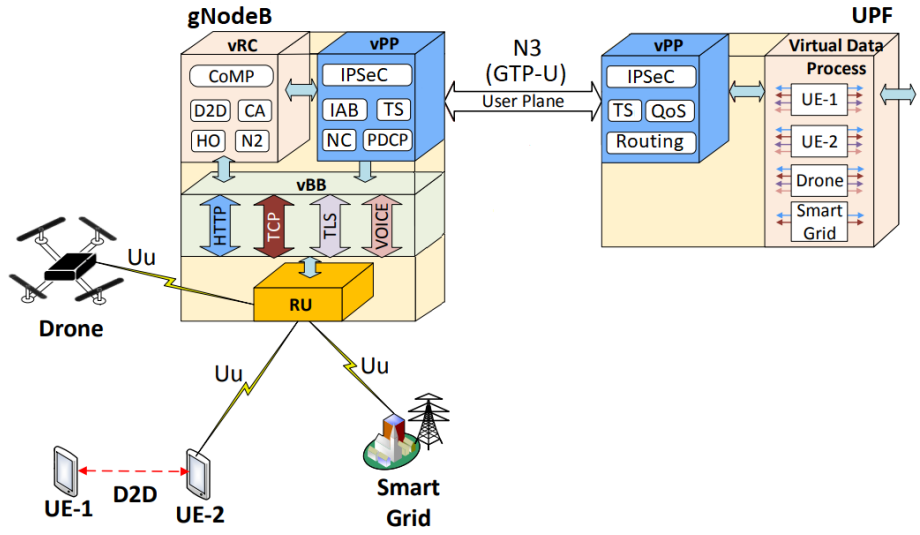

Fig. 2: Next-generation SBA of mobile network elements (UP perspective for the core network) without transport network protocol update.

where $\mathrm{P_c(r_i)}$ is the probability that the cache contains the i-th data in the set $\mathcal{R}$ (i.e. r_i) at the time t_{unit} of the request to download data and $P_c(\mathcal{R})$ is the probability that the cache contains the whole set $\mathcal{R}$ at the time t_{unit} of the request to download data.

Since $\mathrm{D_{queue}}$ defines Input/Output (I/O) processing in the network and the queuing in the node, the benefit of caching is saving CPU processing delay, and this is calculated in

$$D_{acc-node} = D_{acc-proc} + D_{queue} \tag{6}$$

The entire system includes both packet compression and acceleration. Packet compression produces packets of size $M < L$. Then the accelerated transmission delay is calculated as

$$D_{acc-trans} = \frac{M}{R} \tag{7}$$

Finally, we have an E2E delay after applying the proposed scheme to the UP, denoted by $D_{acc\text{-}e2e}$, which can be expressed as

$$D_{acc-e2e} = D_{acc-node} + D_{acc-trans} + D_{prop} \tag{8}$$

4 Architectural Design

4.1 Next-generation service-based mobile network architecture

Next-generation mobile systems are built in SBAs and placed in virtualized environments [31–35]. The main blocks and the microservices defined within these

blocks are shown in Fig. 2. The main aim of this architecture is to transform the monolithic architecture of the telecommunication system to a microservice architecture. On the gNodeB side, there are three main blocks and one Radio Unit (RU). The main blocks are Virtual BaseBand (vBB), Virtual Radio Control (vRC) and Virtual Packet Processor (vPP). There are also two blocks defined in UPF namely vPP and Virtual Data Processor (vDP). There will be many types of new services, such as vehicle-to-everything (V2X), smart grid, drones-enabled communication, etc. Users of these services will be connected to RU over Radio Frequency (RF) signals via the Uu interface. Each user can use one or more different types of services. The incoming RF signals are processed within vBB. In vBB, user traffic types are also identified and categorized by predefined Traffic Flow Template (TFT) classifiers of the services. The vRC is responsible for Control Plane (CP) signalling such as signalling for Device-to-Device (D2D), N2, Handover (HO), etc. and for controlling RAN features such as Coordinated Multipoint (CoMP), Carrier Aggregation (CA), etc.

The vPP is responsible for various types of packet processing operations. The microservices within the vPP are Network Coding (NC), Packet Data Convergence Protocol (PDCP), etc. for RAN side packet processing, Integrated Access Backhaul (IAB) for relay packet processing, IPSec for security processing and finally Transport Service (TS) for UP packet processing. The gNodeB communicates with UPF via a GTP-U tunnel. An important point in this context is that the UP protocol is still the GTP-U in this naive extension of SBA, although the architecture has evolved to a SBA (note that an evolved version of this architecture with backhaul link update protocol is shown in Fig. 3). The TS of vPPs is the service responsible for GTP-U processing. The GTP-U header is added to the GTP-U payload for Upload (UL) traffic and extracted to obtain the GTP-U payload for Download (DL) traffic. TSs in both vPPs are also responsible for establishing a GTP-U tunnel. The TS in the vPP of UPF receives/transmits the GTP-U packet. The vPP of UPF interacts with the vDP. The vDP performs session control and tracks the state of the session for one or more different types of traffic generated by the mobile users. The introduction of new entities such as vBB, vRC, and vPP in the proposed functional split for gNodeBs indicates a different architectural approach to accommodate cache-assisted acceleration in this paper. These entities represent virtualized components that perform specific tasks within the gNodeB architecture. By introducing these new entities, we aim to leverage cache-assisted acceleration techniques within the gNodeB architecture. It's also worth considering that different functional splits have also been standardized by organizations like 3GPP, Small Cell Forum, and O-RAN Alliance to address various network deployment scenarios and requirements. However, the proposed new entities in this paper is offering an alternative approach, specifically focusing on cache-aided acceleration. A comparison table between the proposed functions and the functional splits standardized by 3GPP/O-RAN is also given in Table 6.

4.2 Design Requirements

The proposed acceleration service in the SBA of mobile network nodes is shown in Fig. 3 with highlighted areas where the main components of the solution have been added. The service is aligned with the SBA defined at Release 15 by 3GPP which is

Table 6: Comparison of functional splits

Function	O-RAN/3GPP	Proposed Method
vBB	Centralized Unit (CU) or Distributed Unit	Virtual Baseband Unit
vRC	Radio Intelligence Controller	Virtual Radio Controller
vPP	VNF	Virtual Packet Processor
vDP	DUs and RRUs	Virtual Data Plane

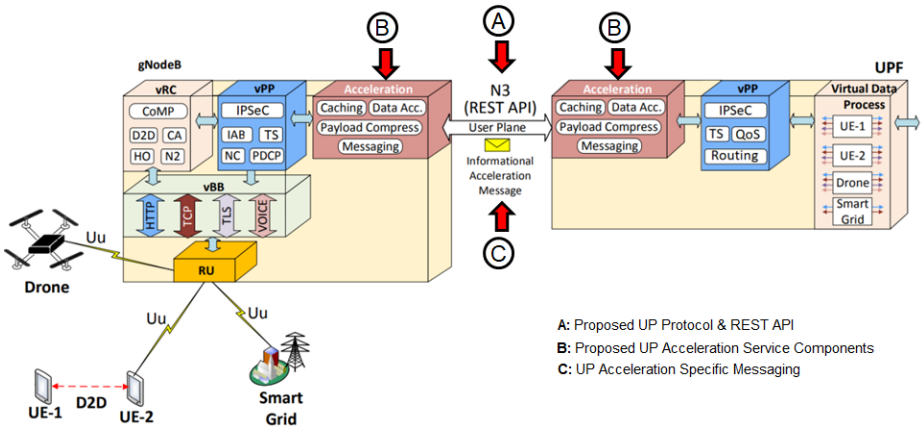

Fig. 3: The proposed acceleration service, REST API and information messages instead of GTP-U for UP transmission.

based on microservices, therefore relies on Representational State Transfer (REST) Application Programming Interface (API) for communication between gNodeB and CN. The acceleration component consists of four microservices: caching, data acceleration, payload compression and messaging microservices. The details of each of these microservices is given in Section 4.3.

User Plane Protocol: From the vPP point of view, the vBB or vDP task is to provide raw data to the vPPs mobile node (namely, the GTP-U payload). The GTP-U protocol provided by the TS is basically tasked with transmitting this raw data to the other node. If a Uniform Resource Locator (URL) is defined for each user and the vBB transmits the raw data to that user's URL, then the raw data can be delivered to the other mobile node via a POST operation. An example URL for a user can be held in the form of $<Cluster\ ID><gNodeB\ ID>/<User\ MSISDN>/<TEID>$. Cluster ID is assigned by the UPF to distinguish groups of gNodeBs. Mobile Station International Subscriber Directory Number (MSISDN) is a feature to distinguish UEs which is known to both the gNodeB and the UPF when the user is connected to the network. TEID which stands for Tunnel Endpoint Identifier is a unique identifier used in the GTP header to differentiate and identify different tunnels within a mobile network. By including the TEID in the URL, it is now possible to uniquely identify and handle different tunnels for a given user, taking into account their respective quality of service requirements.

In this study, a HTTP protocol is proposed to transfer UP data between mobile network elements. The purpose of the proposal to use HTTP as UP protocol can be summarized as follows;

- HTTP is well-defined and standardized. In addition, there have long been a variety of open source tools and sound industrial knowledge for HTTP-based data transmission.
- The UP data from the top vBB can be easily represented by binary data. HTTP can carry binary data [36] and HTTP messages can be sent using REST APIs.
- HTTP allows compression of payload data [36], which the currently used GTP-U protocol does not allow.
- HTTP allows caching [37].

REST API can be used to transfer data between two end nodes. The UP protocols that standardization committees are working on are protocols that can be transmitted through REST API. The reason for proposing a REST API for both the gNodeB and the UPF is to provide a structure that can also be used for European Telecommunications Standards Institute (ETSI) Management and Orchestration (MANO) [38,39] architecture [39]. The main idea is to allow exchange of payload data between client and server in a simple way. In addition, BSs and UPF can actually be considered Virtual Network Functions (VNFs). With ETSI MANO architecture, communication between VNFs is provided via REST API. Therefore, it is proposed to use the UP protocol over REST API so that the data can be transferred immediately via a POST method. GTP-U provides mechanisms for tunnel setup, maintenance, and termination, as well as other message types for various purposes within the user plane. Different message types that serve specific functions within the user plane are also included in GTP-U. A high-level design for mapping some GTP-U functionalities to REST API mechanisms are as follows:
Tunnel Setup: REST API, *Endpoint:* /tunnels, *HTTP Method:* POST, *Request Body:* Include necessary parameters for tunnel setup, such as source and destination endpoints, quality of service requirements, and any other relevant configuration parameters.
Data Transfer: REST API *Endpoint:* /tunnels/tunnel_id/data, *HTTP Method:* POST, *Request Body:* The payload of the HTTP request can contain the user data packets to be transferred. Additional headers or parameters can be included to specify the sequence number, packet size, or any other relevant information.
Echo Request/Response: REST API Endpoint: /echo, *HTTP Method:* POST or GET, *Request Body (if applicable):* Include any necessary information for the echo request. For example, the request body can contain the source and destination endpoints or any additional parameters. Response: The response to the echo request can be a standard HTTP response indicating the reachability status or any other relevant information.
Error Indication: REST API, *Endpoint:* /errors, *HTTP Method:* POST, *Request Body:* Include details about the encountered error or exceptional condition. This can be in the form of error codes, error messages, or any other relevant information.
Tunnel Termination: REST API, *Endpoint:* /tunnels/tunnel_id, *HTTP Method:* DELETE, *Response:* The response to the tunnel termination request can be a standard HTTP response indicating the success or failure of the termination process.

Caching Dynamic Content: Reverse proxies usually cache static content. However, user data changes dynamically. Data exchange between gNodeB and network is encapsulated by UP protocol and constantly changes. In some cases, the same content can not be requested multiple times from the cache content. New generation reverse proxies are capable of caching streaming data. Just as the streaming content is cached, the payload to be used for UP transmission can be transferred as bulk directly to the cache and transmitted instantly via the cache. In this way, each payload packet is encapsulated separately, thus saving the encapsulation processing time to be spent.

4.3 Micro-services

Caching: Caches are mostly for static content. However, user data from the vBB changes dynamically. The data encapsulated by the UP protocol is constantly changing. In some cases, the same content cannot be requested from the cache content more than once. An acceleration component should be able to cache streaming data. Just as the streaming content is cached, the payload to be used for UP transmission can be transferred in bulk directly to the cache and immediately transmitted over the cache [40]. In this way, each payload packet is encapsulated in its entirety to reduce the processing time for encapsulation. Data arriving at vPP is first cached when it reaches the proposed acceleration component. Then, this data is forwarded to the data de(compressor). Caching puts the content into the cache without requiring much CPU processing (such as I/O and with fewer function calls to UP header (de)encapsulation processes). In addition, the data cache can be cached in bulk. Then, pre-fetching could be applied to this bulk data in the cache so that the data can be sent more effectively. This ensures that most of the data is stored in Random Access Memory (RAM) and kept closer to the network I/O.

Fig. 4 shows various implementation of caching within the operating system (OS), namely Kernel space networking in Fig. 4a and user space networking of Fig. 4b. In kernel space networking, systems calls are used to interact with the kernel space. Kernel space consists of network sockets which are responsible for physical and logical communication with the network. User space networking on the other hand serves as application in user space and contains network socket commands. This user space application ensures the communication between the network interface and sockets. A dedicated memory called caching service is required for this application. Larger memory ensures larger cache space and better network application capabilities. Some user-space networking principles such as Data Plane Development Kit (DPDK) can be used by bypassing the kernel during UP packer delivery process. Depending on the service-based design, user-space networking can also be offered as a service (e.g., as a Cloud Native Function (CNF)).

Data Acceleration: This micro-service is used to convert packets that are actually large into many smaller packets with a certain maximum size. The reason for this is that the transmission of the small packets over the network takes less time compared to the large packets, and the packet size can be determined based on the delay and changed in an customizable way.

Payload Compression: The amount of data to be transmitted can be (de)compressed at certain rates, where the (de)compression of the payload data is the second pro-

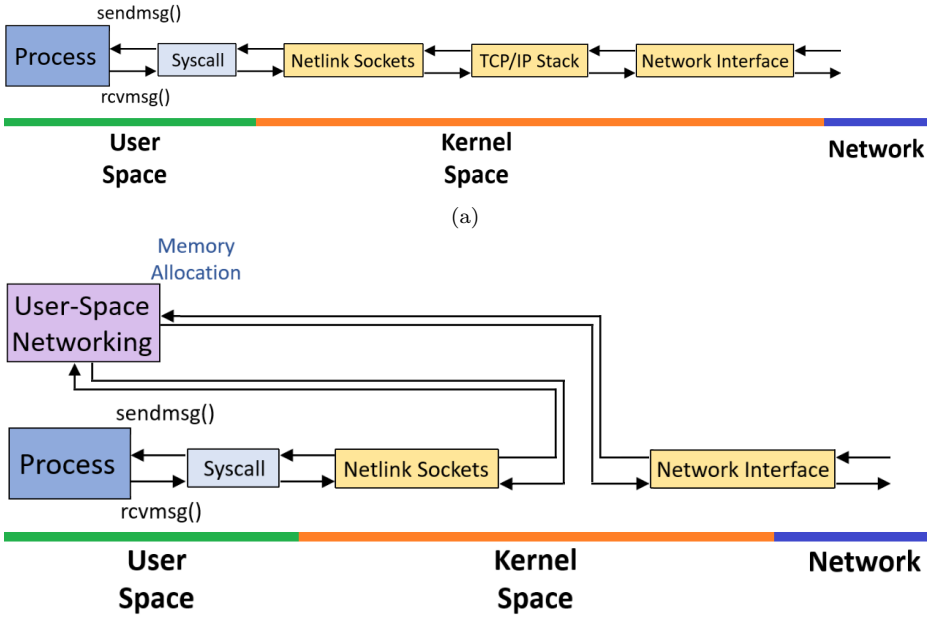

Fig. 4: Different caching implementations within the OS in networking a) Kernel-space networking b) User-space networking.

cessing step within the proposed acceleration component. This results in increased bandwidth and shorter data transmission time. The protocols on which Standard Development Organizations (SDOs) are working support compression.

To avoid unsuccessful data flow in UP traffic, a data compression method should be used that provides minimal data loss in a lossless data compression algorithm [41]. Then, the effects of delay are reduced when the amount of data sent in a unit time is increased. The compressed payload must be decompressed on the opposite node of the communication. These two processes must be performed for both DL direction and UL traffic in coordination between nodes by mutual message exchanges. Compression creates smaller UP packets, which reduces the transmission time of UP packets. The process creates more small packets for the content, reducing the network I/O processes. The packet loss of the smaller packets in the network has less impact on the user traffic.

Messaging: The caching, compression and data formation processes in the reverse proxy will provide acceleration to the UP, but these will also add additional computation requirements (extra RAM and CPU load) to the existing system. Therefore, if there is no problem with backhaul traffic and data transfer for next generation services is done in a robust and uninterrupted manner, reverse proxy can be bypassed. Both UL and DL traffic are affected by delays when there is an issue that increases latency in the mobile backhaul.

The reason for messaging is to detect performance degradation between nodes. Then, the compression ratio and cache size can be changed dynamically by checking the performance of the transport link. The performance of the backhaul link

can be measured by measurement protocols such as IP-Service Level Agreement (SLA) [42] or Two-Way Active Measurement Protocol (TWAMP) [43], which can be used to test backhaul performance. These performance tests can be evaluated as an embedded feature in mobile network elements. The results of the backhaul performance tests are then passed to the acceleration software to make decisions about acceleration operation. The rate of caching may vary depending on the current CPU, memory and cache utilization of the end nodes. A message that measures instantaneous backhaul performance, including fast start/end commands, and session information can be performed between UPF and gNodeB. The acceleration message can be transmitted in protocols such as Internet Message Access Protocol (IMAP) [44], MQ Telemetry Transport (MQTT) [45] or can be completed over REST API.

4.4 Acceleration Procedure

In cases, when there is no performance degradation in the backhaul link, the opposite end node is aware of this situation, so there is no need to register for an acceleration service in the vPP of gNodeB and in the UPF for the UP packets. If there is a delay problem in the backhaul link, UP packets are forwarded to the acceleration service and sent to the opposite end node through REST API. In addition, the UP sessions on both mobile nodes are stored in the TS session database. The UP packets are subjected to compression before transmission and decompressed on the opposite node. On the opposite node, the accelerated traffic is merged. The acceleration algorithm is summarized in Algorithm 1. The algorithm starts with **an initialization step** in line 2, where the cache size and other variables are set. The cache size is determined based on the calculated delay of the transport network and a counter variable is initialized to track the number of accelerated data packets. An array is also initialized to store the invalidated allocated cache. The algorithm then enters the **UP Acceleration loop** in line 6, which continues until the counter variable reaches the cache size. In each iteration of the loop, a data packet is read from a set named R and stored in a temporary variable. The compressed packet is then placed into the cache by adding it to the existing cache data. The counter variable is incremented to keep track of the number of packets processed. **After the UP Acceleration loop completes** in line 11, the final state of the accelerated packets is stored in a variable named c. This represents the compressed packets ready for transmission. **Finally**, the algorithm returns the value of c as the output, providing the accelerated packets that can be sent over the network. In this algorithm, τ is computed from the calculated transport network delay between the BS and CN, and f(τ) maps the transport network delay to the cache size. Note that the delay of the transport network can be due to several reasons such as the number of connected UEs to BS, traffic congestion on routers, processing delays in the core network, etc. The complexity of the algorithm depends on the number of iterations in the UP Acceleration loop. Since the loop continues until the counter variable reaches the cache size, the complexity is $\mathcal{O}(Cache\_Size)$, where $Cache\_Size$ is the size of the cache determined based on the delay of the transport network.

Algorithm 1 UP Acceleration

Input: f(τ)
Output: accelerated packets

procedure ACCELERATION()
  Initialization:
  $Cache\_Size \leftarrow f(\tau)$ ▷ determine the cache size based on calculated transport network delay
  $Counter \leftarrow 0$ ▷ keep track of the number of accelerated data packets
  $c \leftarrow [0]$ ▷ invalidate allocated cache
  UP Acceleration:
  while $Counter < Cache\_Size$ **do**
    $m_i \leftarrow r_i$ ▷ read data r_i in set R
    $c \leftarrow m_i + c$ ▷ place the compressed packet to the cache
    $Counter \leftarrow Counter + 1$
  end while
  return c ▷ final state of the accelerated packets to be transmitted
end procedure

5 Numerical Evaluation

The aim of the numerical evaluation is to show the performance gain obtained by the proposed acceleration method. Since the study in this paper proposes to improve the UP traffic, only the part between the TS of gNodeB and the TS of UPF is considered and shown in the test environment. For comparisons of HTTP with acceleration, we use GTP-U protocol defined in [18], and HTTP without acceleration defined in [22]. The traffic we are using in our simulation is synthetic data. We performed the transfer of traffic from gNodeB to UPF to trigger the existing structured GTP-U and our proposed HTTP. When creating traffic, a file of a certain size is transmitted as a stream. File sizes are sent as specified so that measurements are accurate. The traffic is thus encapsulated with GTU-U or HTTP as in our proposed method. The Apache 2 server is running the protocol defined in Section 4.2. This protocol specifies the procedures required to establish an HTTP/REST connection that will resemble the operation of a GTP tunnel.

5.1 Details of the Testbed

For numerical tests, Graphical Network Simulator-3 (GNS3) [46] is used, which provides an emulation environment very similar to the actual network environment. The test environment is shown in Fig. 5. There are four Virtual Machines (VMs) in the test and all of them are installed with Ubuntu Linux, which has 2GB RAM. VMs on the left-side of Fig. 5 represent the TSs of gNodeBs and VMs on the right-side of Fig. 5 represent TSs of UPFs. All VMs are connected to the routers via an Ethernet interface. The routers are used to create a backhaul environment. The routers in the backhaul environment are configured similarly to devices in a real IP/MPLS network. Open Shortest Path First (OSPF) was used as the routing protocol and MPLS was enabled in Label Distribution Protocol (LDP) mode. All interconnecting links have a maximum bandwidth of 100 Mbps. There are three hops between gNodeBs and UPF nodes.

The TS of gNodeB-1 transmits data to TS of UPF-1 via REST API. TS of gNodeB-2 communicates with TS of UPF-2 via the use of GTP-U tunnelling. For

communication via GTP-U tunnelling, a GTP-U implementation [47] is used for both TS of gNodeB-2 and TS of UPF-2. The specified GTP-U entities indicate this kernel-level implementation. Thus, all packets in GTP-U are transmitted encapsulated between TS of gNodeB-2 and TS of UPF-2 creating a true UP path. The VM representing TS of gNodeB-1, located in the upper left corner of Fig. 5, is installed with an Apache2 [48] web server. A continuous RESTful POST operation is configured to send the data in an encapsulated HTTP format to the TS of UPF-1. Two tests are performed between these two VMs. One is HTTP with acceleration and the other is HTTP without acceleration. For the test of HTTP without acceleration, Apache2 server memory caching was disabled. The Varnish-cache [49] software, a web accelerator, is installed in front of the web server and is responsible for performing the entire acceleration procedure (caching operations, (de)compression, and decomposition of data into small packets). A local script is configured to transfer the generated traffic directly to the Varnish-cache.

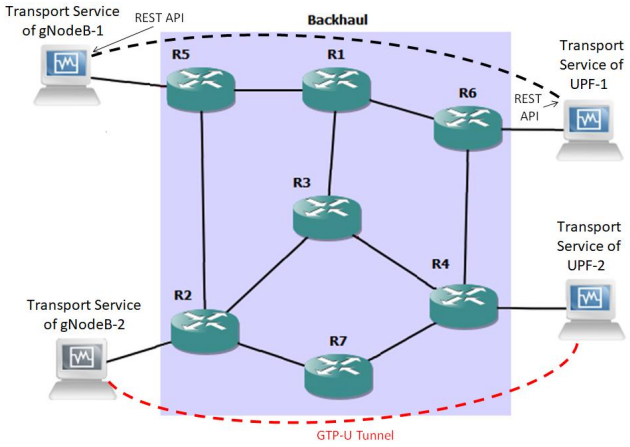

Fig. 5: Test architecture in GNS3 emulation environment with three virtual machines connected via backhaul routers.

In all numerical tests, the compression ratio for the accelerated HTTP implementation is 0.03% and compression is performed using the lossless compression algorithm defined in [50]. Latency is calculated as the time between the generation of a packet from one end node and its reception from the opposite end node, as an average of all packets in the same connection session. Thus, the compression latency is added to the total acceleration time. To overcome the potential bottlenecks of the virtualized environment, the same VM is cloned as four VMs to have the same VM configuration during the emulation. In addition, the generated traffic is kept relatively small according to the capacity of the emulated network to remove the limitations of emulation environment. Finally, the confidence of the results is verified using the state of the art studies presented.

5.2 Performance Results

Fig. 6 shows the relationship between the transfer time of the same data using GTP-U, HTTP, and HTTP for a cache size of 20MB. The transfer time is measured as the total time taken between the TSs of two nodes for the data and is given in seconds. As shown in Fig. 6, GTP-U is on average 35% slower than HTTP. The difference between GTP-U and HTTP increases as the data size increases. The reason for this difference depends on two factors. First, numerical tests have shown that GTP-U incurs an additional overhead of 7.2%, which is slightly lower than the expected state of the art [18], while HTTP incurs only 1.2% percent additional overhead which is in line with the state of the art [22]. Sending larger packets and each additional packet sent to the network inevitably adds delay. The second factor is the GTP-U encapsulation time required by the CPU of the VM. On the other hand, a HTTP cache with a size of 20 MB provides a 9.5% acceleration in data transfer time. Since the cache range is fixed at 20 MB, the cache hit time inevitably decreases when the data size exceeds 20 MB, so the performance improvement from the cache decreases somewhat. When sending 10 MB data, 14.5% and when sending 20 MB data, a 12.5% acceleration in transmission time was achieved. An average increase of about 9% is achieved when the data sent is larger than 20 MB.

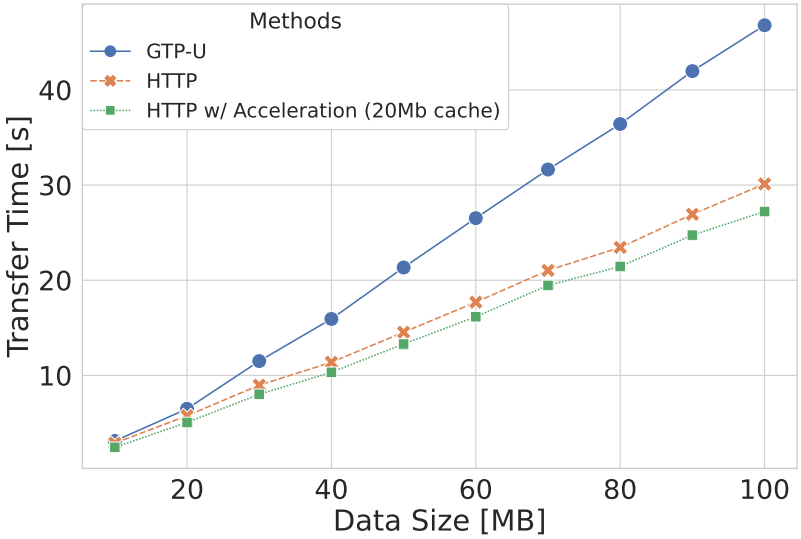

Fig. 6: Transfer time of the same amount of data via GTP-U, HTTP and HTTP with acceleration (20 MB cache).

Fig. 7 shows the generated throughput when the same amount of data is sent over GTP-U, HTTP and HTTP with acceleration implemented with different cache sizes. The data to be transmitted is kept constant at 100 MB and the

delay in the links is zero. The inserted trend line shows the moving average of two consecutive values for tracking. As can be seen, GTP-U achieved a throughput of about 17 Mbps and operated about 40% slower than HTTP without cache (not accelerated). The larger the cache size of HTTP, the higher the throughput rate. A cache size of 100 MB generates traffic about 33 Mbps. The throughput between HTTP with 100 MB cache and HTTP without cache increases by about 26%. With the cache enabled, a speed increase of about 10% is achieved. This becomes clear when HTTP without cache is compared with HTTP for 10 MB cache. Although it is always advantageous to use a large amount of cache, the choice of 20 MB cache can be considered optimal considering the resources that the gNodeB spends on other processes. Note that the main purpose of the gNodeB and the area where it consumes most of its resources are, of course, the processes of the users coming from radio to vBB and not the transport (namely UP).

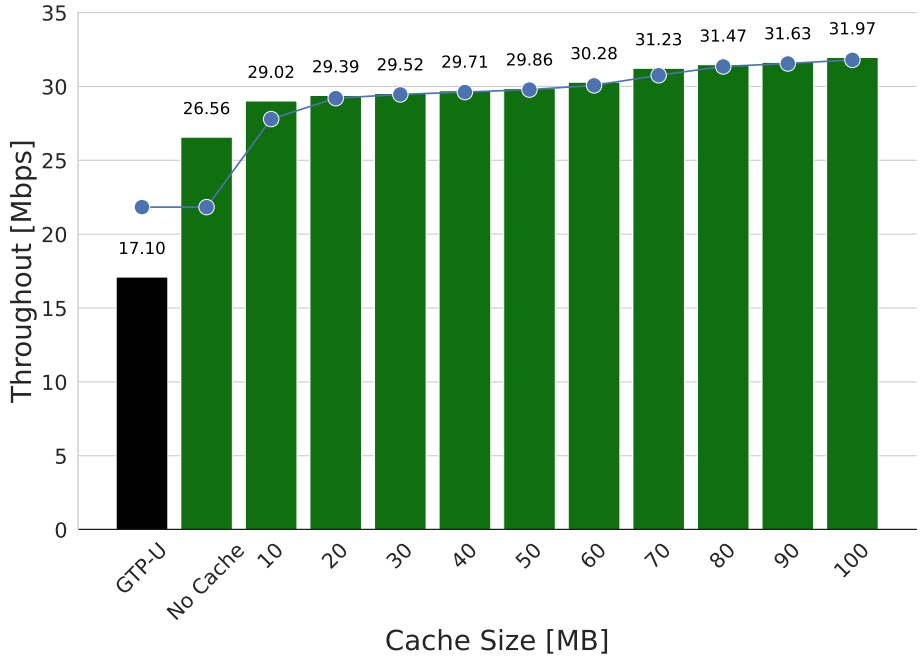

Fig. 7: Calculated throughput for transferring the same data over GTP-U, HTTP and HTTP with acceleration (increasing cache sizes).

Fig. 8 shows the throughput of the three evaluated implementations at different delay times. The transferred data size is again 100 MB and the cache size allowed for acceleration is fixed at 20 MB. As the delay time increases from 40 ms, the downward trend in throughput slows down for all three implementations. This was found to be consistent with the equation 1 and increasing the delay beyond a certain point has less impact on throughput. HTTP with acceleration provides the highest throughput in Fig. 8 for all the delay values. Compared to HTTP, HTTP with acceleration has 11% more throughput on average. On the other hand, HTTP

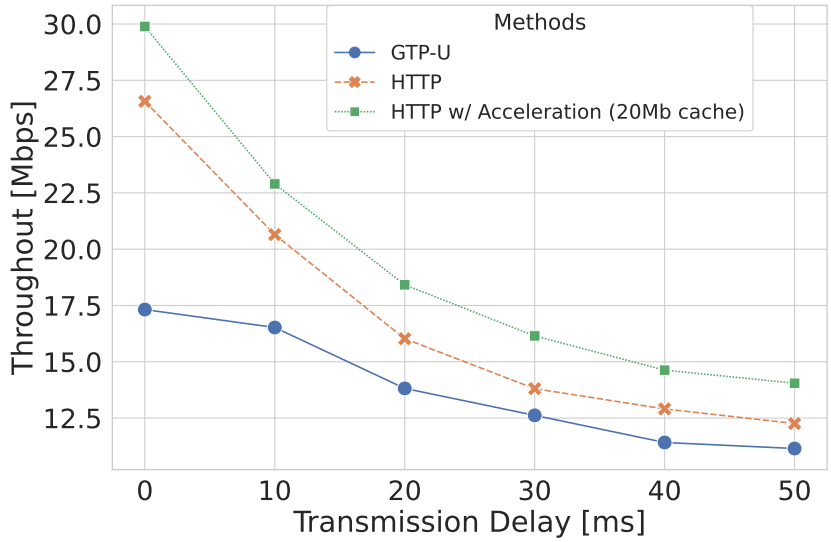

Fig. 8: Throughput with different transmission delay of GTP-U, HTTP and HTTP with acceleration (20 MB cache).

appears to be 9% faster on average than GTP-U. However, the throughput of GTP-U and the other two implementations decreases as the delay increases. The reason for this is that as the transmission delay time increases, the effect on the total E2E delay in (1) also increases, and thus more negative impact on throughput is obtained.

Summary of Improvements:

Table 7 represents the summary of percentage improvements in transfer time for the HTTP and HTTP with Acceleration (20Mb cache) methods compared to the GTP-U method for different data sizes in MB.

Table 7: Transfer time reductions (%) with respect to GTP-U with increasing data size [MB].

Data Size [MB]	HTTP	HTTP w/ Acceleration (20Mb cache)
10	-7.60%	-21.14%
20	-7.76%	-28.74%
30	-9.24%	-34.11%
40	-9.85%	-32.47%
50	-12.91%	-38.23%
60	-17.85%	-26.89%
70	-19.81%	-34.30%
80	-24.22%	-36.97%
90	-28.28%	-40.34%
100	-33.51%	-41.79%

Table 8 represents the summary of the percentage improvements in throughput for different cache sizes of HTTP (without cache) and HTTP with Acceleration methods with respect to the GTP-U method.

Table 8: Throughput performance improvements (%) with respect to GTP-U with increasing cache size.

Cache Size [MB]	Throughout Improvement
No Cache	55.28%
10	69.16%
20	71.16%
30	72.01%
40	72.55%
50	73.33%
60	74.39%
70	74.89%
80	75.29%
90	75.58%
100	76.36%

Table 9 shows represents the percentage improvements in performance for the HTTP and HTTP with Acceleration (20Mb cache) methods compared to the GTP-U method for different transmission delay values.

Table 9: Throughput performance improvements (%) with respect to GTP-U with increasing transmission delay [ms].

Transmission Delay [ms]	HTTP	HTTP w/ Acceleration (20Mb cache)
0	+53.39%	+72.87%
10	+25.35%	+39.61%
20	+15.35%	+33.79%
30	+9.95%	+28.18%
40	+13.08%	+28.58%
50	+10.47%	+25.97%

5.3 Observed Challenges

Caching Size and Invalidation: Dynamic content is generated by scripts that change the content. The cache allocated for UP acceleration is actually covered by the memory requirements on the mobile node. The amount of cache allocated here is proportional to the bandwidth used. In other words, the amount of cache should be increased when more bandwidth is used. If bandwidth is low, it is more efficient to use the memory allocated for the cache for the mobile node's own system requirements than to leave this extra cache unused. The cache content must be updated as long as the data stream continues. Therefore, the cache content should be invalidated at regular intervals. In our tests, we found that this process

should run as *Bandwidth Aware Cache Invalidation*, depending on the traffic of the gNodeB or UPF. Frequent cache invalidation queries may cause software problems on UPF, which is more multithreaded than gNodeB. Note that if cache can be used in UP, a limited or no cache must be implemented on the RAN side, which is a tradeoff for MNO for gNodeB operations.

Encrypted Traffic: By encrypting the received data in the cache, the caching process runs the risk of corrupting the provided integrity. This is a known problem, in which case the cache size can be kept smaller or the UP traffic can be transmitted only over HTTP without cache to combat this effect.

Operating System: The software-based cache stores user plane data in the virtual memory of the node. The decision of what to store in this cache is given by the operating system. Therefore, additional internal software function calls are also required to cache streaming data. In this way, data coming from the vBB can be created and cached after streaming. Furthermore, the performance limitations of the OS kernel network cause another separate performance problem, since OS is also responsible for handling the requests coming from the vBB.

Quality of Service Issues: The real-time user experience (especially for premium users) can be affected by the misconfigured the quality-of-service (QoS) settings [51]. The same issue exists with the proposed acceleration service. The correct identification of the used cache size for different user types must be predefined by the MNO. The same problem can occur in the network slicing or RAN sharing [52] use cases. Therefore, MNOs must agree on the cache size that will be used for their users beforehand depending on the considered use case.

6 Conclusions and Future Work

In this paper, we propose a cache-aided acceleration method for transport problems in the next generation of mobile networks. This method can be used to integrate next generation RAN and CN with the requirements of software changes, i.e., a plug & play approach and reduce the dependency of MNOs on the quality of transport networks. The proposed solution uses the HTTP protocol between the gNodeB and the CN in cellular networks to contribute to the studies that search alternative and possible replacements of the current UP protocol. Due to several incapability of GTP-U protocol, HTTPv3 protocol is used as an alternative solution that is also compatible with User Datagram Protocol (UDP) traffic. The features such as compression, low overhead, caching, etc, that give flexibility to a protocol can be achieved with HTTPv3, which the currently used GTP-U lacks. The results of the numerical tests show a performance increase of about 35% when using HTTP. Caching provides an additional acceleration of about 9.5%. Unlike many previous studies on caching in mobile networks, caching in this paper is not implemented on the RAN or transport network side, but on the UP protocol side. The challenges encountered during the numerical tests are explained using the detailed points. In particular, at points where E2E latency is high, caching must be dynamically enabled to contribute to the operation of the HTTP based UP protocol. This can be evaluated as a future work. This study can also be extended, especially in cellular network scenarios that use Wi-Fi offloading, by accelerating traffic between the Wi-Fi Access Point (AP) and the CN or between

the Wi-Fi controller and the CN, using the HTTP protocol, depending on the deployment scenario.

Compliance with Ethical Standards

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Funding

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO)—Program UNICO I+D funded by MCIN/AEI/ 10.13039/501100011033 under Grant TSI-063000-2021-54, Grant TSI-063000-2021-55, "ERDF A way of making Europe" project under grant PID2021-126431OB-I00.

References

1. Kaveh Ahmadi, S. Pourya Miralavy, and Mona Ghassemian. Software-defined networking to improve handover in mobile edge networks. *International Journal of Communication Systems*, 33(14):e4510, 2020.
2. Engin Zeydan, Omer Dedeoglu, and Yekta Turk. Performance monitoring and evaluation of fttx networks for 5g backhauling. *Telecommunication Systems*, 77:399–412, 2021.
3. Yekta Turk and Engin Zeydan. An experimental measurement analysis of congestion over converged fixed and mobile networks. *Wireless Networks*, 26:1017–1032, 2020.
4. Yekta Turk and Engin Zeydan. Hubble: An optical link management system for dense wavelength division multiplexing networks. *Turkish Journal of Electrical Engineering & Computer Sciences*, 28:743 – 756, 2020.
5. 3GPP Technical Specification. Service requirements for the 5G system. *3GPP TS 22.261 V16.6.0*, 2018.
6. Yekta Turk and Engin Zeydan. A dynamic replication scheme of user plane data over lossy backhaul links. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2019.
7. 3GPP Technical Specification. Technical Specification Group Services and System Aspects; Security architecture and procedures for 5G system (Release 16). *3GPP TS 33.501 V16.0.0*, 2019.
8. J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Performance enhancing proxies intended to mitigate link-related degradations. *IETF RFC 3135*, 2001.
9. E. Zeydan and Y. Turk. On the Impact of Satellite Communications over Mobile Networks: An Experimental Analysis. *IEEE Transactions on Vehicular Technology*, 68(11):11146–11157, 2019.
10. Adax. Adax FastPlane. `http://www.adax.com/wp-content/uploads/2018/11/Adax-FastPlane-v1.pdf`, 2016. [Online; accessed 10-Sept-2020].

11. Ahmad, Ijaz and Suomalainen, Jani and Porambage, Pawani and Gurtov, Andrei and Husko, Jyrki and Höyhtyä, Marko. Security of satellite-terrestrial communications: Challenges and potential solutions. *IEEE Access*, 10:96038–96052, 2022.
12. Jeong, Jaehoon and Shen, Yiwen and Oh, Tae and Céspedes, Sandra and Benamar, Nabil and Wetterwald, Michelle and Härri, Jérôme. A comprehensive survey on vehicular networks for smart roads: A focus on IP-based approaches. *Vehicular Communications*, 29:100334, 2021.
13. Jiang, Peipei and Wang, Qian and Huang, Muqi and Wang, Cong and Li, Qi and Shen, Chao and Ren, Kui. Building in-the-cloud network functions: Security and privacy challenges. *Proceedings of the IEEE*, 109(12):1888–1919, 2021.
14. Yekta Turk, Engin Zeydan, and Cemal Alp Akbulut. On performance analysis of single frequency network with c-ran. *IEEE Access*, 7:1502–1519, 2019.
15. Haque, Md Emdadul and Tariq, Faisal and Khandaker, Muhammad RA and Wong, Kai-Kit and Zhang, Yangyang. A Survey of Scheduling in 5G URLLC and Outlook for Emerging 6G Systems. *IEEE Access*, 2023.
16. Mamane, Asmae and Fattah, Mohammed and El Ghazi, Mohammed and El Bekkali, Moulhime and Balboul, Younes and Mazer, Said. Scheduling algorithms for 5G networks and beyond: Classification and survey. *IEEE Access*, 10:51643–51661, 2022.
17. Yekta Turk, Engin Zeydan, and C. Akbulut. Experimental performance evaluations of comp and ca in centralized radio access networks. *Telecommunication Systems*, 72:115–130, 2019.
18. N. Varis, J. Manner, and J. Heinonen. A layer-2 approach for mobility and transport in the mobile backhaul. In *2011 11th International Conference on ITS Telecommunications*, pages 268–273, 2011.
19. S. Homma, T. Miyasaka, and D. Matsushima, S.and Voyer. User Plane Protocol and Architectural Analysis on 3GPP 5G System. *IETF draft-hmm-dmm-5g-uplane-analysis-00*, 2018.
20. Z. Sarker. QUIC – a vehicle for transport protocol evolution. `https://www.ericsson.com/en/blog/2018/6/quic--a-vehicle-for-transport-protocol-evolution`, 2018. [Online; accessed 20-Aug-2020].
21. Bogineni, K et al. Optimized mobile user plane solutions for 5g. *Internet-Draft*, 2018.
22. H. F. Nielsen. Simple test of amount of system calls in jigsaw. `https://www.w3.org/Protocols/HTTP/Performance/System/SysCalls.html`, 1997. [Online; accessed 24-Sept-2020].
23. IEEE 802.1CM-2018. Time-sensitive networking for fronthaul. *IEEE Standard for Local and metropolitan area networks*, 2018.
24. Hirschman,B. et al. High-performance evolved packet core signaling and bearer processing on general-purpose processors. *IEEE Network*, 29(3):6–14, 2015.
25. Ahluwalia, S. et al. Acceleration of GTP traffic flows, over a satellite link, in a terrestrial wireless mobile communications system. *US Patent No. US2016/0192235A1*, issued on August 21, 2018.
26. Y. Hechthagay et al. Methods and apparatus for optimizing tunneled traffic. *US Patent No. US2015/10057391B2*, issued on November 10, 2018.
27. K. Liu and J. Y. B. Lee. On Improving TCP Performance over Mobile Data Networks. *IEEE Transactions on Mobile Computing*, 15(10):2522–2536, 2016.
28. K. Liu et al. Joint Upload-Download TCP Acceleration over Mobile Data Networks. In *14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2017.
29. Engin Zeydan, Ejder Bastug, Mehdi Bennis, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, and Mérouane Debbah. Big data caching for networking: Moving from cloud to edge. *IEEE Communications Magazine*, 54(9):36–42, 2016.
30. Ejder Baştuğ, Mehdi Bennis, Engin Zeydan, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, and Mérouane Debbah. Big data meets telcos: A proactive caching perspective. *Journal of Communications and Networks*, 17(6):549–557, 2015.
31. Khichane, Abderaouf and Fajjari, Ilhem and Aitsaadi, Nadjib and Gueroui, Mourad. Cloud Native 5G: an Efficient Orchestration of Cloud Native 5G System. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2022.
32. J. B. Moreira et al. Next generation of microservices for the 5G service-based architecture. *Int. J. of Net. Mng.*, 8:1–22, 2020.
33. O-RAN Alliance. O-ran use cases and deployment scenarios. `https://www.o-ran.org/resources`, 2020. [Online; accessed November-2020].

34. Mao Yang, Yong Li, Bo Li, Depeng Jin, and Sheng Chen. Service-oriented 5G network architecture: an end-to-end software defining approach. *International Journal of Communication Systems*, 29(10):1645–1657, 2016.
35. Engin Zeydan, Josep Mangues-Bafalluy, Jorge Baranda, Manuel Requena, and Yekta Turk. Service based virtual ran architecture for next generation cellular systems. *IEEE Access*, 10:9455–9470, 2022.
36. R. Fielding and J. Reschke. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. *IETF RFC 7230*, 2014.
37. R. Fielding, M. Nottingham, and J. Reschke. Hypertext Transfer Protocol (HTTP/1.1): Caching. *IETF RFC 7234*, 2014.
38. ETSI. Network Functions Virtualisation (NFV);Management and Orchestration. *GS NFV-MAN 001 V1.1.1 (2014-12)*, 2014.
39. Engin Zeydan, Josep Mangues-Bafalluy, and Yekta Turk. Intelligent service orchestration in edge cloud networks. *IEEE Network*, 35(6):126–132, 2021.
40. Engin Zeydan, Yekta Turk, and Baris Berk Zorba. Enhancing the capabilities of mobile backhaul: A user plane perspective. *Journal of Network and Systems Management*, 30(31), 2022.
41. Pedrycz, Witold. Granular Data Compression and Representation. *IEEE Transactions on Fuzzy Systems*, 2022.
42. Chiba, M et al. Cisco service-level assurance protocol. *IETF RFC 6812*, 2013.
43. Hedayat, K et al. A two-way active measurement protocol (twamp). *IETF RFC 5357*, 2008.
44. M. Crispin. Internet message access protocol - version 4rev1. *IETF RFC 3501*, 2003.
45. A. Stanford-Clark and A. Nipper. MQTT Version 3.1.1. *OASIS Standard*, 2014.
46. GNS3. GNS3 v2.1.11. `https://www.gns3.com/`, 2018. [Online; accessed 2-Sept-2020].
47. Osmocom. Linux kernel gtp-u. `https://osmocom.org/projects/linux-kernel-gtp-u/wiki`, 2016. [Online; accessed 4-Sept-2020].
48. The Apache HTTP Server Project. Apache2 2.4.4.1. `https://httpd.apache.org/`, 201p. [Online; accessed 7-Sept-2020].
49. P. Henning. Varnish HTTP Cache. `https://varnish-cache.org/`, 2016. [Online; accessed 7-Sept-2020].
50. P Deutsch. Deflate compressed data format specification version 1.3. *IETF RFC 1951*, 1996.
51. Engin Zeydan, Josep Mangues-Bafalluy, Omer Dedeoglu, and Yekta Turk. Performance comparison of qos deployment strategies for cellular network services. *IEEE Access*, 8:176073–176088, 2020.
52. Yekta Turk and Engin Zeydan. On performance analysis of multioperator ran sharing for mobile network operators. *Turkish Journal of Electrical Engineering & Computer Sciences*, 29(2):816–830, 2021.