SUPPORTING OPERATIONAL MANAGEMENT OF IP-BASED VOICE SERVICES

Ph.D. Dissertation

Author:
Tamás Tóthfalusi
Software Information Technology MSc

Supervisor:
Dr. Péter Orosz
associate professor

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS
DEPARTMENT OF TELECOMMUNICATIONS AND MEDIA INFORMATICS
DOCTORAL SCHOOL OF INFORMATICS
2021.

Alulírott, Tóthfalusi Tamás kijelentem, hogy ezt a doktori értekezést magam készítettem és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

A dolgozat bírálatai és a védésről készült jegyzőkönyv a későbbiekben, a Budapesti Műszaki és Gazdaságtudományi Egyetem dékáni hivatalában lesznek elérhetőek.

Budapest, 2021. április 13.

. . . . . .

Magyar nyelvű összefoglaló

A telekommunikációs hálózatok folyamatos fejlődésének következtében (2G/3G/4G/5G) az infrastruktúra egyre bonyolultabbá vált. Az operátoroknál nem ritka, hogy a különböző generációk párhuzamosan működnek egymás mellett. Habár ezek logikailag önálló hálózatok, közöttük együttműködést kell biztosítani. Ez abból fakad, hogy a mobilkészülékek bármely hálózatból bármely hálózatba indíthatnak hívást, vagy átviteli limitációk miatt akár hívás közben is technológiát válthatnak.

Az új generációk új protokollok és többszörös protokollbeágyazások megjelenését is jelentik egyre nagyobb átviteli sebességen (akár 100 Gbit/s). Ebben a környezetben az üzenetek elemzésének módszertana felveti a hardver alapú megoldások igényét. A legtöbb megoldás azonban nem tér ki a többszörös beágyazásokra, az újrakonfigurálhatóságra (gyakran változó protokollok), illetve csak 5 paramétert dekódol az üzenetből (IP címek, portok, protokoll).

A VoLTE (Voice over LTE) szolgáltatás az első olyan mérföldkő, ahol mind a vezérlőforgalom, mind pedig a hangmintákat szállító adatforgalom IP-feletti átvitelt alkalmaz. Ez a technológia önmagában is összetett: használja az LTE és az IMS rendszer elemeit is. A magas szolgáltatásminőség folyamatos biztosításához a korábbi generációknál (2G/3G) elegendő volt a vezérlősík vizsgálata, mivel az áramkörkapcsolt adatsík megfelelő átviteli garanciákat nyújtott. Ezzel ellentétben VoLTE esetén végponttól végpontig IP protokoll felett történik a hangátvitel, ezért a szolgáltatásmenedzsmentben megjelennek az IP hálózatok jól ismert problémái (adatvesztés, újraküldés, átrendeződés). Ezek mind hatással vannak az előfizető által tapasztalt szolgáltatásminőségre, így elengedhetetlen a vezérlősík elemzése mellett a hangminták útvonalának vizsgálata. Az IP-alapú technológia korábban ismeretlen volt az operátorok számára, az új, dinamikus tartalmú és méretű vezérlőüzenetek megjelenésével pedig bonyolultabbá vált a hívásfelépítési üzenetszekvenciák követése. Az üzemeltetett rendszer magasabb szintű elemzéséhez elengedhetetlen az IMS specifikus hibaesemények felismerése, valamint megkülönböztetése a normál híváshoz tartozó üzenetszekvenciáktól. Az adatsík átviteli garanciáinak ellenőrzése üzenetszintű metrikákkal vagy szolgáltatásminőség becsléssel történhet. A jelenlegi metrikák azonban nem fedik le a telekommunikációs hálózatok viselkedését, és nem különítik el élesen a mobilkészülék szempontjából az üzenetvesztés és üzenetátrendeződés fogalmát. Az előfizetői jogok miatt a referenciát használó modellek alkalmazása ebben a környezetben pedig nem lehetséges.

Kutatómunkám során a VoLTE rendszer vizsgálatával, és az előzőekben ismertetett, operátori szemszögből fontos problématerületekkel foglalkoztam (többszörösen beágyazott protokollok előfeldolgozása, vezérlősík elemzése, adatsíkban utazó hanghívások elemzése), ugyanis jelenleg nincs olyan átfogó módszertan, amely globális képet ad az üzemeltetett VoLTE szolgáltatásról. A disszertációmban olyan új módszereket és metrikákat mutatok be, amelyek nagyban segítik az operátorok mindennapi munkáját. Eredményeimet három téziscsoportra bontottam. Az első téziscsoport a többszörösen beágyazott protokollok hardveralapú elemzésének témakörét vizsgálja, amely két tézist tartalmaz. A problématerület vizsgálatának eredményeként bemutatom a hardveralapú protokoll elemzés követelményeit és limitációit. A téziscsoportban ismertetek egy olyan új módszert, amely segíti a megtervezését egy nagysebességű, többszörösen beágyazott protokoll fejlécet kezelő, tetszőleges paramétert dekódoló, FPGA-alapú, veszteségmentes protokoll elemző modulnak. A bemutatott módszer alapján megterveztem egy 100 Gbit/s átviteli sebességű, 8 különböző fejlécet felismerő

protokoll dekóder modult, amely 14 előre definiált metaadat kiolvasására képes. Az implementáció működését laboratóriumi mérésekkel is igazoltam, FPGA-alapú, nagysebességű (10 Gbit/s és 100 Gbit/s) hálózati interfészekkel rendelkező eszközök felhasználásával.

A második és a harmadik téziscsoport a VoLTE szolgáltatás magasabb szintű elemzésével foglalkozik. A második téziscsoport összesen három tézist tartalmaz, amelyek a vezérlősík elemzésével kapcsolatos új eredményeket ismertetik. Kutatásaim során az IMS alrendszer interfészeit vizsgáltam, és új módszert adtam a SIP protokoll alapú hívásrekord összeállításra. A hívásrekordok olyan információ kivonatok, amelyek kulcsparaméterekkel jellemeznek egy adott üzenetszekvenciát hibaesemények felderítéséhez. Az IMS alrendszer bevezetésével új vezérlőprotokollként jelent meg a dinamikus méretű és tartalmú SIP. Ezzel együtt az elemzés is bonyolultabbá vált, hiszen a berendezések gyakran elrejtik vagy lecserélik a hívásfelépítés során használt azonosítókat a különböző interfészeken. A tézisekben ismertetek egy olyan új összeállítási módszert, amelyben meghatároztam a rekordok összeállításához és címkézéséhez szükséges kulcsparamétereket, valamint a rekord nyitási és zárási feltételeket. Az előfizetői aktivitás különböző IMS interfészeken történő követéséhez új kulcsparamétereket adtam, valamint foglalkoztam különböző IMS specifikus üzenetszekvenciákkal is.

A harmadik téziscsoport három tézist fog össze, amelyek az adatsíkban megjelenő IP-alapú hanghívások minőségelemzésével foglalkoznak. A becslő modelleket vizsgálva két kategóriát különböztethetünk meg: referencia alapú és referencia nélküli modellek. Az adatsík átviteli garanciáinak ellenőrzéséhez a referencia hangmintát is felhasználó modellek nem alkalmazhatóak egy folyamatosan működő rendszer esetén. A hullámforma elemzéséhez a hangminta nem állhat rendelkezésre, mert az sértené az előfizetői jogokat. Egy referencia nélküli megoldás jó választás lehet, azonban a jelenlegi módszerek által használt metrikák nem fedik le az üzenetvesztés témakörét beérkezett üzenetek esetén az alkalmazás szemszögéből. Munkám során ezért vizsgáltam a beérkezett üzenetek érvényességét is. Ismertetek egy olyan új metrikaszámoló módszert, amely három különböző üzenetvesztés kategóriába osztályozza az üzeneteket, élesen elkülönítve az üzenetvesztés és üzenetátrendeződés fogalomkörét. Az osztályozáshoz egy olyan új, időablak alapú módszert mutatok be, amely során minden beérkező üzenet esetén egy referenciapontok által meghatározott időablak alapján dől el az üzenet kategóriája. A kutatómunkám során felismertem az úgynevezett lassú és dinamikus hangminták megkülönböztetésének szükségességét, amelyhez meghatároztam a peremfeltételeket is. A téziscsoport második részében bemutatom, hogy az időablak módszer alapján számolt üzenetvesztés metrikákból előállítható egy olyan új érték, amely korrelál a referencia alapú minőségbecslő módszerek eredményével.

A többszörösen beágyazott protokollok előfeldolgozása, a vezérlőforgalom elemzése, a hibaesemények azonosítása és előfizetőhöz rendelése új módszerek igényét vetik fel az operátorok számára. Az üzemeltetett rendszerről alkotott teljes kép eléréséhez szükséges a folyamatos elemzés, amely segítséget nyújt a hibaokok felderítésében, vagy akár egy új hálózatrész telepítése esetén is megfelelő támogatást ad. A disszertációmban új módszereket adtam a protokollok előfeldolgozására, a vezérlősík, valamint az adatsík elemzésére is. Kiindulva az alacsonyszintű üzenetdekódolástól, foglalkoztam a magasabb szintű hívásrekord összeállítással, valamint minőségbecsléssel is. Az új eredményeket felhasználva az operátorok egy globális képet kaphatnak az üzemeltetett VoLTE szolgáltatás állapotáról.

We can't solve problems by using the same kind of thinking we used when we created them.

Albert Einstein

Contents

Chapter 1

Introduction and general overview

The evolution of the telecommunication core network architectures significantly accelerated in recent years (from 2G to 4G, or even 5G). The primary factor that drives the evolution of these critical infrastructures is the emergence of new services requiring increased bandwidth. However, these technologies mean independent architectures, the network operators often manage several generations together, in parallel. These independent core networks must communicate with each other, because the mobile phones support different technologies and the call setup can travel from any network to any network. There are also special call scenarios, in which the users change technology during a call, or fall back to a previous technology because of area limitations or error events. Each independent generation has its own protocol converter and gateway function to support these transit events.

Every new technology adds another complexity factor to the management of the whole telecommunication core network because of the previously mentioned compatibility requirements. The VoLTE (Voice over LTE) technology is a complex solution in itself, because the call services use two different, and logically independent architectures: LTE (or 4G) and IMS (IP Multimedia Subsystem). Against the previous generations, VoLTE applies IP-based communication to grant real-time voice services. This is the first architecture, which uses IP packets in the control plane and also in the data plane. The benefits and the drawbacks of an IP-based network appear in the telecommunication, and there are new scenarios and failure events which are unknown in previous architectures. The operators have two main tasks: understand and manage the complex, heterogeneous core network and identify the error events in complex call sessions. There is a need for a continuous service quality analysis to verify the operation of the core network devices and to measure the transmission guarantees in the data plane, which transports the voice packets. Since the IP-based control and voice packets travel between the mobile phones in a VoLTE - VoLTE call scenario, the whole telecommunication network influences the listening quality and the user impression about the VoLTE service.

The increasing packet-arrival rate necessitates efficient solutions for on-the-fly packet parsing, which is an essential and very important pre-processing phase of a higher layer management system. The deep analysis capabilities of software-based approaches can be enhanced by hardware-based support on time-critical packet parsing and classification. Moreover, some payload inspection tasks can be carried out in hardware as well, further reducing the resources spent on software-based solutions.

To inspect the operation of a telecommunication core network domain (e.g., IP Multimedia Subsystem - IMS domain), the operators need new methods and management solutions to detect and understand the network failures. The control part of a call setup procedure is as important, as the user plane communication. The management systems of the new architectures should recognize complex call scenarios, temporary identifiers, failure events, and also new packet-level metrics from the application's viewpoint.

1.1 Motivation and research objectives

Summarizing my previous work experience, I have been working for more than 8 years in telecommunication field. During this period, I collected several feedback from the network operators, and I met the challenges in everyday work. I combined my experiences with my research interests, which include hardware-based network packet processing and VoLTE signaling analysis. During my research work, I focused on new methods that help on the network operators' everyday tasks. Starting from the low-level (hardware-accelerated) packet processing, I also focused on the higher-level VoLTE control plane and data plane analysis, including Call Data Record generation and real-time voice quality estimation topics.

Since the generations of the telecommunication architectures operate in parallel, there are many interfaces with many protocols within the core network. The new generations (e.g., 4G or VoLTE) apply new, IP-based protocols in the control plane and also in the data plane. To introduce and operate the VoLTE technology next to the 2G and 3G network nodes adds another complexity factor for the everyday work, and results several unexpected cases and complex message sequences. The VoLTE service is a complex solution in itself: the architecture uses the LTE core network elements and also the IMS (IP Multimedia Subsystem) network domain. It could be logically further splitted into signaling plane and user plane. The IMS domain grants a common architecture for the call control mechanism, which is available through user plane tunnels (created by the LTE network elements) for the VoLTE-capable mobile phones.

Since the architecture is distributed and new IP-based protocols (e.g., Session Initiation Protocol) appeared in the telecommunication field, the operators need new methods to get a compact information from the state of the network. This feedback could be helpful during a continuous root-cause analysis, or during a new network node installation procedure. Before the LTE and VoLTE services, the IP-based model was unknown in the telecommunication, since it was a closed, reliable architecture. After the introduction of the IP-based operation, the number of failure events also increased. The new control protocols often hide the user identifiers, and also change the session identifiers through a call setup procedure.

Analyzing the control message sequences is necessary but not sufficient task for a global view. Since the data plane transports the voice frames in IP packets, the reliability of the voice channels is reduced. There could be use cases, in which the control part shows successful operation, but the voice frames suffer from network impairment (e.g., packet loss). To get a real-time, complete overview about the operated VoLTE system, the user plane has to be also analyzed next to the control traffic (see Figure 1.1).

The gateways, the protocol converters between the independent telecommunication technologies and the tunneling methods together result several, protocol-specific embeddings. Parsing multiple embedded packets in the user plane at relatively high throughput (e.g., 100 Gbit/s) claims hardware-based methods. Most solutions do not handle multiple, core network-specific protocol embeddings, and recognize only the basic 5-tuple parameters (IP addresses, ports, protocol). This is further complicated by the adaptivity and reconfiguration property of the parsing model, since the protocol embedding in the telecommunication field changes relatively often. Parsing a packet at 100 Gbit/s throughput needs new processing architectures, since the packet speed is 1 packet per clock cycle in worst case. To serve the next, higher-level processing phases with valid, lossless information, the operators need new

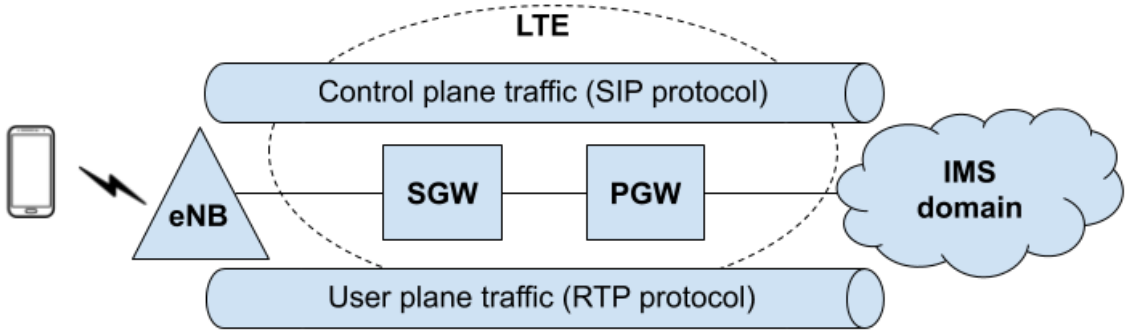

Figure 1.1: IMS domain-related control plane and user plane traffic

hardware-based solutions.

Applying performance measurement standards to get a feedback from the network properties is an essential procedure, which produces simple, packet-level metrics (e.g., packet delay, packet loss). However, a higher-level QoS estimation from these simple metrics is a challenging procedure. If we want to classify the voice quality estimation models, we could reduce the available standards and solutions into two main categories:

- Full-Reference models, and
- No-Reference models.

A Full-Reference (FR) model produces precise feedback about the listening quality, since it compares the degraded waveform with the original one. It calculates many waveform-specific parameters from the original and also from the degraded voice sample to measure the changes and estimate a quality index. Since the algorithms apply several parameters from the original voice sample, the FR model is not a real-life scenario for a continuous VoLTE analysis. It could be used for ad-hoc tests with predefined voice materials, but examining the voice payload in a real-time traffic raises user-privacy issues.

A No-Reference (NR) model could be a concrete choice for quality estimation, since it applies only packet-level metrics and predefined, codec-related parameters. However, the current packet-level metrics do not separate clearly the reorder and loss events in real-time cases. Since the real-time applications use small buffers, there could be several cases, which result a loss event at application-level, despite the packet is arrived in the buffer. This property of the packet-level metrics means that they could not be used for a precise real-time voice quality estimation.

Taking the deficiency of the models into account (see Figure 1.2), there is a claim for a new, packet-level metric set to separate and clearly recognize the loss events from the real-time application's view. Using this new metric set, a new No-Reference model could produce a more precise quality value for real-time voice traffic in VoLTE services.

After the definition of the industry requirements, I found that the exact answer is missing from the related works. The methods and solutions are missing for the new management services, which were real, scientific and also industrial challenges for me.

During my research work, I focused on the previously mentioned problem spaces and claims. I worked on new methods and models for the network operators, to get an appropriate

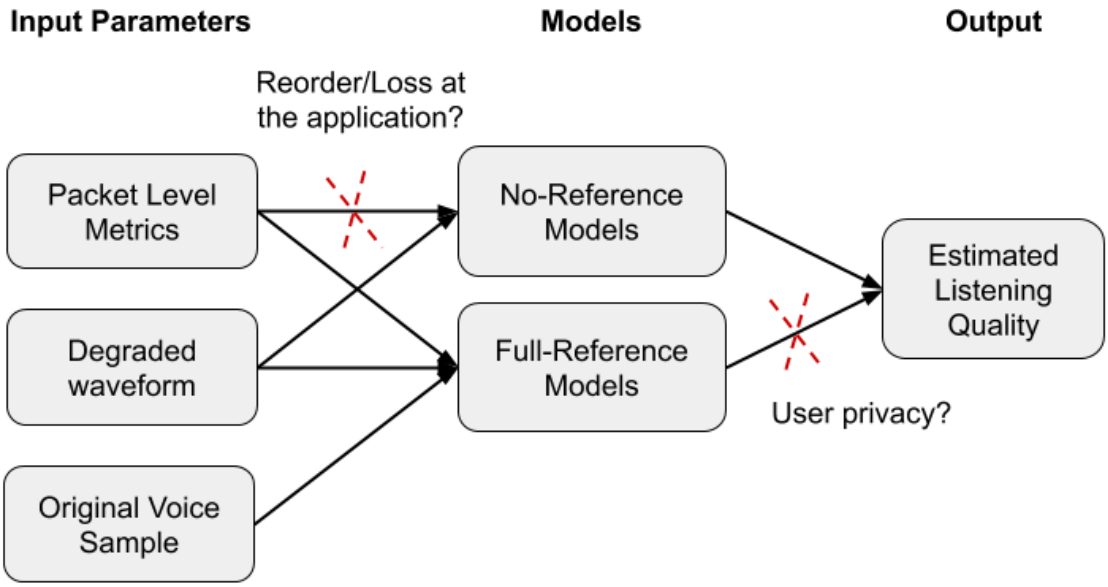

Figure 1.2: Voice quality estimation related motivational challenges in VoLTE services

feedback about the state of the currently operational VoLTE domain, and notice the quality degradation in IP-based voice calls. My dissertation covers the control plane and also the user plane, to provide a global feedback about the real-time state of the IMS domain.

1.2 Overview of the new results

The dissertation summarizes the new results about my previous research work, in which I focused on new methods that help on the network operators' everyday tasks. My research interest focuses on hardware- and software-based VoLTE service analysis, providing new methods, metrics and models.

The first thesis group aims at presenting a new method to support the designing phase of an FPGA-based, real-time, lossless packet parsing engine, to process complex protocol stacks on even 100 Gbit/s throughput capable core network interfaces. The proposed method is validated on FPGA-based network devices with high-speed interfaces.

The second thesis group defines the novelties in higher-level VoLTE service analysis, regarding call control. Call Data Records (CDRs) are essential information sets for network operators, which provides a continuous feedback from a telecommunication service. During my research work, I focused on new SIP protocol specific CDR generation methods, and I provided a complete solution for IMS domain-related challenges. I defined message sequence identifiers to trace the call setup procedure through the whole network. I also defined record closing events and exceptions, to handle the specialties of the IMS architecture.

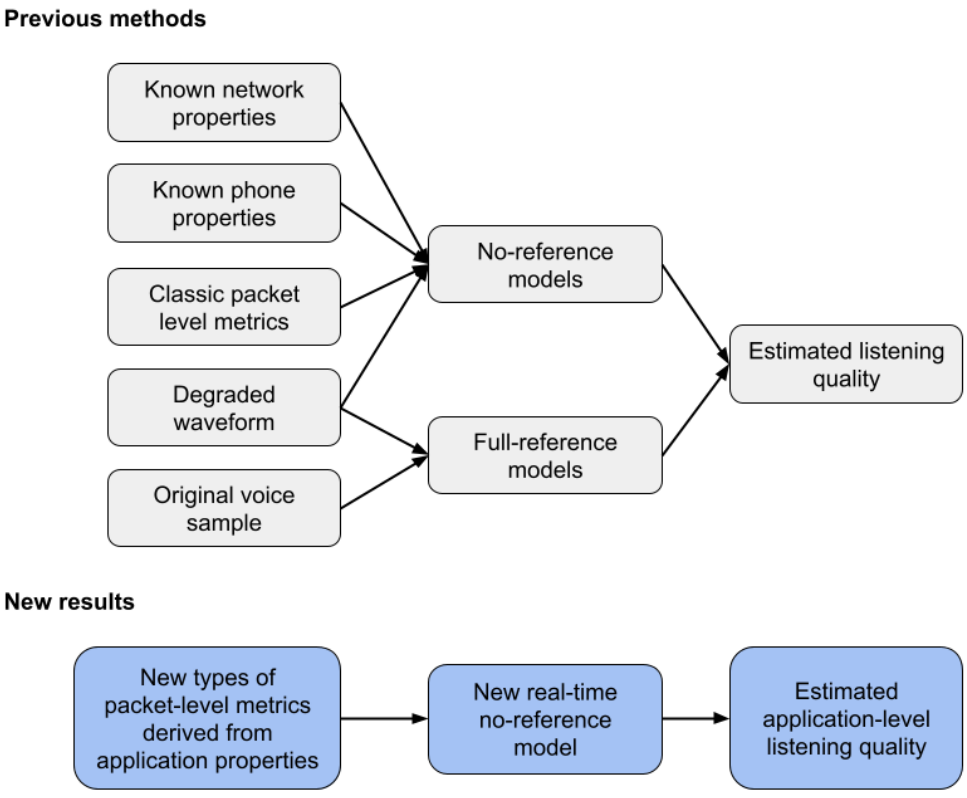

Figure 1.3: Novelties in contrast to the previous models

The third thesis group proposes the new results about data plane analysis. During the investigation of the user plane (voice) traffic, I found that the pure packet-level metrics cannot be used for a precise call quality estimation model. In contrast to previous works, I present new loss categories (early arrived loss, late arrived loss and not-arrived loss) and a

new metric calculation method (see Figure 1.3) to support a new no-reference model from the application's viewpoint. This model operates on the new loss categories and produces a MOS (Mean Opinion Score) [1] index value. The parameters for the new model are derived from the operation mode of the application (e.g., buffer usage, buffer length), and the new metric set treats the reorder-loss concept. Instead of analyzing media data, I modeled the endpoint's operation using a time windowing method. Since I present a no-reference model, it does not violate subscription-related rights.

1.3 Structure of the dissertation

The dissertation is organized into three main chapters. Chapter 2 is about hardware-accelerated packet parsing. I present briefly the FPGA technology and the requirements of the hardware-based design. I also present the MAC (Media Access Control) module operation modes and the parse graph model. The first thesis group presents the challenges and the new results in the above-mentioned research area, in which I defined the viable options for the challenges in a worst-case setup. The first thesis group implies two theses.

In Chapter 3 and Chapter 4 I investigate the higher-level analysis of telecommunication protocols and telecommunication services. Chapter 3 presents the challenges of the signaling plane and implies three theses about control traffic analysis. Chapter 4 is about user plane-related service quality estimation, which includes three theses.

Finally, I present the applied results by the industry in Chapter 5. It is followed by the summary, the cited references and the list of my own publications.

Chapter 2

Hardware-based parsing of multi-encapsulated protocol headers

2.1 Introduction to network packet processing

Carrying out network monitoring tasks remains a continuous challenge, partially because the line rate reaches and exceeds 100 Gbit/s. Besides the increasing data rate, the advent of programmable networks necessitates efficient solutions for supporting packet processing tasks in an adaptive way. Introducing a modification of a protocol or any new protocol in such a flexible infrastructure implies a novel management approach incorporating network monitoring equipment with reconfigurable architecture. The requirement for high throughput and high-level of reconfiguration together put Field Programmable Gate Array (FPGA) technology into the focus of high-performance networking.

Since on-line packet parsing requires a large amount of processing power in order to analyze high volume network traffic in real-time, network bandwidth beyond 10 Gbit/s brings novel challenges for software-based solutions [2]. In a 100 Gbit/s core network, for example, packets should be processed on-line at a rate of 1.5 x 10^8 packets per second, in the worst case. The requirement for hardware assistance of any packet processing phase is therefore emerging with the evolution of the core networks. Accordingly, some reasonable questions arise: i) How hardware acceleration can assist the different phases of the processing? ii) What are the benefits and drawbacks implementing some parts of the packet processing in hardware? Getting advanced Field Programmable Gate Arrays (FPGAs) [3] Network Processors (NPs) [4], or Graphics Processing Units (GPUs) [5] involved in the network functions opens the way for resource-intensive tasks to be accelerated and therefore enables software-based packet processing to step up in the performance scale. Although NPs [6] and GPUs [7] are capable to speed up some well-defined tasks, I focused in my research on the acceleration features of the FPGA technology. Due to its hardware-level reconfiguration property, an FPGA can adapt to the specific requirements of the traffic inspection process. When real-time operation is a requirement (e.g., in an intrusion detection and prevention system or in a real-time traffic analysis system), a further advantage of the FPGA technology is the ability to build an implementation that processes incoming traffic with non-varying latency.

Within a network packet processing system, the analysis process can be divided up to three phases, which are evolved into substantive research areas in the recent years:

- packet parsing,
- packet classification, and
- payload inspection.

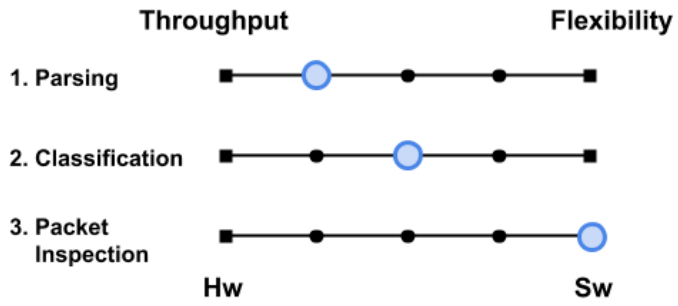

Figure 2.1: Design trade-off at the phases of network packet processing

First, packet parsing is required to identify the different fields of the protocol messages, as well as to reveal protocol multi-encapsulation (e.g., IP-in-IP, Ethernet over MPLS, telecommunication-specific protocol embeddings). The aim of packet parsing is to recognize the various protocol headers encapsulated in the packet header, and extract meaningful information from them. Second, packet classification covers the analysis of well-defined protocol fields, according to a set of filtering rules. Third, the final processing challenge is payload inspection. Based on the result of the online inspection, each individual TCP/UDP flow can be assigned to an application class or to a dedicated application. The primary question related to all packet processing phases is the design trade-off between throughput and flexibility (see Figure 2.1), which determines the requirement for hardware acceleration to achieve high throughput.

During my research work, I investigated the FPGA acceleration of the packet parsing phase considering 100 Gbit/s networking, as a worst-case scenario. I worked on new methods to help the designing phase of a lossless, real-time hardware architecture for complex protocol stack parsing.

2.2 Theoretical background of packet parsing

2.2.1 Protocol parsing

Analyzing an application-layer protocol in LTE and VoLTE services requires essential first steps, which are important but not always mentioned phases: lower layer protocol parsing and information preparation from the extracted header fields. The starting problem set in a telecommunication core network environment consists the following challenges:

- Dynamic protocol headers,
- Dynamic protocol payloads,
- Multi-encapsulated protocols, and
- Frequently changing protocol embedding.

The applied protocols and the protocol embedding is an ever-changing feature of a telecommunication core network. The evolution of the network architectures (2G, 3G, 4G, 5G) requires a complex communication process, because there is an important requirement about the new systems: they should communicate with the previous generations, because of stability and service continuity tasks. The frequently changing network standards bring new protocols or new parameters into the communication sequence, which results in frequently changing embeddings and contents. To visualize and traverse a multi-encapsulated protocol scheme, the parse graph [8] (see Figure 2.2) could be an appropriate description form.

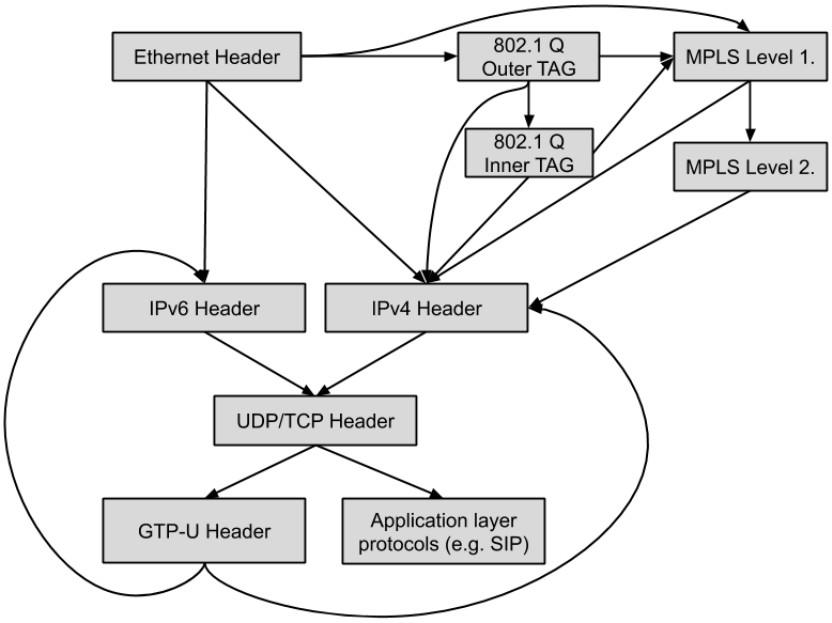

Figure 2.2: An example of a possible parse graph

The parse graph is a directed graph, where the starting node is the protocol header of the lowest layer within the examined system, and the other nodes are the possible next protocol

headers in the encapsulation. If we examine the VoLTE service and the IMS (IP Multimedia Subsystem) domain, the communication is based on IP-packets, and the application-layer protocols use TCP, SCTP or UDP in the transport layer. If we examine the user plane, the mobile-originated IPv6 packets are encapsulated into GTP-U protocol, and forwarded to the service-defined gateway. The node transition in the parse graph exactly represents the possible protocol encapsulation cases. It defines, which protocols we would like to handle from the possible header combinations, and models the complexity of the packet parsing system, and also the complexity of the whole packet processing system.

Two parse graph-related main phases could be differentiated:

- Creation of the parse graph based on the requirement specification,
- Maintenance, which is based on the changes of the telecommunication protocols (add new nodes, remove nodes, add new node transitions, remove node transitions).

However, the parse graph helps to traverse a message at a predefined depth, there is another task in packet parsing: extract and collect header field information for the next processing steps. The basic and well-known header information set is the 5-tuple, which is collected from:

- IP source address,
- IP destination address,
- Protocol field from the IP header,
- Source port, and
- Destination port.

Since the 5-tuple is not always enough information for a special task, there are also custom subsets (n-tuples). These predefined information sets appeared because of the SDN-based (Software Defined Networks) systems, in which the controller unit requires more feedback about the network situations.

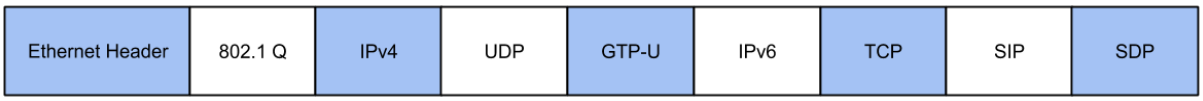

Figure 2.3: An example of a possible telecommunication protocol stack

For a complex VoLTE analysis system the basic 5-tuple is also insufficient amount of information. The telecommunication-specific protocols and headers include service-specific information, which is needed for a higher-level analysis or root-cause analysis. The definition of the extracted information is an important task, and is based on the scope of the packet processing system. The specification should also define the layer of the information. The application layer protocols (e.g., SIP) often use TCP to grant message delivery, which is

encapsulated in the GTP-U protocol between the base station and the gateway to the public data network (Packet Data Network Gateway - PGW). Since GTP-U uses UDP, the TCP-based communication could be often encapsulated into an outer UDP layer (see Figure 2.3).

The packet parsing of a complex and frequently-changing protocol structure raises an important question: How can we model and process a multi-encapsulated protocol scheme in hardware? Since the header embeddings could be properly described with parse graph, it also defines the internal control structure of a hardware-based (e.g., FPGA-based) parsing engine.

Today's state-of-the-art parsing engines [9] [10] handle a large scale of encapsulation structures (i.e., implement a complex parse graph) and extract the appropriate protocol fields (n-tuples) from multi-encapsulated packets that are commonly traversing core carrier networks. The packet parsing process requires up-to-date knowledge of the protocols appearing in the network messages. This does not only mean standardized protocol parameters, but current proprietary implementations, and protocol dynamics, as well.

Adaptation of the parser engine to new protocols is a critical requirement in today's networks. Therefore, updating or reconfiguring the parse graph should be a regular task for network operators when a new service appears in the network.

Based on the method of reconfiguration, parsing engines commonly implement one of the following design principles:

1. Parse graph and extracted n-tuple are predefined in compilation time,
2. Parse graph and extracted n-tuple are reconfigurable in run time with service interruption,
3. Parse graph and extracted n-tuple are reconfigurable in run time without service interruption.

A predefined parser (1.) may support alternative header structures while it is still a closed-design, in which header field extraction always results in a pre-defined n-tuple. The fields of the n-tuple are fixed at compilation time. In contrast, a reconfigurable parser (2.), (3.) has a flexible architecture that enables add, delete and modify operations on the parse graph and therefore it is expandable with new fields and protocols. Nevertheless, the extracted fields are fixed. To overcome this limitation, a parser programming language often belongs to a reconfigurable design, which is object-oriented in many cases to enable a convenient way of creating the header structure definition graph [9] [10] [11]. The third parser type (3.) is an extended version of the second one (2.), where, besides the parse graph, the subset of the extracted metadata is also reconfigurable. New header fields – which were previously not supported by the engine – can be added. For extracting protocol metadata, the packet's header sequence needs to be decoded. To identify the subsequent header, the parser typically interprets the next header field of the current header. However, there are slim headers, which contain no information about the ensuing protocol (e.g., MPLS header contains only a Bottom of Stack bit for this purpose, while the encapsulated protocol can be arbitrary, such as, another MPLS, IPv4/IPv6, or even Ethernet over MPLS). In these cases, recognition of the subsequent header eventuates in more complex steps within the parse graph. To find the appropriate header, executing a simplified pattern matching algorithm on the next header is a common practice.

2.2.2 Hardware-accelerated solutions under 100 Gbit/s

Kobiersky et al. [9] proposed a packet header analysis and field extraction architecture, which is able to process network traffic at 20 Gbit/s on a Xilinx Virtex-5 FPGA. Packet processing implementation is generated from a standard XML protocol scheme. The parse graph is expandable with new network protocols. Operational frequency is in the range of 115 MHz and 165 MHz, depending on the input data width.

Kozanitis et al. [12] proposed a design called Kangaroo system, which is a flexible packet parsing solution supporting 40 Gbit/s throughput. The implementation uses a look-ahead technique to parse several protocol headers in one step. Offsets are calculated using Content Addressable Memory (CAM). The design was prototyped on the NetFPGA-1G board [13], which contains a Virtex-2 Pro chip from Xilinx Inc. The achievable operational frequency, after the place and route phase, was 70 MHz. However, it can be implemented in a standard 400 MHz ASIC, in which the architecture supports 40 Gbit/s throughput. This solution extracts two 32-bit fields for six protocols.

Gibb et al. [14] proposed an abstract model for packet parsers, which can be operated in a 10 Gbit/s switch as a multi-instance design. The model enables implementation on reconfigurable computing devices, i.e., FPGAs. The abstract design contains three main stages: header identification, field extraction and field buffer. The control structure of the first two submodules is based on a state machine, in which steps are reconfigurable through Ternary Content Addressable Memory (TCAM). Authors generated 500 different parser engines from the elaborated model with different input parameters. Then, results have been compared to each other. The study revealed that the occupied chip area depends on the field buffer and the memory size.

Duan et al. [15] designed a NetFPGA-10G prototype for presenting a self-adaptive programming mechanism, namely SAP. The system offers reconfigurable parsing, packet processing and adaptive control features for the network data plane. The parser module uses RAM to get the offset information for the various protocols, and uses a TCAM-based solution for the field matching mechanism. To perform a programmable processing engine, a look up module binds the match results with the actions. While the presented works are optimized for 10 Gbit/s or 40 Gbit/s line rates, state machines and packet buffering are not viable options for higher throughput systems. In the next subsection, we present related works of higher performance packet parsers.

2.2.3 Hardware-based solutions for 100 Gbit/s and above

Attig and Brebner [10] introduced a high-level packet parsing description language. Codes written in this language can be compiled to a high-performance FPGA device. The generated architecture allows run time programmability during the field extraction operation, as well. The solution supports 400 Gbit/s line rate parsing on a single Virtex-7 FPGA, with a data path wider than 512-bit. The language supports an optional header structure, which enables decoding new network protocols.

Pus et al. [16] found latency to be a critical performance factor in high throughput systems. However, due to the high frequency constraint, a heavily pipelined design cannot be optimized for latency or chip area. Accordingly, in this work authors describe a hand-optimized parser, which is able to process data at rates higher than 100 Gbit/s. The architecture relies on uniform pipeline stages, which are optimized for dedicated tasks. A classic 5-tuple prototype was implemented on a Xilinx Virtex-7 870HT chip.

Brebner and Jiang [11] proposed a high-level object-oriented language, namely PX, which is designed for packet processing engine specification. The language covers the hardware designer's tasks and lets the user deal with the main targets. The compiler generates a pipeline architecture with live re-programming property. Their paper gives an overview of the main generated functions: a parser and a classifier engine. These functions are demonstrated through a 100 Gbit/s OpenFlow implementation, in which the parsing engine extracts 12 OpenFlow fields from four headers. The architecture uses 512-bit data path for 100 Gbit/s throughput.

Benácek et al. [8] [17] presented a high-level parse graph description language, namely P4. The compact solution contains a code generator part, which converts the P4 graph into synthesizable VHDL code. Test results show that the generated VHDL implementation is able to process traffic at 100 Gbit/s speed on a Virtex-7 FPGA device. P4 defines five basic language tools to determine header format, state machine, extracted headers, actions, and control flow. The generated implementation applies a pipeline architecture with 512-bit wide data bus.

Hu et al. [18] proposed an SDN-based architecture. The solution uses the P4 abstract language for the forwarding plane to resolve the scalability problem space.

Li et al. [19] introduced a programmable parser, namely P5, which is able to operate at 100 Gbit/s speed on 128 bytes packets. The solution uses a circular pipeline for parsing (lower than 200 ns latency), where each header is parsed in a loop. An action RAM is used to obtain the parsing information, which fields are needed to be stored. The prototype is demonstrated to operate at 200 MHz on an Altera FPGA-based platform, namely NetMagic (which only has 8 x 1 GE interfaces).

Pus et al. [20] proposed a pipelined packet parser architecture for over 100 Gbit/s operation throughput. The authors offer 2048-bit wide data path for over 100 Gbit/s. Each pipeline stage uses extra input information about the protocol stack (e.g., header type, header offset, partially aligned start, last byte of the packet). The hand-optimized prototype is able to process more minimum-sized packets in a 2048-bit datapath. Test results show that 400 Gbit/s throughput is achievable in a Virtex-7 FPGA.

Da Silva et al. [21] used P4 for describing the parser graph, and achieved a low-latency implementation when compared to [14] and [8]. The authors used high-level synthesis, their

RTL code is generated from the C++ description using Xilinx Vivado HLS and synthesized with Xilinx Vivado. They found that to achieve 100 Gbit/s throughput, their optimum design trade-off is at 320 bits data bus length with 312.5 MHz clock frequency with a Virtex-7 FPGA

Orosz et al. [22] created a parse graph of 14 elements, covering the typical combinations of L2-L4 protocol encapsulations. This solution supports GTP (GPRS tunneling protocol), a typical element of mobile core networks, as well. The base platform – called C-GEP – is an actual Virtex-6 FPGA-based, 100 Gbit/s-capable network node. The design trade-off found by the authors in this architecture is 312.5 MHz operating frequency and 512 bits long data bus. Among its various applications, C-GEP is used as a high-speed network monitoring equipment [23].

2.3 Technological background of hardware-based packet processing

2.3.1 The FPGA technology

The FPGA (Field Programmable Gate Array) concept means a reconfigurable IC (Integrated Circuit), which contains more thousands of general elements (e.g., shift registers, flip-flops, adder-multipliers) in a single chip [24]. The general logic elements are often ordered into logic blocks, which blocks are organized in a well-defined matrix. The connections between the general elements are reconfigurable through high-level hardware description languages (e.g., VHDL and Verilog). The operation of an algorithm is based on clock cycles, as in a general CPU. However, the latency is fixed and the operations are well-defined for each clock cycle. One of the challenges for a hardware designer is to meet the frequency constraints through the configured interconnections. The number of logical elements and the maximum achievable latency for an interconnection are based on the FPGA family and on the type of the chip.

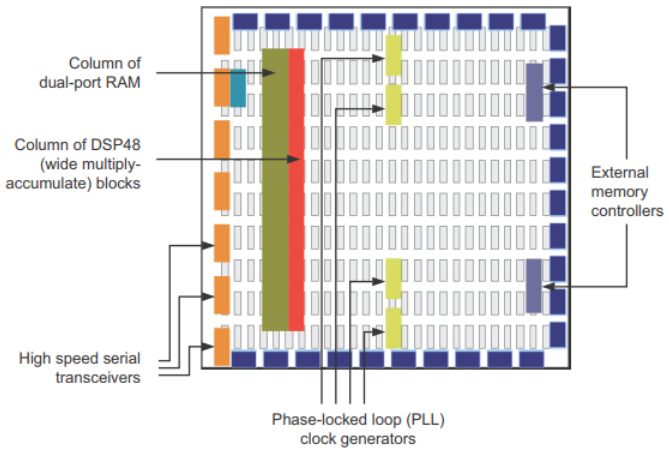

Figure 2.4: A general FPGA architecture [24]

The main advantage of the architecture against a general processor is the parallelism. Since bitwise operations run in parallel, an FPGA device can very effectively assist a software solution in the critical network packet processing tasks (e.g., reception, parsing, classification). However, it also has limitations, which are traceable to the physical resource restriction. A general design goal of a network packet processing system is the high throughput, which means that the hardware modules within the FPGA device must operate at a relatively high frequency. This minimum frequency is a constraint (from design perspective) for the connection path between the logic elements. During the place and route process, the designer software tries to meet all constraints, but the achievable operations per clock cycle can be often a design limitation factor. In addition, the maximum operational frequency is an FPGA bottleneck, because it lies between 20 MHz and 400 MHz, even in the state-of-the-art category devices. The available frequency for the data path also depends on the logical complexity. In order to get a high operational frequency, as a general rule, simple

operations should be used in each clock cycle. In this case, the data path is shorter and therefore signals can propagate at a higher frequency. Another problem can be the different clock domains, which is a typical network packet processing problem. Notably, the receive modules operating on different clock sources from the transmit modules. The synchronization has to be solved. Finally, a basic hardware bottleneck is the limited physical resources. During the planning phase, the main goals should be summarized, to choose an appropriate FPGA chip family, because in later implementation phases the size of the chip, the number and types of I/O PINs and the number of logic elements is not extendable.

2.3.2 Packet processing in FPGA

FPGA's for packet processing contains pre-defined hardware elements for Ethernet frame transmission and reception. This module is called as Media Access Control module (MAC), which hides the physical layer operation, and also the requirements of the communication (e.g., interframe gap, checksum calculation, etc.). It grants an appropriate input/output black box for a packet processing engine. The operational frequency and the data path width depend on the interface throughput, which defines the requirements of the packet processing phase.

802.3ae operation mode

The operational frequency of the generic 1 Gbit/s Ethernet MAC module is 125 MHz, and its data path is 8-bit wide. The physical size of the MAC module inside the FPGA is relatively small, typically, not more than 5% of a low-end FPGA. The clock frequency is not a challenge even in a low-end FPGA, such as Spartan-3A or Spartan-6 from Xilinx inc., which are good choices for basic packet processing tasks on 1 Gbit/s Ethernet links, i.e., 5-tuple packet parsing or packet timestamping. Because of the 8-bit data path, metadata for packet parsing or packet classification has to be collected byte-by-byte. For example, destination MAC address needs 6 clock cycles to be stored before a parsing step [25]. At 1 Gbit/s bandwidth the processing system has enough clock cycles to perform the main packet processing tasks, according to the worst case, which is 64 clock cycles for a 64-byte Ethernet frame. In a best-practice design, the entire user data path is 64-bit wide within the chip. In that case, the processing modules have 8 clock cycles to parse and classify each incoming packet, which period is long enough to enable using finite state machines.

802.3an operation mode

Stepping up to 10 Gbit/s, the operational frequency of the Ethernet MAC module is increased to 156.25 MHz, and its data path is 64-bit wide. The size of the on-the-market available MAC IP cores varies between 5% and 25% of the logic elements, depending on the type of the FPGA chip. The 64-bit data path enables more protocol fields to arrive in one 64-bit word. Therefore more parsing information can arrive in one clock cycle. Well-prepared finite state machines can handle the processing phases effectively, but each module has to store or pass the incoming data to another module for further processing within 8 clock cycles (see the worst case, which is 8 clock cycles). The higher operational frequency (156.25 MHz) requires

a mid-class hardware device. However, stepping up to this bandwidth does not imply new internal design in terms of data path width and number of clock cycles between consecutive packets. Handling multiple data lanes increases the size of the 10 Gbit/s MAC module to even 25% of the chip, depending on the target FPGA family.

802.3ba operation mode

Operating at 100 Gbit/s implies significantly higher core frequency as well as extremely large, 512-bit wide data path. In this case, every incoming 512-bit word can contain a new, minimum-sized Ethernet frame. This property raises many processing issues. First of all, a processing module cannot store the incoming packets, otherwise, it should store 1.5 x 10^8 packets per second, in worst case. Another challenge is parsing unknown protocol structures, since there is only one clock cycle to decode the protocol fields and classify the packet based on the result. Meanwhile, a new packet can arrive at the next rising edge of the clock signal. In addition, design planning phase is also more complex: the Media Access Control sublayer may be implemented with two significantly different data path handling methods: segmented and non-segmented. Selecting the appropriate mode enables us to optimize the trade-off between frequency and data path width, in which the only constraint is the line-rate processing. The bitrate predefines the values of the mentioned two parameters (see Figure 2.5).

Segmented vs. Non-Segmented mode

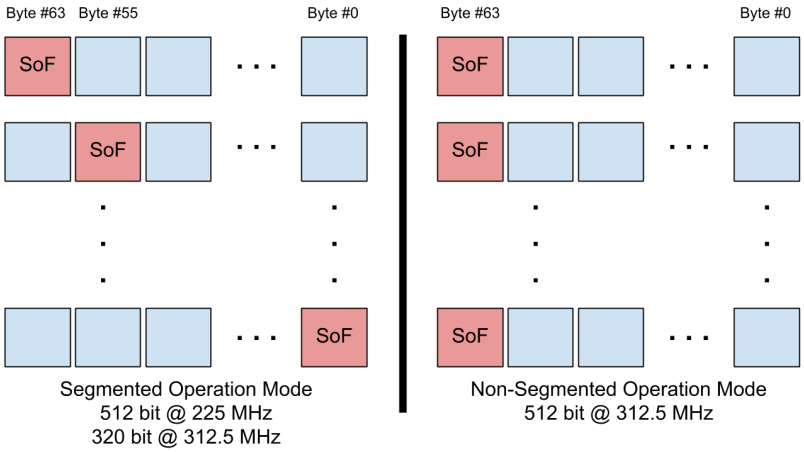

Figure 2.5: Design trade-offs (Segmented / Non-Segmented)

Non-segmented mode is the easier to work with, because the first byte of the packet is always placed at the beginning of a 512-bit word. For example, if the packet ends in the current word, the new incoming packet should start always in the successive word, and the first byte of the packet is always in the byte position 63. Due to the Start of Frame (SOF) synchronization process, the frequency is higher than in segmented mode for the same data path width. The general operational frequency of the MAC in non-segmented mode is 312.5 MHz. Operating a complex logic, i.e., 100 Gbit/s MAC at such a frequency is a

real design and implementation challenge in FPGAs, since higher frequency implies smaller latency within the architecture. Considering the internal design of the MAC, segmented operation mode is more straightforward to implement. Nevertheless, due to the additional logics, it results on larger application modules. The main difference is the positioning of the first byte of a new package, which, in this case, can be placed at every 8-byte boundary of the 512-bit word. Since the Start of Frame signal is not synchronized to the next word's first byte, frequency is lower than in non-segmented mode. The common operational frequency can be 225 or 312.5 MHz, depending on the data path width. There are design trade-offs between segmented and non-segmented modes, and there is no generic solution. The architecture depends on the purpose. To maintain the line-rate processing, the segmented MAC enables a lower operational frequency, and thus the designer can use more complex operations, or alternatively, the data width can be narrower, while keeping up the same frequency. In the latter case, there is a higher probability for a protocol field to span over two or more consecutive 320-bit words. For example, a 60-byte IPv4 header (which is the maximum length of the header) may span over even 3 x 320-bit words (see Fig. 2.6), while using 512-bit data path, the 60-byte IPv4 header can be delivered in 2 x 512 bit words, as with non-segmented processing. In segmented mode, the number of resources occupied by the MAC module is smaller since there is no synchronization logic. However, offset and index calculations need extra logic to listen to the first byte of the packets. In non-segmented mode, the parsing module does not need this extra resource at the cost of a larger physical size.

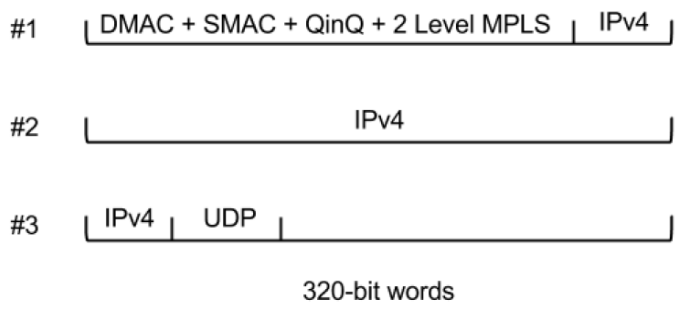

Figure 2.6: IPv4 header overflow

In a parsing engine, the internal modules have to ensure the decoded protocol fields, so-called metadata, to be handled in-sync with the packet they belong to. In the classification phase, the destination of the packet depends on these metadata. In non-segmented mode, due to the static position of the first byte, the path of metadata can be fixed. In segmented mode, the position of the first byte is variable and therefore in-sync handling should be implemented. In a 100 Gbit/s throughput capable subsystem the type of the FPGA family is limited by the type of the I/O transceivers. High throughput transceivers are available only in high-end devices. Moreover, multi-lane architectures (i.e., ten 64-bit lanes) with multiple transceivers occupy a dominant part of the chip. A 100 Gbit/s MAC module can own even the half of the hardware device.

2.4 New results in hardware-based packet parsing

Thesis group 1 - I have investigated the limitations of hardware- and software-based packet parsing methods to process multi-encapsulated protocol headers. I have provided a new synthesized method to support the designing phase of a real-time, lossless packet parsing engine in an FPGA-based architecture. Based on the proposed method, I have designed and implemented an optimal packet parsing engine. It features 100 Gbit/s throughput, 8 different protocol headers and 14-tuple worst-case parameters. I have validated the designed architecture with laboratory measurements against the lossless and real-time criteria. [J2] [C7] [C8] [C9] [C10] [C11]

2.4.1 Challenges about packet parsing in FPGA

As previously mentioned, the aim of a packet parsing system in the telecommunication field is to recognize the protocol embeddings and transmit the extracted n-tuples to the next processing phases. The sequential operation is a common design principle in the 1 Gbit/s and 10 Gbit/s link speed, which is not a viable option in a 100 Gbit/s packet processing engine. The minimum Ethernet frame size and the data path width together give a design limitation for the 100 Gbit/s processing modules, where, in the worst case, a minimum-sized packet can arrive in every clock cycle. Because of these properties, the sequential operation with temporary data storing cannot guarantee lossless operation. The part of a computation should pass to the next processing step, where each processing stage (i.e., pipeline stage) operates at high frequency with simple operations.

Pure software- or pure hardware-based solutions also have limitations, which come from the architectural and from the operational properties. The main disadvantage of software-based processing methods against hardware-based ones is related to execution time. This is partially caused by the pure existence of the initial, minimal, constant delay of fetching the packets from the interface card to be analyzed by the processor. Besides this, there is a variation in processing delay. It depends on how the processing software is scheduled to actually get it through the operating systems' control. Depending on the depth of the analysis, processing time may differ from packet to packet. An extensive generic survey on the existing high-performance software-based packet processing frameworks is presented by Moreno et al. [26]. The software-based parsing and classifier algorithms are working with packets that are previously stored in the main memory by the operating system. Since the kernel-level packet processing involves polling-based reception and multiple packet queues within the processing path, the availability of the received packet for the user space parsing application is neither real-time nor guaranteed. The availability of the line rate operation faces serious bottlenecks. Furthermore, a software solution cannot guarantee lossless packet reception, since it is executed on shared hardware resources and its CPU time is controlled by the operating system's task scheduler. If a CPU core reaches the maximum load, packets may get dropped, depending on the packet queue length.

In contrast, a hardware solution is designed for dedicated purposes and tasks. Depending on the requirements, the hardware-assisted DPI may have drawbacks. The primary bottle-

neck is the maximum available frequency, as mentioned in the last section. The second performance issue is also related to the size of the mapped design: the physical space required within the circuit. A hardware device has a given number of logical elements, which sets a limit to the size, functionality and complexity of the implementation. The output of the Place and Route operation – that results in the physical layout of the physical elements and the allocated routes between them inside the FPGA – depends on the latency. The performance of an implementation can be enhanced by using a higher capacity chip, but the physical resource set always remains a limiting factor. In contrast to the typical software development process, FPGA programmers need a lot of time and effort to reduce the size of their implementations, or to map it to another type of FPGAs.

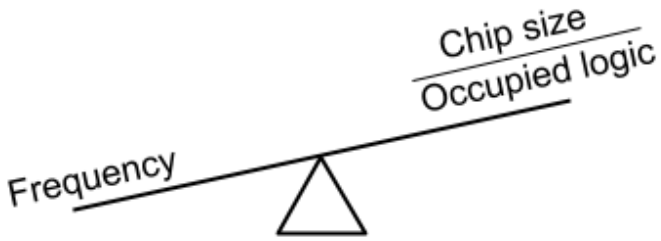

Figure 2.7: Latency and other requirements necessitate FPGA design trade-offs for high-performance packet processing

The number of logical units and the maximum available frequency together dictate the operational constraints for a given algorithm and its implementation. These properties restrict the operational complexity of FPGA-based DPI methods (see Figure 4). All of the memory and control elements for packet header parsing rules, for the packet classification rules, and for the payload inspection have to fit into the FPGA. In order to detect complex header structures, the parsing algorithm must handle a lot of cases and branches. However, the number of protocol combinations is not arbitrary, because of the logical complexity and the physical resource limitations. The multiple instantiations and the parallel processing are also constrained by the size of the chip. For example, if we have a hardware implementation, which is capable of 10 Gbit/s of throughput, a parallel 4-instance configuration can perform at 40 Gbit/s. However, the number of occupied logical units shows non-linear increment, and the longer internal routes reduce the maximum achievable frequency [8] [20]. Following this example, a 100 Gbit/s design is much more complex, and we have to instantiate ten copies of this module. In addition, the algorithm has to distribute the data path, synchronize the instances, and control the modules. In contrast to the 10-instance module, it is a better choice to use a one-module approach, which is able to achieve 100 Gbit/s of throughput with one single instance. Having only a single instance makes controlling simpler, requires less hardware resources, and eliminates the need of synchronization logic between the instances. Considering the specified internal resources (e.g., number of logical and memory blocks) of the state-of-the-art FPGAs (e.g., Xilinx UltraScale), using multiple instances of a 100 Gbit/s-capable module, a throughput of 400 Gbit/s, or even 1 Tbit/s can be reasonable to achieve. Nevertheless, it is not trivial to implement a working 100 Gbit/s design in practice, because of the presented constraints. The packet parsing algorithm should co-operate with an FPGA

implemented Media Access Control (MAC) module that, at this speed, occupies a large portion of the chip, alone. As the number of occupied logical elements increases, finding electrical routes with the required latency constraint becomes more and more challenging.

2.4.2 Thesis 1.1

Thesis 1.1 - I have created a comparative study about the limitations of hardware- and software-based packet parsing solutions to process multi-encapsulated protocol headers at high throughput. Based on this work, I synthesized a new method to design adaptive, real-time, lossless packet parser architectures in FPGA supporting the operational management of IP-based mobile core networks. [J2] [C11]

As previously mentioned, the aim of packet parsing is to identify the various protocol headers encapsulated in the packet header, and extract meaningful information from them. These methods recognize the protocol header structure, multiple encapsulations, and are able to handle the so-called shim header embedding.

In my research work, I focused on the planning phases of a complex parsing solution. I defined the viable options for each phase for a worst-case setup to support the decisions during the design steps.

In comparison to packet processing at 10 Gbit/s, data rates at 100 Gbit/s and beyond imply higher demand for the FPGA internal design and resources. This practically means higher operating frequency and a significantly larger data path width, typically 320-bits or 512-bits. For the target throughput, the effective frequency and data path width combination depend on the design of the Media Access Control (MAC) module. There are two commonly implemented output modes: segmented and non-segmented. In segmented mode, each word can contain portions of multiple data frames, while non-segmented mode restricts frames to start only at the first byte of each word. In the available 100 Gbit/s Ethernet MAC IP cores [27] [28] [29], the frequency of the segmented operation is typically 312.5 MHz, combined with a 320-bit data path. Although this seems to be an implementation-specific constraint, it is wise to respect it during system design. However, there is a lower frequency mode for the target throughput that is 225 MHz with a data path of 512-bit. In non-segmented mode, the commonly applied frequency (in MAC IP cores) is 312.5 MHz and the data path is also 512-bit wide. Considering the operation modes, we can assume a 320-bit or 512-bit wide data path as a general choice of implementation. Based on my study, I found that the 512-bit data path is a feasible route within a medium category FPGA chip (e.g., Virtex6 Family from Xilinx Inc.), but applying a wider data path (e.g., collecting more 512-bit word into one 2048-bit word) conflicts with frequency requirements.

In order to maintain a uniform data path and a single clock domain within the FPGA, the submodules within the parsing engine should adapt to these design constraints. The parser engine must treat incoming packets at line rate. Since a minimum-sized Ethernet frame fits into one 512-bit word, a 100 Gbit/s parser engine must accept a new incoming packet in every clock cycle, in the worst-case scenario. This implies the processing of 1.5×10^8 packets per second. The requirement for such high throughput rules out parser engines based on store-and-forward (e.g., packet buffering and state machines) architecture. To achieve the required core frequency, simple hardware operations must be used, where the logic is able to operate on new incoming data in every clock cycle. I found that this design implies pipeline architecture.

Regarding the previous works [15] [8] [19] [20], I found that the best practice of the

hardware-assisted packet parsing is based on a multi-stage pipeline design [16] [17] [19], which suits well to the physical architecture of the FPGA chip (see Figure 2.8).

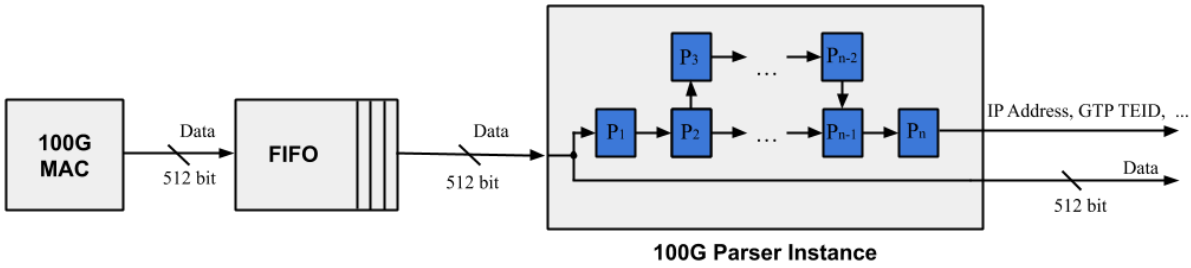

Figure 2.8: Pipeline-based 100 Gbit/s protocol parser architecture in FPGA

My method is aimed to design a pipeline architecture, which includes the following three main design steps:

- Create a protocol graph from the predefined protocol headers and header encapsulations based on the defined graph criteria.
- Design the stage categories and also the structure of the pipeline based on the graph nodes.
- Design the operations of each pipeline stage based on the formalized criteria.

The first step is to model the protocol hierarchy regarding the analyzed network interfaces. To model the protocol encapsulations and processing orders, the parse graph is an effective solution. The designed protocol graph must be a directed, acyclic graph, where the nodes are the protocol headers. When a protocol header appears multiply in the encapsulation (e.g., IPinIP, internal and external IP header in the GTP-U tunneling) then each occurrence must be represented as an independent node.

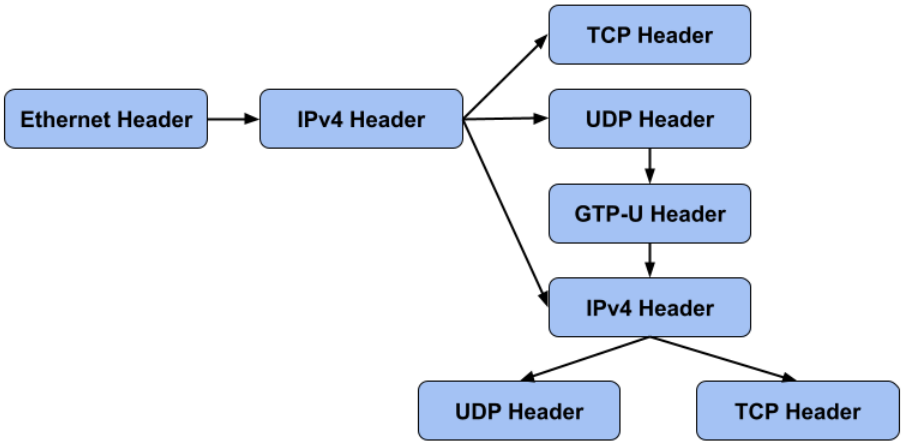

Figure 2.9: A simple parse graph to represent GTP-U encapsulation

The nodes and the node transitions could be fitted into pipeline stages and stage transitions, which is the second step of the method. This architecture also supports the reconfiguration requirements, new nodes (new protocol headers) could be easily inserted into the pipeline. However, analyzing a complex parse graph with high throughput often demands an extremely large pipeline structure, in which latency is directly proportional to the number of operations (i.e., number of used clock cycles).

Let P denote the pipeline and let P_i be the i^{th} stage within P where $i = 0, 1, \ldots n$, and n is the number of pipeline stages. Each P_i has its own well-defined task based on the parse graph nodes. These stages should be not complex (low number of logic levels), and thus are capable of operating at a relatively high frequency. In terms of I/O interfacing, the FPGA device fits very well into the physical route of the packet, continuing the data path as a pipeline inside the chip. There is a trade-off between operational frequency and data path width. The required throughput (packet rate) can be assured with multiple data path width and frequency combinations.

Evolving the architecture of P is depending on the function of the parser. Each P_i can be classified into one of the following three classes, depending on its task. Let $C_{P_i} \in \{C1, C2, C3\}$ denote the complexity of P_i.

- C1: P_i operates on a simple, well-defined task, working on elementary operations. Parsing of a protocol header is performed by multiple consecutive P_i.
- C2: Pi operates on one full header of the protocol structure.
- C3: Pi operates on multiple headers within the protocol structure.

P can be also classified depending on the chain of the architecture:

- linear: P has a series of consecutive P_i stages.
- non-linear: P_i stages can operate in parallel relative to each other.

During the second phase of the method, which deals with fitting the graph node into pipeline stages, each node and stage should be translated into the previously defined categories.

The third step covers the operations in each stage regarding the throughput and the extracted tuples criteria. In this step, the operating frequency should be a priority. This is because to operate the parser at higher throughput with a predefined data path the frequency should be increased. In addition, to process the packet stream on a narrower data path for lower or optimal resource utilization, also the frequency should be analyzed. I determined the following designing criteria about the maximum operating frequency.

Let $O_j(P_i)$ denote the j^{th} operation in P_i, and $j = 0, 1, \ldots m_i$, where m_i is the number of operations in P_i. Let fopmax($O_j(P_i)$) denote the maximum frequency of the $O_j(P_i)$ operation. Then fPimax(min(fopmax($O_j(P_i)$))) is the maximum achievable frequency for P_i.

Let $k_i \in R^+$, where k_i is the weight for the routability of P_i within the chip. Then $f_{P_i} = k_i$ x $fPimax$ is the achievable maximum frequency for P_i provided by the place and

route process. Then $f_e = min(f_{P_i})$ is the achievable maximum operational frequency of the pipeline architecture.

The complexity of the protocol header structure is the main design challenge during the planning process to achieve the highest f_e. As an example, dynamic IPv4 header is advisable to parse with multiple P_i stages. If the header sizes in the protocol structure are not variable, the complexity of the stages does not set a bottleneck for operating f_e frequency. The main design goal for a P pipeline architecture is to define optimal C_{P_i} classes for each stage in order to achieve the highest frequency on a given FPGA device. Handling packet data inside the parser engine, which eliminates the need for packet buffers, is a drawback of the pipelining technique in terms of resource utilization. This internal data path can occupy significant chip area, depending on the implementation mode and the target hardware type. As an example, Virtex-7 FPGA family is currently a popular choice for high-performance parser engine implementations, where basic elements within the Configurable Logic Blocks (CLBs) are D-type flip-flops, multiplexers and shift registers, grouped in slices. When the internal packet data path is built from D-type flip-flops, it needs 64 slices for each 1 clock cycle long parsing step. According to an experimental simulation with a medium category Virtex-7 chip, a parser with near 0.5 us latency therefore requires 79360 D-type flipflops just for the internal traveling of the related packets, which sums up in 9920 slices. This means that a single data path can occupy nearly 15% of a medium category Virtex-7 chip.

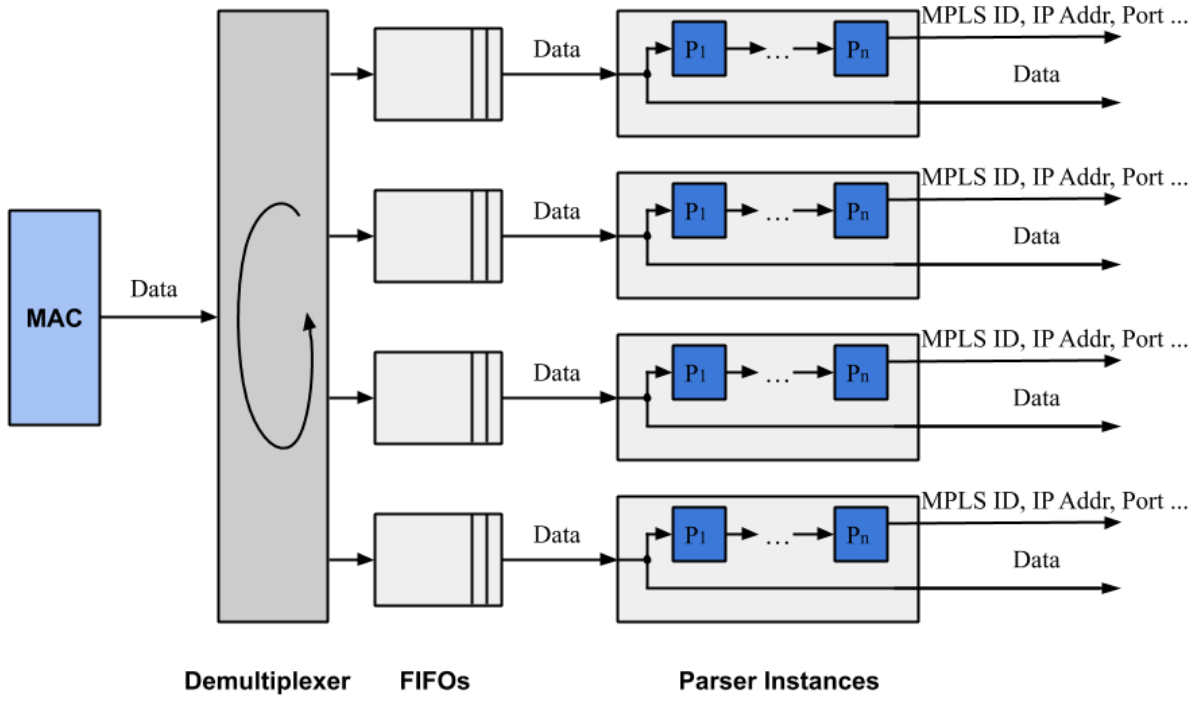

Figure 2.10: Multiple instance parser engine

Another factor for resource allocation is the design principle itself: single-instance (Figure 2.8) versus multi-instance (Figure 2.10) engine. For 100 Gbit/s of performance, a single-instance parser seems to be a reasonable design, since core frequency related to 512-bit data per cycle can meet the throughput requirement. However, in case of a complex

parse graph, this design may not be able to operate at 312.5 MHz and therefore should be substituted with multiple lower-throughput instances. While this engine operates at a lower frequency, the appropriate scheduling algorithm performing synchronization between instances claims extra logic. As the result of the parsing phase, the engine produces a set of protocol metadata directly acquired from the captured packets. In terms of functionality, configurational flexibility of the parse graph is important, in order to support the successive packet processing phase (i.e., the classification engine) with the aptitude to adapt to a wide scale of network configurations. Actually, it is one of the key features of the parser engine to enable the modification of the parse graph to update protocol field layout or inserting new networking protocols.

Summarizing the packet parser planning phases, the following high-level steps could be differentiated:

- Model a parse graph based on the protocol hierarchy of the analyzed architecture. The parse graph should represent all protocol headers, and a node should not contain a transition to itself.
- Define the extracted protocol information (n-tuples) based on the aim of the packet processing system.
- Define the applied MAC module, which also gives the data path and frequency requirement for the parser.
- Define the P_i pipeline stages based on the parse graph, and also the C_{P_i} complexity of the stages.
- Based on the C_{P_i} complexity and the f_e maximum achievable frequency, P_i may have to be further splitted into more stages.
- Route the original data path through the pipeline, which guarantees a general interface (new stages could be easily inserted, each stage gets all data, lossless operation without buffers).

2.4.3 Thesis 1.2

Thesis 1.2 - Based on the method presented in Thesis 1.1, I have designed and implemented a packet parser architecture to process multi-encapsulated LTE user plane traffic at 100 Gbit/s throughput, recognizing 8 different protocol headers and extracting 14-tuple. I have performed the functional verification with simulated operation on prerecorded traffic samples, and also with generated protocol header encapsulations. Using FPGA-based networking devices, I have validated the designed architecture with laboratory measurements against the lossless and real-time criteria. [C7] [C8] [C9] [C10]

As a verification and validation process, I worked on a predefined requirement specification to design a packet parser engine in hardware. I applied Thesis 1.1 to design and implement the processing architecture within the FPGA chip. During my work, I realized the system design in VHDL language on a Virtex-6 FPGA-based network device, namely C-GEP. C-GEP is a single-FPGA packet processing platform from AITIA International Inc., handling 100 Gbit/s Ethernet traffic. C-GEP has its own 100 Gbit/s PCS/PMA (Physical Coding Sublayer/Physical Medium Attachment) and MAC (Media Access Control) protocol stack implementation provided by AITIA. The MAC module is fully compliant to IEEE 802.3ba-2010 [30] and operates in non-segmented mode. The 100 Gbit/s modules are field-tested on C-GEP with CFP SR optical modules. The architecture of the C-GEP device provides a general platform for network measuring components. As the platform is a combination of high-speed interfaces, a reconfigurable hardware chip and an embedded computer, it can be used for various specific tasks.

The requirement specification against the packet parser engine was the following:

- Real-time, lossless operation,
- 100 Gbit/s throughput,
- Non-segmented MAC operation mode, which requires 512-bit wide data path combined with 312.5 MHz operational frequency,
- 14 predefined tuples,
- 8 different protocol headers (Ethernet, VLAN, MPLS, EoMPLS, IPv4, UDP, TCP, GTP-U),
- Xilinx Virtex FPGA family.

The FPGA's internal data path maintains the 512-bit wide path in each of its sub-modules. To achieve the target throughput, the parser engine has to operate at a core frequency of 312.5 MHz.

As the first step, I created a parse graph from the protocol headers and their possible combinations (see Figure 2.11). To support the pipeline stage fitting phase (and also for better visualization), I duplicated some protocol headers in the parse graph (e.g., IPv4,

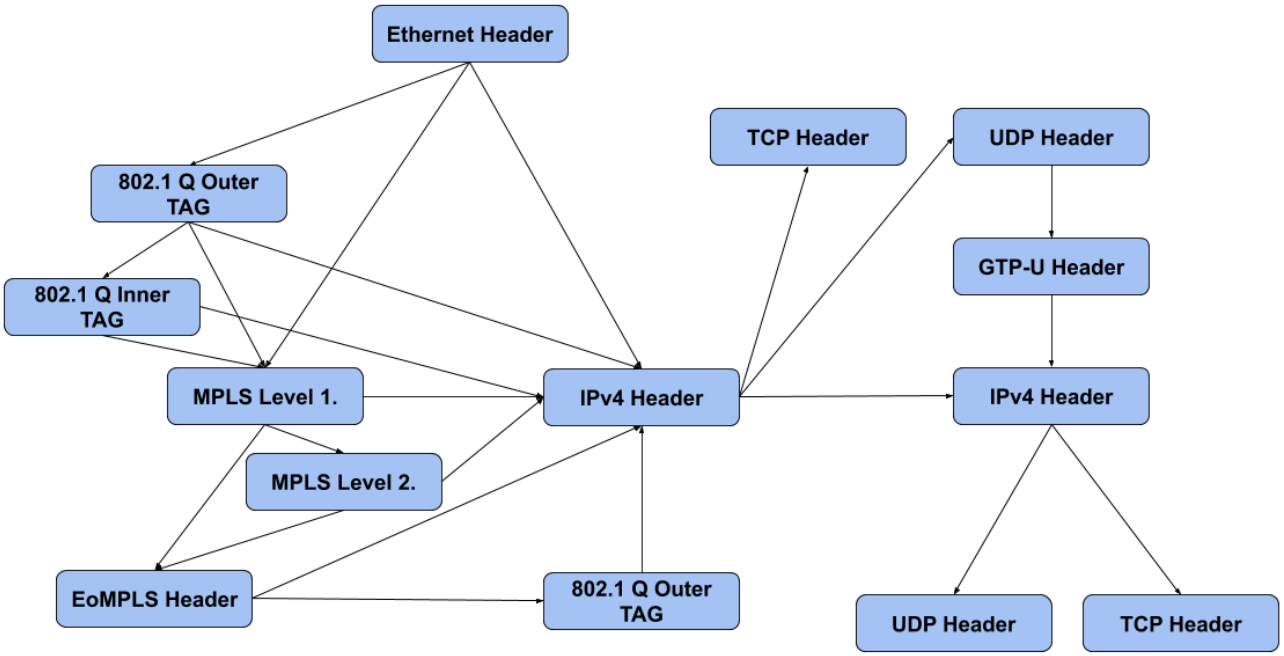

Figure 2.11: The parse graph of the designed system

UDP, TCP). Since the pipeline should be a one-way route for a real-time, lossless operation, the parse graph should not contain backward transitions.

Considering today's networking trends, evolving network services result in the appearance of new protocols or protocol fields at an increasing frequency. To cover the spectrum of network protocols present in core IP networks, my parser engine enables to identify multi-encapsulated packets beyond decoding the classical 5-tuple, i.e., IP address pair, port pair and transport protocol.

Based on the requirement specification, the engine extracts 14 tuples, which are the following:

- Ethernet MAC Addresses,
- Outer VLAN tag,
- Inner VLAN tag (QinQ),
- MPLS tags (two levels),
- EoMPLS MAC Addresses,
- VLAN tag in EoMPLS,
- IPv4 addresses, also within GTP-U encapsulation, or in IPinIP encapsulation,
- UDP or TCP ports, also within GTP-U encapsulation

The dynamic length of the IPv4 header (using optional fields) is also handled during the parsing process. The decoded protocol metadata are synchronized to the original packet, and buffered for further processing. For assessing QoS of a network service, lossless packet processing is an essential requirement. Since the 100 Gbit/s MAC module operates in non-segmented mode, every 512-bit word (and therefore every clock cycle) can contain a new, minimum-sized packet. This property dictates a very strict timing constraint for the parser engine, which has to decode the header structure within one clock cycle. Using buffers or storage memory is not a viable option at this core frequency, because the intra-chip routing to the memory elements is often a timing-critical point. In addition, memory modules are just temporal solutions, since transient traffic – such as a burst of minimum-sized packets – can cause overflow and packet drop. According to Thesis 1.1, the processing steps of the parsing engine are designed and integrated into a pipelined architecture.

Each pipeline stage has a well-defined task during packet parsing to handle the identification and extraction of an embedded header. Since the core frequency is critical, one stage must contain simple operations on a predefined header.

Based on the parse graph nodes and node transitions, I planned a high-level pipeline architecture (see Figure 2.12), in which the pipeline stages are derived from the parse graph (presented by Figure 2.11). I concatenated the VLAN and MPLS nodes into the second stage, to simplify the architecture.

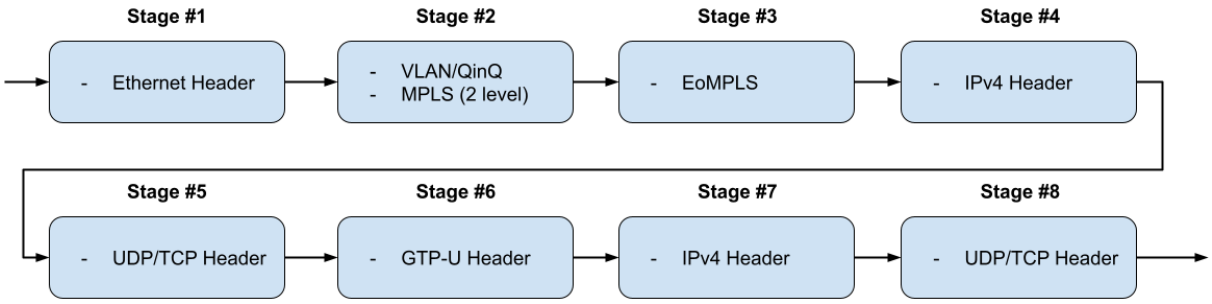

Figure 2.12: High-level overview of the pipeline stages, based on the parse graph.

After the high-level pipeline planning phase, I analyzed the frequency requirement. I found, that some stages must be further splitted, because the operations do not perform the 312.5 MHz requirement. I created the following new pipeline from stages #3, #4, #6 and #7:

- #3: 5 new stages (EoMPLS decoding, PseudoWrite recognizing, VLAN recognizing, IPv4 recognizing and next header start position calculation),
- #4: 4 new stages (Dynamic header length handling, next header start position calculation),
- #6: 3 new stages (GTP-U recognition, TEID extraction, next header start position calculation),
- #7: 4 new stages (Dynamic header length handling, next header start position calculation).

After finalizing the pipeline architecture, it contained 20 stages. To verify the operation of the designed packet parsing system, I applied two, simulation-based verification methods. The first method was based on pcap samples to verify the field extractions. The pcap files were used as test cases during the simulation procedure, and the simulations were done in ISim (Xilinx ISE Simulator).

The applied pcap files contained the following protocol embeddings:

- DHCP messages (Ethernet, IPv4, UDP),
- TCP messages (Ethernet, IPV4, TCP),
- QinQ tunneling (Ethernet, QinQ, IPv4, ICMP),
- MPLS 1 or 2 layers (Ethernet, MPLS, MPLS, IPv4, TCP),
- EoMPLS, with two MPLS layers (Ethernet, MPLS, MPLS, PseudoWire, Ethernet, VLAN, IPv4, ICMP),
- GTP-U tunneling (Ethernet, IPv4, UDP, GTP-U, IPv4, TCP),
- IPv4 header with optional header part (Ethernet, MPLS, IPv4, RSVP),
- IPinIP (Ethernet, IPv4, IPv4, ICMP),
- ARP messages (Ethernet, QinQ, ARP).

The second method was based on randomly generated protocol contents. The protocol hierarchy was also generated by the simulation, based on all possible header combinations and header lengths, and the generated bytes were the input of the pipeline. The output of the pipeline was a bit field (reported into a text file), which contained the recognized header fields. The simulation also reported information about the test cases into the text file (see Figure 2.13). As a result, the simulation examined 10164 different cases, in which the parser recognized all expected protocol headers and fields.

```
5552# -- V0: 1 M0: 2 Eth2: 0 V2: 0 IPv4_0_L: 10 GTP_En: 0 GTP_Flag: 0 IPv4_in_IPv4: 0 IPv4_1_L: 12   --- READY_FIELDS: 1011001100000 : OK
RESULT: D0:OK || S0:OK || V0:OK || M0:OK || M1:OK || IPv4_0_SA:OK || IPv4_0_DA:OK || UDP/TCP_0_SP:OK || UDP/TCP_0_DP:OK || FULL : OK

5553# -- V0: 1 M0: 2 Eth2: 0 V2: 0 IPv4_0_L: 10 GTP_En: 0 GTP_Flag: 0 IPv4_in_IPv4: 0 IPv4_1_L: 13   --- READY_FIELDS: 1011001100000 : OK
RESULT: D0:OK || S0:OK || V0:OK || M0:OK || M1:OK || IPv4_0_SA:OK || IPv4_0_DA:OK || UDP/TCP_0_SP:OK || UDP/TCP_0_DP:OK || FULL : OK

5554# -- V0: 1 M0: 2 Eth2: 0 V2: 0 IPv4_0_L: 10 GTP_En: 0 GTP_Flag: 0 IPv4_in_IPv4: 0 IPv4_1_L: 14   --- READY_FIELDS: 1011001100000 : OK
RESULT: D0:OK || S0:OK || V0:OK || M0:OK || M1:OK || IPv4_0_SA:OK || IPv4_0_DA:OK || UDP/TCP_0_SP:OK || UDP/TCP_0_DP:OK || FULL : OK

5555# -- V0: 1 M0: 2 Eth2: 0 V2: 0 IPv4_0_L: 10 GTP_En: 0 GTP_Flag: 0 IPv4_in_IPv4: 0 IPv4_1_L: 15   --- READY_FIELDS: 1011001100000 : OK
RESULT: D0:OK || S0:OK || V0:OK || M0:OK || M1:OK || IPv4_0_SA:OK || IPv4_0_DA:OK || UDP/TCP_0_SP:OK || UDP/TCP_0_DP:OK || FULL : OK

5556# -- V0: 1 M0: 2 Eth2: 0 V2: 0 IPv4_0_L: 10 GTP_En: 0 GTP_Flag: 0 IPv4_in_IPv4: 1 IPv4_1_L: 5   --- READY_FIELDS: 1011001000110 : OK
RESULT: D0:OK || S0:OK || V0:OK || M0:OK || M1:OK || IPv4_0_SA:OK || IPv4_0_DA:OK || IPv4_1_SA:OK || IPv4_1_DA:OK || UDP/TCP_1_SP:OK
|| UDP/TCP_1_DP:OK || FULL : OK

5557# -- V0: 1 M0: 2 Eth2: 0 V2: 0 IPv4_0_L: 10 GTP_En: 0 GTP_Flag: 0 IPv4_in_IPv4: 1 IPv4_1_L: 6   --- READY_FIELDS: 1011001000110 : OK
RESULT: D0:OK || S0:OK || V0:OK || M0:OK || M1:OK || IPv4_0_SA:OK || IPv4_0_DA:OK || IPv4_1_SA:OK || IPv4_1_DA:OK || UDP/TCP_1_SP:OK
|| UDP/TCP_1_DP:OK || FULL : OK
```

Figure 2.13: Sample of the script-based testing results

To validate the operation of the hardware-based packet parser solution, I applied a test setup from two C-GEP devices and a PC with a 10 Gbit/s throghput capable network interface card (see Figure 2.14).

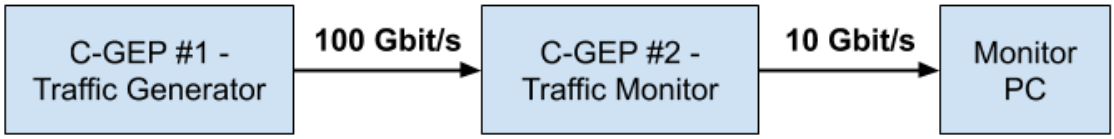

Figure 2.14: Test setup for the validation phase

Since the FPGA chip in the C-GEP device is reconfigurable, it can be also configured as traffic generator. Configuring the platform as an IP packet generator, it can generate multi-encapsulated traffic from the board to stress test the packet parsing engine. I programmed the traffic generator to C-GEP #1 and the packet parsing engine to C-GEP #2. To validate the extracted header fields during operation, I also implemented a packet classifier module, which applied the recognized 14 tuples in filter rules. The C-GEP #2 forwards the fitting packets to the PC, where I captured the incoming traffic with Wireshark [31]. The test setup and the operation of the packet processing engine were presented at IM-2015 (IFIP/IEEE International Symposium on Integrated Network Management) and at HPSR-2015 (IEEE 16^{th} International Conference on High-Performance Switching and Routing) conferences.

2.4.4 Conclusion about FPGA-based parsing

While the evolution of network and telecommunication infrastructures is driven by two factors: the increasing amount of user data and the emergence of new services, there is a novel paradigm, namely the concept of programmable networks that shapes and redefines the architecture of IP-based network infrastructures. The throughput grows also constantly, and new protocols or protocol embeddings appear frequently within the network nodes. All of the mentioned changes together necessitate a new adaptive way of network management. To give performance as well as flexibility, reconfigurable hardware may appear as the central building block of a high-performance network management system.

An FPGA device can be a good alternative for a real-time, lossless packet pre-processing system, because of the parallel architecture and its support of high-end interface technology. To parse on-line a high volume network traffic in a 100 Gbit/s core network, the designed architecture within the FPGA chip should be adapted to the new requirements.

These requirements include the following:

- should process 1.5 x 10^8 packets per second in worst-case,
- should process 1 minimum-sized Ethernet frame in 1 clock cycle,
- should support wide, even 512-bit wide data path,
- should support relatively high, even 312.5 MHz operation frequency.

The general Media Access Control (MAC) operation modes for 1 Gbit/s and 10 Gbit/s are clear and easy to work with. At this speed, the 8-bit or 64-bit wide data path is a well-established solution, and Finite State Machines (FSMs) could cover the problem spaces. 100 Gbit/s is a big step from the previous standards (1 and 10 Gbit/s), not only in frequency but in data path width and operation modes too. The segmented and non-segmented 100 Gbit/s MAC operation modes also have advantages and drawbacks, which can be a design bottleneck in a 100 Gbit/s system, according to the type of the FPGA chip.

During my research work, I synthesized a new method to support the designing phase of an adaptive, real-time, lossless, high-throughput capable, hardware-based packet parsing engine. I proposed the limitations of the hardware-based parsing, and also presented the minimum data path width and frequency requirements. I found that the parse graph is the best practice to model multi-protocol embeddings in a frequently changing environment. The nodes and the node transitions could be fitted into a pipeline architecture, which suits well the physical architecture of the FPGA chip.

Based on the proposed method, I demonstrated (designed, verified and also validated) a packet parser engine on high-throughput capable network device (C-GEP) with several protocol embeddings and traffic samples. My approach is able to operate on 8 different protocol headers at 100 Gbit/s throughput, and extracts 14 different tuples.

Chapter 3

Control traffic analysis of IMS-based services

3.1 Introduction to VoLTE services

Wireless data traffic is increasing exponentially worldwide [32]. The IP-packet based voice services, due to the improved service quality and the high-performance of personal devices, became the dominators of the global voice communication in the last couple of years.

VoLTE uses the LTE (Long Term Evolution) infrastructure, as its name suggests. Beside this, it utilizes IP Multimedia Subsystem (IMS), which is responsible for service control - that includes call control, among others - adds another factor of complexity. IMS has a quite different type of organizational philosophy when compared to those of the mobile cores [33]. IMS is a set of entities where telecommunication operators handle the IP-based call control function, and various other control functions for IP-based services. Moreover, the VoLTE service has to interact with the 2G and 3G infrastructure, since callers and callees are often attached to those, while simultaneously being attached to 4G - and this can complicate the call tracing.

Supporting and managing this growth of traffic on the signaling links poses a great challenge to the service and network operators. A key management task is to obtain direct feedback about the quality of the provided voice service. At this point, the most important question is how the provider can monitor the service whether it complies with the raised expectation against user experience.

The operator has two options:

- monitor call control traffic, and
- analyze the packet-level QoS parameters (e.g., delay, delay variation, packet loss and reordering).

From the network and service management point of view, it is important to gather and analyze the control traffic - especially the detection and the root cause analysis of failures. Fault management has become very complex and requires deep telecommunication knowledge. The main challenge here is to understand the 2G, 3G, 4G and also 5G mobile

core - and not only their architectures but their interworking as well. To see a proper big picture with all the details, the control traffic captured on various interfaces has to be correlated - i.e. identifiers appearing on one interface help to assemble session data records of an other interface [34]. There are key identifiers that get mapped from one to the other while the subscriber is wandering among these technologies; these should be tracked and handled properly.

The traffic capturing task is often based on passive network TAPs, since the interfaces between the network nodes are closed, and the communication should not be distracted. Passive monitoring is supposed to be lossless: when the links are tapped, and the probes receive data in a non-intrusive manner, they cannot ask for re-sending anything. What they missed seeing, they have lost capturing. Based on the monitoring data, engineers can support performance management, network optimization, as well as failure detection, which is one of the most important tasks for operations and maintenance.

Although many network errors could be eliminated based on a continuous control traffic analysis, the user data, which is sent directly by the user equipment, is invisible in this layer. The control protocols grant authentication and authorization procedures, call session setup and breakdown mechanisms, and tunnel setup for the user data.

This chapter presents novel methods for assisting network and service management tasks. It presents a novel assembly mechanism for passive monitoring-based SIP CDR (Call Data Record) generation, and also defines the key parameters of the SIP protocol to follow a call session through the IMS-related interfaces.

The current chapter is organized as follows. Chapter 3.2 provides a theoretical (including an overview about network assessment solutions) and Chapter 3.3 presents a technical recap for the main material of Chapter 3, covering the LTE and IMS architecture, and some monitoring background for SIP (Session Initiation Protocol) specialties. Chapter 3.4 introduces the new results based on the specific challenges.

3.2 Theoretical background for control plane analysis

To analyze the control plane performance of the IMS domain (and also of the VoLTE system), specific message sequences need to be considered. To evaluate these control traffic related use-cases (e.g., call setup and location update) specific methods, key parameters and performance indicators need to be defined.

3.2.1 Performance indicators for core network signaling

Network Key Performance Indicators (KPIs) provide feedback about the core network performance. It helps to take decisions about dimensioning or detect network failures and bottlenecks. To sum up the VoLTE related KPIs, there are two 3GPP standards [35] [36] which defines control signaling-based indicators. The [35] document specifies Evolved Packet Core (EPC) related metrics, and [36] defines IP Multimedia Subsystem (IMS) related measurement methods.

LTE-EPC related indicators

To analyze the LTE core network, the KPIs could be categorized as follows: accessibility, mobility and utilization KPIs. Accessibility KPIs (i.e., attach success ratio, bearer creation success rate, bearer setup time and service request success rate) are specific S1AP and GTP-C protocol-based metrics. The attach success ratio represents the rate of successful authentication, which is based on S6a interface-specific Diameter transactions. The bearer creation related metrics represents the tunnel creation success rate for the user traffic (e.g., public Internet traffic or IMS-related voice and control traffic). The service request success ratio defines the LTE service acceptability. Mobility KPIs provide feedback about incoming and outgoing handover events, and tracking area (radio cell) changing events. Handover metrics are used to evaluate the service stability and reliability. Utilization KPI (i.e., active EPS bearer utilization) describes the rate of the current and the maximum number of achievable dedicated bearers (GTP tunnels).

P. Varga et al. [37] proposed the challenges and solutions for LTE network monitoring, and summarized the LTE-related KPI-s (see Table 3.1) differentiated by interface identifier and protocol type.

IMS-related indicators

The 3GPP-defined [36] IMS KPIs are categorized as follows: accessibility, retainability and utilization. The IMS-related KPIs are mainly based on the SIP signaling protocol. The accessibility KPIs provide feedback about registration and re-registration success into the S-CSCF (Serving Call/Session Control Function) submodule, session setup time and session establishment success ratio from the user side, and also from the IMS side. Call or session drop ratio could be used to evaluate retainability, and the mean of the answered sessions could be used to evaluate the performance of the network nodes, and the stability of the network.

The status of the user registrations and the initial registration success rate also evaluates the performance of the IMS domain. The session setup time could be used to estimate the

KPI	LTE Interface	Protocol type
Attach success ratio	S1-MME	S1AP
Dedicated bearer average and maximum setup time	S1-MME	S1AP
Service Request success ratio	S1-MME	S1AP
Connection drop ratio	S1-MME	S1AP
Create session success ratio	S5/S11	GTP-C
Create bearer success ratio	S5/S11	GTP-C
Modify bearer success ratio	S5/S11	GTP-C
Delete session success ratio	S5/S11	GTP-C
Downlink Data Notification success ratio	S5/S11	GTP-C
Update Location success ratio	S6a	Diameter
Authentication Information success ratio	S6a	Diameter
User data traffic related statistics	S1-U/S5	GTP-U

Table 3.1: LTE related core network KPIs [37]

users satisfaction. It should be noted that the session establishment success ratio should be calculated from the users and also from the network's perspective.

3.2.2 State-of-the-art analysis of network assessment solutions

Breda and Mendes [38] proposed a QoS monitoring and failure detection solution in voice communication. The algorithm is based on Call Data Records (CDRs). The proposed system classifies the generated CDRs into predefined events (e.g., Carrier loss, Called party does not answer).

Lutiis and Lombardo [39] proposed a monitoring tool that focuses on the security features for IMS (IP Multimedia Subsystem). The authors describe that the main challenge for a monitoring system is to handle the ever-growing subscriber base, and it must be scalable to work properly even during the peak hours. The proposed anomaly detection algorithm (SAD - SIP Anomaly Detection) is based on CDRs, and it calculates an E entropy value for 60 minutes time-windows.

Nassar et al. [40] proposed an intrusion detection method for SIP protocol based IMS interfaces. The work is based on anomaly and attack classification. Besides the SIP record profiling, they also examine the billing information and the server logs.

Raouyane et al. [41] aimed to verify the QoS in IMS networks using eTOM (enhanced Telecom Operations Map), as an IMS monitoring and Management system. The authors applied web services to demonstrate the distributed ar- chitecture. The work defines the required data collection points of the network, and calculates general KPIs (e.g., response time and availability).

Hoffstadt et al. [42] designed a SIP protocol-based monitoring and analyzing system (STR - SIP Trace Recorder). The message flow can be differentiated into three steps: capturing, parsing and storing into database. During the automatic analysis phase, the STR system is used to analyze SIP-based attacks in different scenarios. To evaluate the system, the authors generated attack signatures.

L. Zhang et al. [43] proposed a monitoring system for SIP protocol, which is able to deal with 1 Gbit/s speed traffic flow. The paper deeply analyzes the SIP session and call procedures. To test the capability of the work, the authors used their proprietary laboratory environment as a test-bed. The results show that the software-based implementation is able to deal with 6000 simultaneous SIP sessions.

J. Balcerzak et al. [44] proposed a monitoring model based on the registration procedure, in order to detect anomalies in IMS domains. Since the SIP Register messages take a great proportion (in their model: about 35 percent) of the total SIP signaling of the IMS, it could be an efficient base for first level problem diagnosis. The authors defined different KPIs based on registration-specific counters.

J. Hyun et al. [45] proposed a Deep Packet Inspection (DPI) method to classify VoLTE traffic, using the SIP User-Agent field. The classification method is able to achieve 3.8 Gbit/s processing speed to differentiate SIP and also RTP traffic. The authors used captured, real LTE network traffic for the evaluation phase. They further elaborated this work in [46].

Li et. al. [47] presented their security assessment methods and metrics on one of the first VoLTE deployments. They discovered several vulnerabilities in both the control and the data plane. During their analysis, they had various interesting findings, including that the device OS and chipset fail to prohibit non-VoLTE apps from accessing and injecting packets into VoLTE control and data planes. Such vulnerabilities pose extra tasks on network failure management, as well.

3.3 Technical background for Voice over LTE analysis

3.3.1 LTE Architecture

The LTE (Long Term Evolution) is standardized by ETSI (European Telecommunications Standards Institute) as a packet-optimized, wireless data communications technology for mobile devices. The LTE projects launched to study requirements for a new air interface (focusing on the PS domain), called Evolved Universal Terrestrial Radio Access (E-UTRA). Some of the LTE requirements are: 100 Mbps downlink and 50 Mbps uplink peak data rates, reduced latency, scalable bandwidth and support for interworking with existing 2G/3G technologies. The LTE Evolved Packet Core (EPC) [48] comprises merely packet-switched network elements; it only supports the legacy circuit-switched functions through IP-based packet transfer. The network elements (see Figure 3.1 and 3.2) in the core network architecture supports the user authentication and authorization procedures, handover operations between base stations, GTP tunnel setup (and removal) for the public data network and Single Radio Voice Call Continuity (SRVCC) to other telecommunication architectures. The standard also introduces new control protocols between the functional nodes: S1AP (S1 Application Protocol), Diameter and GTPv2.

The main functional entities of the LTE EPC are briefly the following.

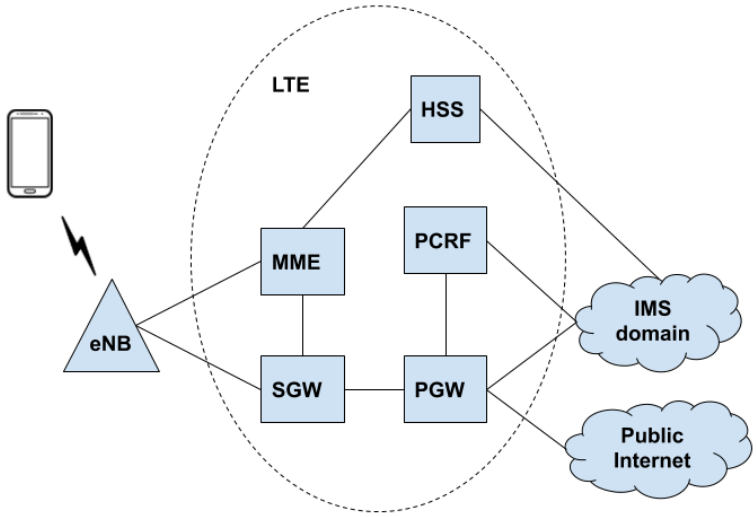

Figure 3.1: LTE core network architecture

The MME (Mobility Management Entity) lies on the border of the EPC and EUTRAN (Evolved Universal Mobile Telecommunications System Terrestrial Radio Access Network) - and is mainly responsible for mobility management. Its main functions include controlling handovers between eNodeBs (Evolved NodeB, Base Station), SGWs (Serving Gateways) or MMEs, user identification, authentication and controlling the roaming functions. MME registers and handles the User Equipment (UE) in his own area, registers where the UE is located, either exactly at eNodeB level (if communication is active), or within a Tracking Area (TA), which is designated to a group of eNodeBs (in case of passive UE, no active connection is needed).

The SGW (Serving Gateway) is responsible for user traffic stream handling, and controlling the allocation of resource capacities (e.g., the changes or deletion of sessions and finishing IP connections). From the eNodeB point of view, it is a fixed anchor node to access the public internet or the IMS domain. The SGW controls the User Plane tunnels with GPRS Tunneling Protocol (GTP) [44], although this can also be guided by the MME or the PGW - depending on the process rules.

PGW (Packet Data Network Gateway) can be seen as an edge node of the EPC, since it ensures the connections to external data networks (e.g., the Internet or a private corporate network) and handles the UE's data traffic that is entering or leaving the core network. Besides, it offers interfaces for further functions such as QoS control or billing.

HSS (Home Subscriber Server) takes the roles of HLR/AuC (Home Location Register/Authentication Centre) in the LTE network. It can be seen as the ultimate data storage that contains the subscribers' service-related data. The HSS stores the profile of the subscribers, containing the enabled services and accesses (e.g., allowed roaming services to external networks). It is mainly responsible for access management and authentication, and it also registers the subscribers' position within the network. The HSS cooperates with the MME in all UE-related change events that are administered by the EPC.

PCRF (Policy and Charging Rules Function) is based on its predefined policies and QoS-related rules, it sends control information to the PGW. This set of information is called "Policy Control and Charging rules", and it is exchanged between the PCRF and the PGW when a new bearer is set up, e.g., in case a new UE activates new PDP to the network or a new UE requires a data plane bearer with a different QoS policy.

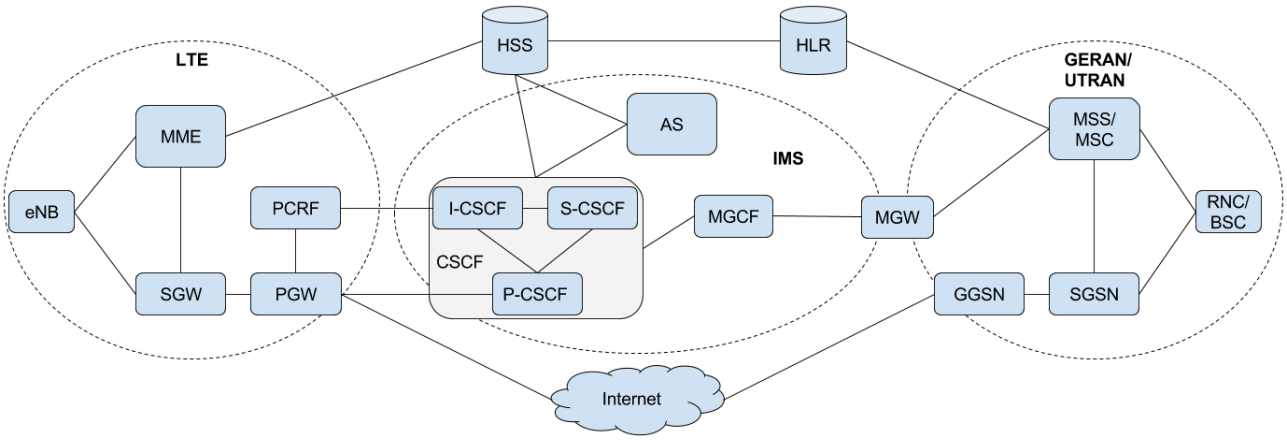

Figure 3.2: Core network architecture serving Voice over LTE - including 2G, 3G, 4G and IMS components

3.3.2 IMS Architecture and monitoring points

The IMS (IP Multimedia Subsystem) is a global architecture, constructed as a standardized platform [33] for the different telecommunication networks. The basic idea is to produce a common, Internet-based architecture to grant communication between different telecommunication networking technologies (e.g., 2G, 3G, 4G, 5G, fixed phone, Internet).

The main benefits of IMS are:

- Homogenized handling of multimedia services within the control plane,
- Providing QoS support for real-time multimedia services,
- Enabling IP-based multimedia services for mobile users.

Since these different technologies use various call control protocols in their various interfaces, IMS introduced a homogeneous usage of call control, through SIP (Session Initiation Protocol, [49], [50]) protocol. This also means that technology-dependent call control protocol messages and parameters need to be translated to SIP. Such protocols are, e.g., ISUP (ISDN User Part) [51] between fixed telephone exchanges, or BSSAP (Base Station System Application Part) [52] for the 2G access network. There are standardized protocol converter functions defined in order to handle these multi-protocol dependencies. As an example, MGCF (Media Gateway Control Function, [53]) operates at the edge of the IMS domain to cover this functionality, among others. Figure 3.3 depicts the architecture of an IMS domain.

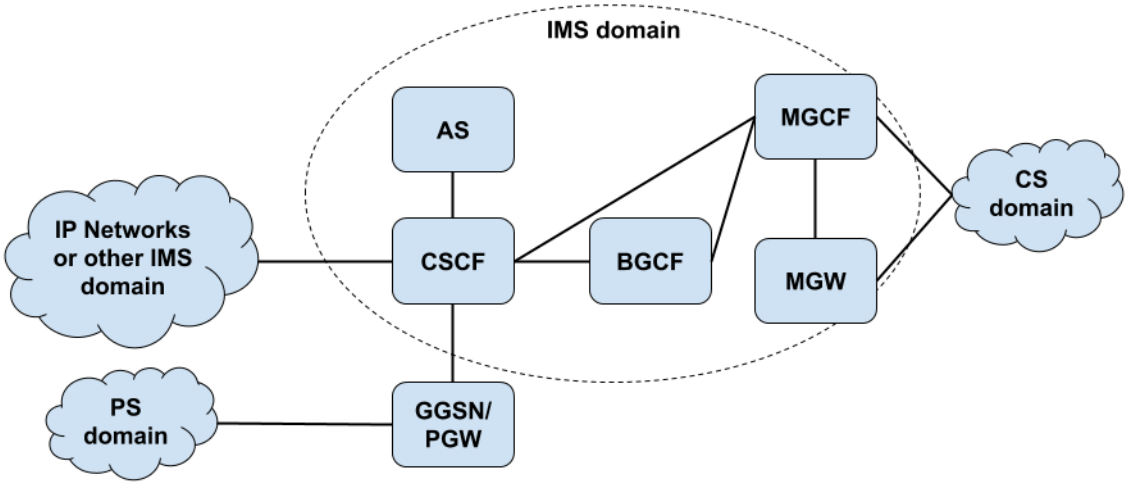

Figure 3.3: IMS interworking with other networks

The central IMS modules often serve millions of users, which opens major networking challenges to solve. In order to reduce the network load, the traffic can be distributed by using prefilter modules. However, it should be noted that the centralized functions obstruct scalability, hence they could lead to architectural drawbacks. IMS is essentially a control system [54], which enables for the registered users to find other users and application servers - in order to build up multimedia connections (e.g., conference call or instant messaging).

The architecture manages and maintains merely the control messages; the media traffic is routed independently of the control traffic between the end-users (e.g., RTP protocol-based user traffic).

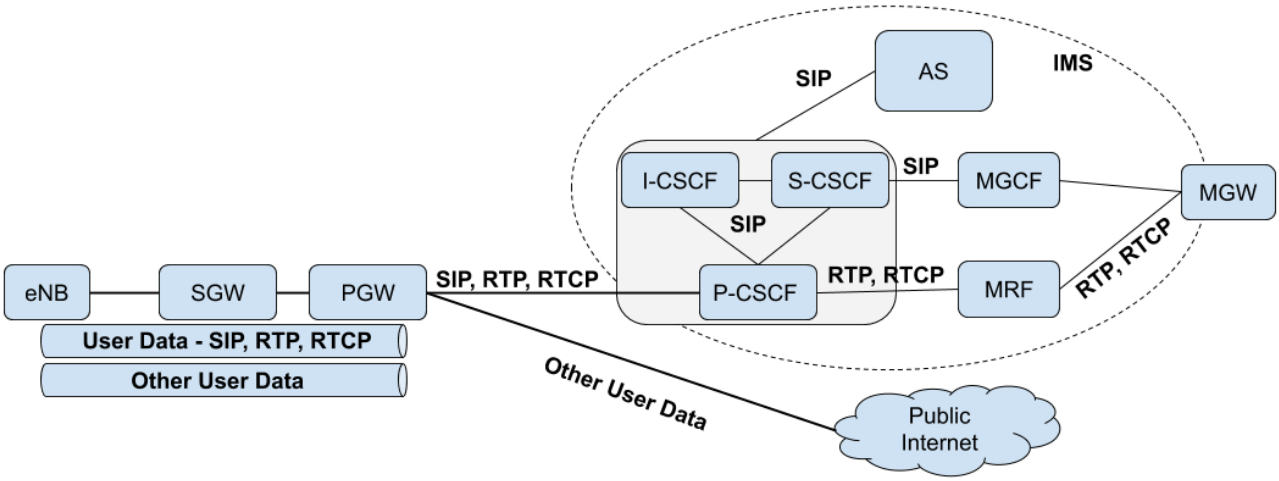

Figure 3.4: User data traffic differentiation

The central element of the architecture is the CSCF (Call Session Control Function). It is functionally splitted into three submodules:

- P-CSCF (Proxy Call/Session Control Function),
- S-CSCF (Serving Call/Session Control Function), and
- I-CSCF (Interrogating Call/Session Control Function).

Figure 3.4 represents the differentiated CSCF-functions within a domain. These submodules serve routing and management tasks, and operate together as the main CSCF network node.

The P-CSCF part is the first connection point at the edge of the IMS domain, which allows a secure entry point for the end-users. It often uses IPSec to grant secrecy for IMS-related traffic. It also routes the incoming control and media traffic to the proper node (e.g., registration-related traffic to the I-CSCF, session establishment messages to the S-CSCF, media-related packets to the MGW).

The main task of the S-CSCF module is to build up and manage connections between the IMS-registered users. It operates as the central element of the signaling plane and handles the registration and user services. During the user authentication procedure, the S-CSCF requests user-related information from the AuC (Authentication Center) through the HSS (Home Subscriber Server), which are key elements of 3G and 4G mobile architectures. After the successful registration, it handles the user-control messages and acts as an anchor point for the registered users.

The Interrogating part - so-called I-CSCF - is located at the edge of the IMS domain, and acts as a forwarding point for remote servers. In general, it supports the communication from and towards other IMS domains [54]. The central control plane grants access independency

for the end-users, through protocol converter components and gateway components towards the Internet world.

The MGCF (Media Gateway Control Function) module supports ISUP - SIP signaling protocol conversion. It enables the connection from 2G/3G mobile networks.

The MSAN (Multi-Service Access Node) network element is for fixed phones; it allows communication through the IP-based systems. VoIP and VoLTE phones or other devices with SIP protocol capabilities are connected through IP-clouds to the IMS domain, using a GGSN, a PDN GW (Packet Data Network Gateway), or just a router, depending on the media access.

Besides CSCF, IMS contains further modules (e.g., Application Server - AS, Media Resource Function - MRFC, SBC - Session Border Controller), which are differentiated based on their services. This dissertation concentrates only on those main nodes, which play key roles in a session's lifetime.

The Application Server is directly connected to the S-CSCF submodule. It is necessary to allow conference calls, charging and other multimedia services for the registered users. The module contains the management logic itself to provide and execute multimedia services (e.g., videoconferencing, gaming, file sharing).

The MRF (Media Resource Function) is responsible for media stream control. It can be separated into MRFP (Media Resource Function Processor) and MRFC (Media Resource Function Control) submodules. MRFP supports media manipulation (e.g., playback function, playing tones), conferencing, DTMF user activity. MRFC is the controller part, which manages the resources of MRFP.

VoLTE interface overview and monitoring points

VoLTE service assessment is challenging for the operators, because there is an overwhelming variety of important interfaces between LTE EPC and IMS nodes (see Figure 3.5), which should be examined by online monitoring and analyzing systems. The following short overview describes the interfaces of the main architectural elements. For further, detailed reading on VoLTE architecture, refer to [55]. Focusing just on the LTE EPC architecture, the S1-MME, S1-U, S6a, S11, Sv and SGi interfaces are the main points to monitor for control traffic analysis. The S1-MME interface is used between the eNodeB and MME to carry information on authentication, bearer (de/)activation, handover, or tracking area update procedures. The S11 interface is used to control the default and dedicated bearer establishment for the UE through a tunneling mechanism. The tunnel or bearer is activated for the user traffic on the S1-U interface between the eNodeB and SGW, and the PGW provides the final routing towards the public Internet or the IMS cloud.

The SGi interface between PGW and P-CSCF carries the SIP protocol, which is encapsulated into a tunneling protocol (GTP) through the S1-U interface. To analyze the IMS registration procedure, the Rx, Gx, Cx and SGi interface should be monitored. The Cx interface is used by the CSCF (Call Session Control Function) for authentication key requesting, and the Rx/Gx interface is used to inform the PCRF about default or dedicated bearer activation. For an efficient root cause analysis in the IMS domain, the ingress and egress ports of the CSCF nodes and the AS should be monitored. To analyze the PS to CS switching mechanisms, the MGCF (Media Gateway Control Function) and the Sv interface

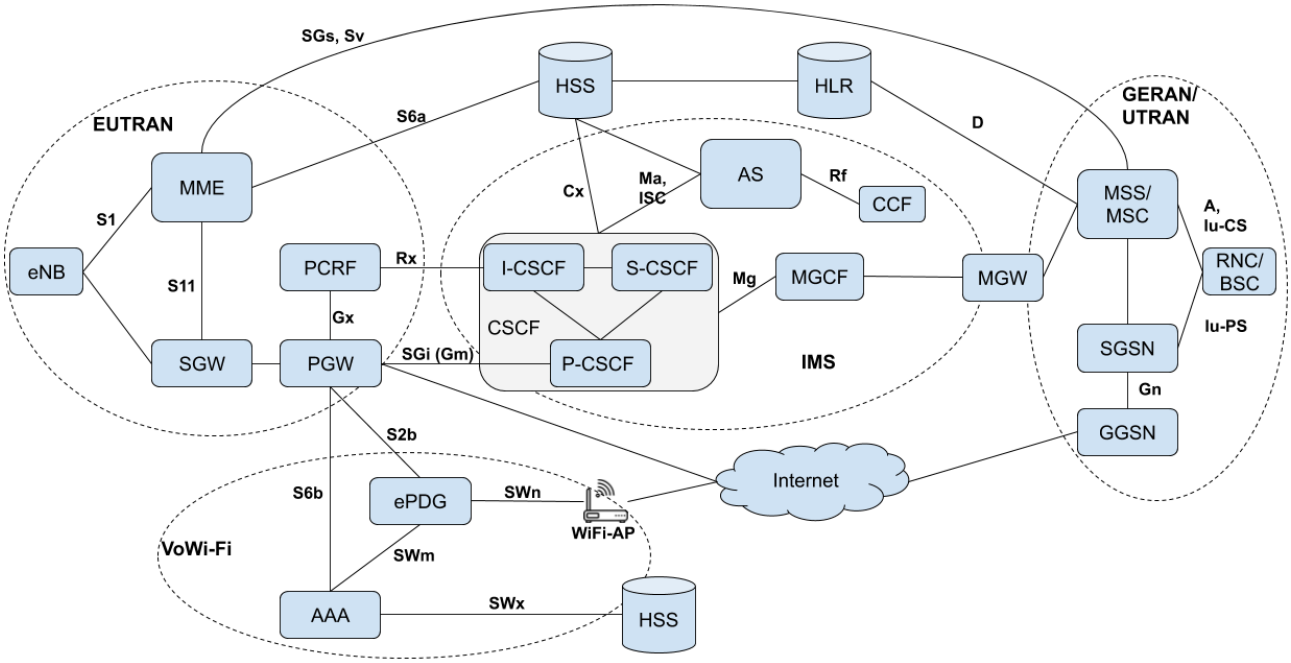

Figure 3.5: Core network interfaces in accordance with VoLTE

should be monitored.

The P-CSCF plays a security guard's role, all its external connections are ciphered through IPSec. This makes the monitoring of this interface very complex - and still, it reveals very little about the IMS's behavior. Since the P-CSCF is mostly responsible for security-related issues, faults during call control can be captured on the internal leg of P-CSCF, as well. It would make sense to monitor this leg - in theory. However, in practice, IMS vendors often integrate the logical P/I/S-CSCF modules into one physical entity. This means that the internal IMS interfaces are not available for any kind of monitoring. On the other hand, if any traffic between the P/I/S-CSCF entities is accessible, it is worth taking that/those interface(s) into account as extra monitoring point(s). The Application Server and the MMTel (Multimedia Telephony Service) modules of IMS get call control information as well, since they have to maintain data related to charging, user authorization and QoS-control. This way the external interfaces of these modules should also be considered as monitoring points.

3.3.3 SIP protocol

The dominant signaling protocol of the IMS domain is SIP (Session Initiation Protocol). It uses IPv4 or IPv6 in the network layer, and UDP, TCP or SCTP in the transport layer. Since it is a UTF-8 text-based protocol, the order of the fields can be varied - as well as the size and the number of parameters. The header field format is based on the HTTP header fields syntax and semantics.

Explaining the tasks of each message and header field is not within the scope of this section, it focuses only on call setup related parameters that are needed for understanding CDR generation algorithms.

Request

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;
     branch=z9hG4bKnas
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;
      tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

Response

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pc33.atlanta.com;
     branch=z9hG4bKnas;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;
      tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0
```

Figure 3.6: Request and Response message example - RFC-3261 (Section 24.2)

Figure 3.6 represents an example request message content. Generally, the To field specifies the callee and the From field indicates the caller during a multimedia conversation. The Call-ID field defines a case-sensitive, unique identifier for a coherent message sequence, and the CSeq field grants the transaction orderliness with a sequence number and a method definition. Fields with P- prefix (e.g., P-Asserted-Identity, P-Access-Network-Info) include private, user-related information about the authentication and authorization procedures, tracking area identifiers, charging information, etc. Optionally, the SIP message carries an embedded content (e.g., another SIP message, an XML content, or a Session Description Protocol part), which is indicated by the Content-type field.

Based on the best practices of protocol technology, we can differentiate two types of messages: Request and Response. The first line of each message has a mandatory format, which greatly supports the type-recognition process. According to the SIP terminology, the first line of a Request message is called Request-Line, and its parameters are Method, Request-URI and SIP version (see Figure 3.6). The basic request message types are Register, Invite, Bye, Cancel, Options, Ack, Subscribe, Notify, Message, and Prack [49], [56]. Among these, the Invite message marks the start of each call - hence it is very important when generating SIP call data records. The terminology also differentiates initial-Invite and Re-Invite methods. The Re-Invite message contains a tag parameter in the To field, which

indicates, that the session setup was successful, and the Dialog-ID is complete. Since the tag parameter of the To field is generated by the receiver side, it is missing from the first Invite message, which is called as initial-Invite.

Parsing a Response message, the first line is called as Status-line, in which the mandatory parameters are: SIP version, Status code and Reason phrase (see Figure 3.6). The standard also defines the communication partners. The UAC (User Agent Client) generates the Request message, and the UAS (User Agent Server) generates the Response.

Actually, the Status-line of a Response message has two parts: a (numerical) Status code and a (textual) description (e.g., 200 OK). The first digit of the 3-digit Status code is the class code. The 1XX class is for the provisional responses (e.g., Trying, Ringing, Session Progress). The 2XX class means successful operation, and 3XX reports a redirection event. The 4XX, 5XX and 6XX classes are used to indicate error or failure events. The SIP terminology differentiates three types of communication: transaction, dialog and session.

A SIP transaction starts with a request message and ends with a final response. A SIP dialog is a longer conversation between the end-users, containing many transactions. A dialog can be clearly identified by the Dialog-ID, which is concatenated from the value of the Call-ID field, the tag parameter of the From field, and the tag parameter of the To field. This triplet defines a SIP Dialog between the UAC and UAS. Based on the request message types, two Dialogs can be differentiated: the Invite- and the Subscribe-dialog. The Subscribe-dialog is used to sign up for a service or listening to an event. The Invite-dialog sets up a multimedia (e.g., for audio, video, gaming) session, called as SIP Session. The Session is built-up successfully when a three-way handshake (INVITE, 200 OK, ACK messages) at the beginning of the Dialog is successful. The end of a Session is not so trivial, the following categories could be recognized:

- BYE transaction ended session,
- CANCEL transaction ended session, and
- Failure code ended session.

As SIP is a text-based protocol, in which the order of the fields is not predefined, the message parsing and information searching is a challenging procedure. The IMS nodes (e.g., CSCF) handle internal databases and records: huge amount of data about the session states and user-related information. Since the header format and the length are not fixed, the nodes often translate the internal database contents into UTF-8 text and send it through the network. It is an easier way at the node side, but more challenging at the network operators side, not to mention the network monitoring system side. Searching information in a SIP message header raises further challenges:

- Some fields appear more times in the header, with different contents.
- A SIP message can encapsulate another SIP message (this property is marked in the Content-Type field), which often contains user identifiers (e.g., IMSI).
- The MSISDN of the caller or callee appears in many fields in different formats (e.g., with or without MCC/MNC, with or without sip: / tel: prefix).

3.3.4 Call Data Records

A continuous supervision of a complex VoLTE telecommunication architecture requires efficient, task-optimized models for the network and service operators. Since the architecture is distributed and heterogenous, to capture, store and analyze the ever-growing control traffic bit-by-bit seems an almost impossible mission. That is the reason, why the operators use higher-level information sets for daily service inspection. To get a high-level overview about the network node communication, a widespread model is the Call Data Record (CDR). CDRs are data records that contain extracted information from a predefined message sequence. However, the name refers to tracking call-related information, it is often used for tracking all activity. Generally, it belongs to one interface or one node communication, but it could also summarize more interfaces (this is what operators prefer). It depends on the object.

To sum up the properties, the CDRs are used for:

- tracking end-user activity (e.g., authorization and authentication, tunnel creation/deletion),
- tracking call activity (e.g., it summarizes information about call durations, call originations and destinations, call forwarding, conference participants)
- billing verification (it contains precise, event-defined timestamps)
- failure detection (it should support a proper root-cause analysis procedure)
- service quality verification (e.g., call quality estimation)

Since Call Data Records contain compact information from several protocol-specific parameters, there are many approaches that apply these records as a basic data set for further analysis. CDRs are also essential starting points for automatic failure and network anomaly detection [57], criminal network analysis [58], telecommunication architecture dimensioning [59], tourist identification [60], marketing goals and customer clastering [61], or event estimating dependency of call duration and call quality [62].

The CDRs often have two parts:

- metadata, and
- message digest.

The metadata part often contains the type of the record (e.g., the type of the control event), record opening and closing events, IP addresses, event timestamps, extracted or derived user identifiers (e.g., IMSI, MSISDN, IMEI) and key parameters for further processing mechanisms. The message digest implies the message sequence, retransmission events, failure events and extracted, service-specific protocol parameters.

CDRs contain high-level information about all calls or event-specific transactions that are generated by the network nodes. Since the VoLTE architecture is heterogeneous and operates with different protocols, a call setup procedure often uses 4 independent control protocols

(i.e., S1AP, Diameter, GTP, SIP). The record generation raises a basic question: How can we assemble and recognize a user activity? As the communication between the network nodes is based on transactions (request - answer message pairs), it could be a starting point. Protocol-specific identifiers could be used to collect message pairs, and the transaction types could be define the event type. In contrast, a call/session setup procedure is a longer message sequence, and the assembling algorithm should handle protocol-specific key parameters to collect the information.

Since the VoLTE service is built on IP-based communication, packet loss event could be occur even during a connection-oriented operation (e.g., QoS-enabled buffer operation). That is the reason, why the expected message sequence could be different from the captured traffic. A treatment method to deal with this property is the timeout definition for waiting for a control message. To define and set a proper, interface-specific timeout value is an operator task, which is an indispensable operational step for a continuous CDR generation mechanism.

3.4 New results in control traffic analysis

Thesis group 2 - I have defined management methods for the signaling plane of the VoLTE ecosystems. I have determined the requirements of constructing a VoLTE Call Data Record to inspect the entire IMS domain. Using the requirements, I have provided a method to assemble SIP protocol-based Call Data Records. I have introduced key parameters to cross-correlate the Call Data Records between the SIP protocol-based IMS interfaces. I have validated the implemented form of the method against industrial conditions. [J3] [C4] [C6]

3.4.1 Challenges with SIP CDRs

Although the VoLTE service has great promises - from high service exibility to previously unseen QoS measures - so far it poses more challenges than expected. The serving architecture is diverse, and the flexible services mean variable use-cases with complex message sequence charts. In order to be able to control the procedures - during deployment as well as during the operation - we need to have proper network and service management in place. Providing technical solutions to cover all requirements of the VoLTE service is still a great challenge for operators. Network monitoring and call data record assembling is one of the important methods to support service verification, deployment and operations. It includes management tools that help to provide better visibility of the various procedures. To get a detailed overview about the VoLTE service, both the LTE Evolved Packet Core (EPC) and the IP Multimedia Subsystem (IMS) need to be monitored. Network-wide data capture and analysis for the EPC and the IMS require new processing methods. These would allow operators to correlate control and user plane information of various interfaces and protocols. There are many obstacles to overcome here, including ciphered control messages and global identifiers hidden by temporary ones.

Call Data Records (CDRs) cannot be assembled by simply capturing signaling on a few given links. Information is fragmented, hence on-the-fly cross-correlation of key parameters is required. In order to effectively utilize the network and service monitoring system, operators need new methods to correlate the information of various interfaces. The challenge is rooted in complexity, CDR assembling is not always trivial because of the complex network implementations. Since in VoLTE services the IMS is the common domain for call and service control procedures, I focused on new SIP session tracking methods and SIP-based CDR assembling algorithms in my research work.

The challenges about SIP-based call control analysis can be traced back to two problem spaces:

- IMS is a distributed, scalable architecture (more nodes = more resources = more complex message sequences),
- Temporary identifiers and hidden parameters.

From the operator's viewpoint, to understand an incorrect call setup procedure, the first step during the root-cause analysis is the MSISDN-based CDR search. The search

results should represent the message sequence and highlight the failure events. The failure detection mechanism could be automatic or unique, task-specific, the basic goal in every case is to grant proper call records as a starting point. A call setup procedure (a SIP message sequence that establishes a Session) should be followed through the IMS-related interfaces within and even between the domains. This is a real challenging problem because of the distributed architecture and the specific node behaviors. The session-related messages appear in several interfaces (in general, the call setup messages travel through 4-5 interfaces in one IMS domain) as we progress through the call setup procedure. To cover two IMS domains, the number of concerned interfaces could be more than 10. Following the two domain-based examples, if we catch all SIP messages within the nodes and differentiate the message sequences by IP addresses, we get more than 10 different records to the same call. Since the records include the same call setup information, the message sequences should be linked and labeled by the same information (e.g., caller/callee MSISDN). To link and label the message sequences is also a challenging task, because the identifiers could be changed during the node transitions. There are certain nodes that change the globally unique Call-ID identifier in the SIP messages. In terms of the MSISDNs, the nodes also hide the MCC/MNC codes or the whole MSISDN, and transform it to "anonymous" text. As we move forward through the interfaces during a call setup procedure, we may have less and less information to label the message sequences (and the call data records). The operators need new methods and key parameters to clearly follow the call setup sequences through all concerned interfaces.

Another challenging task is to define the opening and closing events of a session. This information could be used to verify the trigger events for the billing information. The session closing definition is also important for the record generation algorithm, to close the previously opened records and to indicate that the call is ended. Since the network is distributed, complex uses-cases and node-specific behaviors transform this task to a challenging process. The operators need new definitions to understand the complex message sequences and identify session closing events and exceptions as well.

3.4.2 Thesis 2.1

Thesis 2.1 - I have provided a new SIP protocol-based Call Data Record assembly method for real-life VoLTE scenarios. I have validated the implemented form of the method against industrial conditions. [J3] [C4]

To get a general method for SIP-based Call Data Record assembling, the following requirements should be defined for the message sequences:

- Definition of the record types,
- Identification of the record opening events,
- Key parameters for the transaction and dialog identification,
- Definition of the record closing events.

I investigated each requirement and provided the key element for assembling CDRs from SIP signaling messages within the IMS domain.

Record type definition

I found that SIP transactions and dialogs should be differentiated based on their initial request message. Since some of these request types are initiated (and then closed) asynchronously from each other, different record types should be defined for each. I defined the following Call Data Record types based on these initial messages: INVITE, REGISTER, SUBSCRIBE, OPTIONS, MESSAGE, and INFO. Figure 3.7 depicts the CDR types with their basic CDR-related messages.

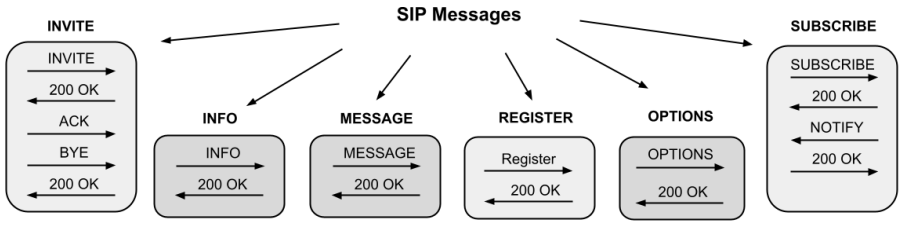

Figure 3.7: Different SIP CDR types

Record opening events

Out of the previously defined record types, the INVITE CDR type is the one that contains the actual Call Data Record, in the classic sense. In other words, an INVITE CDR is a SIP call dialog, which starts from the initial Invite message and finishes at the end of the call (with various ways). Since a call setup procedure contains more transactions, INVITE CDRs include Update [63], Cancel and often Info [64] transactions, as well. Nevertheless, not all Invite messages mark a newly starting call. Some Invite messages do actually start the call, while others arrive during the same call, carrying feature modification information. I found

that the Invite messages should be differentiated in the call flow, based on the value of the To field tag parameter. If the To tag parameter does not exist, it should be considered as an initial-Invite Request (starting a call and also a CDR record), otherwise it is a Re-Invite (modifying parameters, e.g., QoS settings, call hold, RTP port changing) [65]. Considering INVITE CDRs, they start with an initial-Invite request message. The other type of records should start with the first request message (e.g., a REGISTER CDR starts with a SIP Register request message).

Key parameters for transaction and dialog identification

The trivial way to follow a SIP call's control through passive monitoring is by using the RFC-defined [49] Dialog-ID. The Dialog-ID is a parameter derived from three field parameters of the SIP message: Call-ID, From tag and To tag. The Call-ID is a unique ID within the IMS domain at a given time. Besides, the From and To tag parameters are minimum 32-bit wide truly random numbers. Altogether, this triplet (the Dialog-ID) ensures strong identification. The Dialog-ID clearly associates longer message sequences (e.g., the INVITE- or SUBSCRIBE dialogs), and can be used to pair request-response messages of other transactions. The preparation of the Dialog-ID is a two-step process. The initial message between two User Agents never contains a To tag. After receiving the Request message, the UAS generates the Dialog-ID and sends it back within the To field of the Response message. After this tag-exchange procedure, each user knows the full Dialog-ID, and uses it as a unique ID of the call. The IMS also records this ID for routing functions. In the case of simple Request-Response transactions (e.g., OPTIONS, MESSAGE), which contain only two messages, every new Request triggers a new To tag generation process. In contrast, SIP dialogs always use one To tag during the dialog lifecycle. Similarly to the two-message transactions, the initial-Invite message does not contain a To tag. Actually, this is the property that is used to identify it as an initial-Invite.

According to RFC-3261 [49] the UAS generates a To tag parameter (in its response), when the request does not contain it. This also means that the CDR assembling algorithm does not have any To tag to use (as part of the Dialog-ID) until the final Response of the initial-Invite arrives. As an example of such a case, let us take a look at Figure 3.8. This shows a SIP dialog ended by a Cancel transaction. Regarding the call cancellation procedure, the Cancel request message must contain the same From and To field parameters, as the initial-Invite message did. As discussed earlier, the initial-Invite does not contain any To tags, hence this parameter will be empty in the Cancel message, as well. The reception of a Cancel request (with empty To tag) triggers a new tag-generation process - and the 200 OK (CANCEL) response hence contains a new To tag, different to the ones seen in previous dialog messages. As this simple example reveals, although the Dialog-ID is a strong key, I found that its usage in general is cumbersome. Until waiting for the final response message, the Call-ID and the From tag together could be a strong enough, unique CDR identifier. It should be noted that, based on laboratory measurements at a nation-wide telecommunication operator, Call-ID alone is also a strong key. Nevertheless, From and To tag parameters grant a safe and global CDR collection technique.

Furthermore, including endpoint IP addresses in the CDR identifier key is very beneficial; mostly for indexing purposes. Telecommunication operators often analyze the IMS domain's

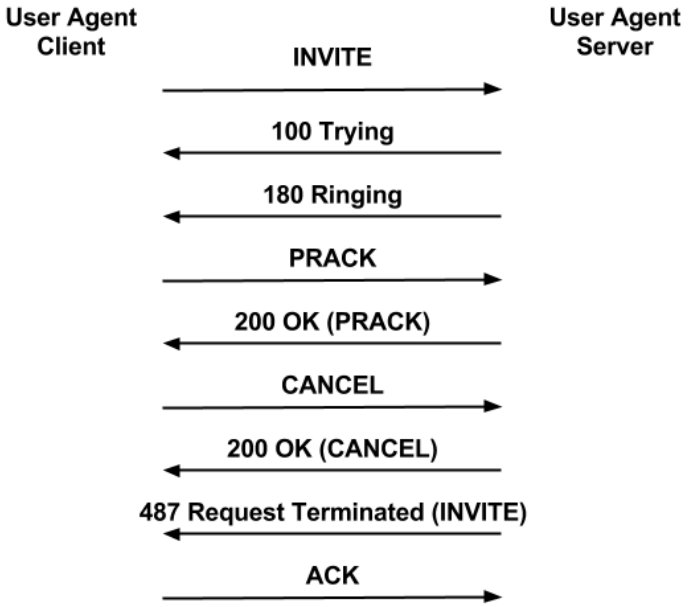

Figure 3.8: Signaling messages of a canceled INVITE dialog

traffic on given signaling links, and they are interested in statistics between network nodes.

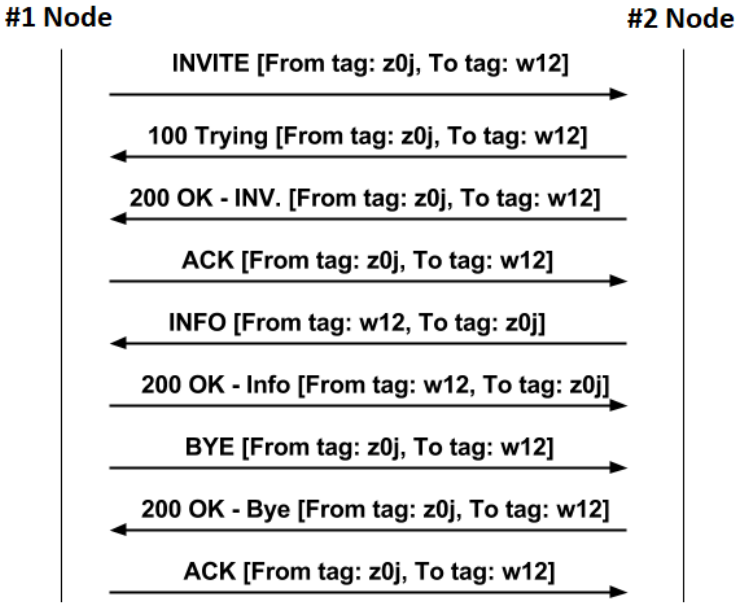

Figure 3.9: From and To field parameter values change their place in INVITE dialog messages

That is the reason why IP addresses should be included (as the second part) in CDR identifier keys. Beside the Dialog-ID triplet, IP addresses support a clear association between calls and network links.

Based on my investigation work I also found that the From and To fields could be swapped within a dialog. However, the RFC [49] recommends that the content of the From and To

fields should be the same, the roles (UAS and UAC), as well as the values, could be changed during the SIP dialog. Using the previously defined general key format (Dialod-ID and IP addresses), the CDR assembling algorithm should support the From and To field exchange procedure. Figure 3.9 depicts an Invite-dialog, in which the From and To field values change their place in the INFO Request-Response messages. The From tag of the 7th message is equal to the To tag of the previous messages. This suggests that field parameters should be handled with certain dynamic placing within the CDR keys.

Record closing events

During the CDR Assembling procedure, opening a new record is clearly based on the initial message of a transaction or a dialog (i.e., initial-Invite message triggers a new CDR in the case of a new SIP call). In contrast, it is a complex task to recognize the record-closing event. According to the RFC papers [49] [66], transactions and dialogs have different closing methods.

The INVITE dialogs complete, if one of the following trigger events occurs:

- "487 Request Terminated" message responds to the initial-Invite (triggered by CANCEL transaction),
- "408 Request Timeout" or "481 Call/Transaction does not exist" message responds to Re-Invite message,
- BYE transaction, if the BYE message contains valid From and To tag parameters,
- 3XX/4XX/5XX/6XX class type Response messages within the Invite-dialog.

The SUSCRIBE dialogs complete, if one of the following events occurs:

- The Expires parameter arrives, or the value of the Expires field is 0,
- The Response message to the Subscribe request arrives with 401, 405, 410, 480-485, 489, 501 or 604 status code,
- UAS does not receive periodic Subscribe messages within the "Expires" interval.

First of all, an IMS node does not initiate the closing of the Invite-dialog after sending a BYE message, until the "200 OK" response does not return. UAs often send periodic BYE (e.g., in 1, 2 or 4 seconds), until a response or a timeout event. RFC papers [49] [67] often refer to the 4XX Response messages as transaction- or dialog-closing messages.

Regarding the examination of SIP signaling in real network scenarios, I identified that there are Status code exceptions that belong to the 4XX (client failure) class and do not finalize a dialog or a transaction sequence. These identified exceptions are summarized in Table 3.2. The CDR assembling algorithm should handle these cases, in order to provide a proper overview about the core network.

Based on the original assumption about the status code classes, the 407 response code (Proxy Authentication Required) message could even be a CDR closing event. Nevertheless,

Type of CDR	Closing reasons (Response Class)	Exceptions
OPTIONS	2XX, 3XX, 4XX, 5XX, 6XX	403 (Forbidden) 407 (Proxy Auth. Required)
MESSAGE	2XX, 3XX, 4XX, 5XX, 6XX	403 (Forbidden) 407 (Proxy Auth. Required)
REGISTER	2XX, 3XX, 4XX, 5XX, 6XX	401 (Unauthorized)
INFO	2XX, 3XX, 4XX, 5XX, 6XX	403 (Forbidden) 407 (Proxy Auth. Required)
SUBSCRIBE	2XX, 3XX, 4XX, 5XX, 6XX	407 (Proxy Auth. Required)
INVITE	3XX, 4XX, 5XX, 6XX	403 (Forbidden) 407 (Proxy Auth. Required) 488 (Not Acceptable Here) 491 (Request Pending)

Table 3.2: CDR Closing reasons and exceptions

after examining precisely the new request after a 407 Response message, I found that the UAC does not generate new Call-ID and From tag parameter for the retransmitted initial-Invite. If the rule for creating (opening) a new CDR is the arrival of an initial-Invite message, the retransmitted request-response pair of initial-Invites also opens a new CDR. This is not the desired behavior from CDR analysis point of view. We can overcome this issue by associating these messages with the Dialog-ID and using extra information about the 407 responses. In this way, the retransmitted messages can be collected into the same CDR as the original request - (407) response pair. Similarly, "491 Request Pending" also indicates a failed process, but does not mean the end of a SIP dialog. After receiving 491 Response messages, the UAC retransmits the Request message with the same Dialog-ID parameters. Furthermore, the "403 Forbidden" response has 4XX class type status code - but does not complete a dialog in reality, either. The code 403 merely means that the UAS received the previous request, but it refuses to process it. Based on the examination of SIP sessions, I found that the 403 response message does not end the call. It is because this failure code is related only to one request message, and not the whole dialog. However, in REGISTER transactions, after a 403 Response message the UAC can decide to finish the Register procedure, if it seems to be a periodic failure. Last, but not least, the "401 Unauthorized" Response message falls into this exceptional category, as well. This message appears in REGISTER transactions, as an authentication challenge. Although the code itself belongs to an error class, this is a normal operation when trying to establish a call.

The 401 response message contains the RAND parameter for the user authentication process, and triggers a new Register request with the same Call-ID and From tag parameter. When adding the Register transactions into CDRs, the CDR assembling algorithm can handle the 401 status code as a normal part of the authentication procedure, and does not close the CDR. This is the correct behavior, since the upcoming request-response pair for the same call contains the same key parameters. Instead of two CDRs, the four messages could belong to the same CDR. An example of this is depicted by Figure 3.10. This idea has a drawback,

```
From MSISDN = "+367******1"
To MSISDN   = "+367******1"

Call-ID = "4kkz1zz310[redacted]"

Number of stored messages = 4

Message #1:
  Date & time = 2017.01.06 09:00:00.000'213'4
  Link        = <B00<  IMS_CSCF
  Method      = REGISTER
  CSeq number = 3528
  CSeq method = REGISTER
  From URI    = "+367******2@[redacted];user=phone"
  From tag    = "mk1cijkc"
  To URI      = "+367******2@[redacted];user=phone"

Message #2:
  Date & time   = 2017.01.06 09:00:00.132'730'2
  Link          = <B00<  IMS_CSCF
  Status code   = 401  Unauthorized
  Reason phrase = "Unauthorized"
  CSeq number   = 3528
  CSeq method   = REGISTER
  From URI      = "+367******2@[redacted];user=phone"
  From tag      = "mk1cijkc"
  To URI        = "+367******2@[redacted];user=phone"
  To tag        = "mhr0cikr"

Message #3:
  Date & time = 2017.01.06 09:00:00.150'214'2
  Link        = <B00<  IMS_CSCF
  Method      = REGISTER
  CSeq number = 3529
  CSeq method = REGISTER
  From URI    = "+367******2@[redacted];user=phone"
  From tag    = "mk1cijkc"
  To URI      = "+367******2@[redacted];user=phone"

Message #4:
  Date & time   = 2017.01.06 09:00:00.254'134'6
  Link          = <B00<  IMS_CSCF
  Status code   = 200  OK
  Reason phrase = "OK"
  CSeq number   = 3529
  CSeq method   = REGISTER
  From URI      = "+367******2@[redacted];user=phone"
  From tag      = "mk1cijkc"
  To URI        = "+367******2@[redacted];user=phone"
  To tag        = "0xhmyosn"
```

Figure 3.10: Example content of a REGISTER-type CDR: associating many Register messages into one CDR

too. I experienced the following network behavior, when the registration process fails: the UAC sends periodic Register messages, and receives 401 responses. Considering this 401 Response as a non-closing event, the CDR could contain more than four messages, and the statistics do not represent the retransmission as network failure.

SRVCC procedure as a monitoring challenge

Single Radio Voice Call Continuity (SRVCC) is a handover procedure, where the MME provides session continuity from the packet-switched domain (LTE/IMS) to the circuit-switched domain, during an ongoing VoLTE call. According to my monitoring experience at a nationwide operator, currently 2.96% of the VoLTE calls end with SRVCC (see Figure 3.11) due to

resource constraints. Another switching mode is the CS Fallback (CSFB) bridging method [68], which is used to ensure LTE data services combined with 2G/3G voice services. The CSFB capability is developed to change PS technology to CS before the incoming or outgoing voice calls are established.

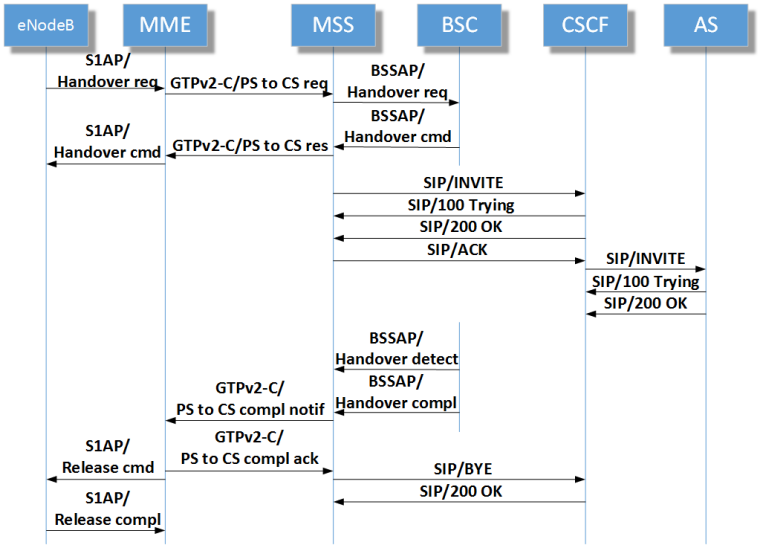

Figure 3.11: Simplified message sequence chart for the SRVCC procedure

To support the domain transfer at system level, an SRVCC enhanced Mobile Switching Center server (MSS) is required in the network architecture. This MSS has a specific GTP protocol-based interface (Sv) to accept control messages from the MME. The Sv interface is used by the MME to require resources for circuit-switched access. The MSS has two main tasks during SRVCC. First, it is responsible for reserving resources from the BSC/RNC (Base Station Controller/Radio Network Controller) through the A/IuCS interface. Second, it is responsible for starting an INVITE session to the IMS on behalf of the UE using a specific address (Session Transfer Number for SRVCC, STN-SR), which is chosen by the MME. The latter procedure is performed by the assistance of the MGCF function, which node translates the traditional call control messages into SIP. After the AS (Application Server) recognizes a call establishment from the MSS, it terminates the original SIP sessions (VoLTE call) in the IMS. When the handover procedure is unsuccessful or the MME canceled the handover requirement, the MSC server also terminates the established call to the CSCF. Monitoring all of the concerned protocols (e.g., S1AP, GTP, BSSAP, RANAP, SIP, Diameter) during an SRVCC procedure is a technical challenge, because the monitoring needs to cover the whole network. Each monitoring point should be synchronized for a precise timestamping mechanism [69]. In a case of encrypted traffic, the deciphering mechanism is also a resource-intensive task [70].

Call Data Record (CDR) assembling is challenging for each LTE/VoLTE specific protocol. The cross-correlation between the interface-specific CDRs needs common historical databases [71] and SIP CDR assembling needs some improvements for SRVCC-cases. As the MSS starts a new SIP call within the already existing call, the Call-ID parameter can be used to collect the messages on the Mg interface into one CDR. The INVITE message triggers another

session between the CSCF and AS. However, this session does not end with the expected BYE transaction; but with a final 200 OK response to the INVITE.

I found that the STN-SR address could be used as a CDR closing reason, which is located in this case in the To field. When the INVITE contains the STNSR address, the 200 OK final response should trigger a CDR closing mechanism. I found that this half-session can be monitored only on the Ma interface, between CSCF and AS. The INVITE session between the MSS and CSCF (Mg interface) ends with a normal BYE transaction.

Validation

This thesis has been formed based on the requirements of the standards, the vendor-specific documentation and the integrated solutions that can be found at network operators. The validation has to be done to decide whether the solutions cover these requirements. This thesis has an implemented version, which is already operational at network operators. The validation has been done based on the success of this implementation.

The first step of the validation was manual testing: whether the solution covers all-day typical cases that are related to the problem domain. Based on these, an automatic integration testing framework has been developed, which is used for continuous integration purposes.

To create a test bed for the validation procedure, I collected manually 60 test samples from real-time control traffic at Magyar Telekom, which is a Hungarian telecommunication service provider. The source of the message sequences could be differentiated into two cases:

1. I created test VoLTE calls in a closed laboratory environment at the telecommunication provider, with custom firmware-based mobile phones,

2. I collected real-life traffic samples from the core network.

The test call scenarios covers the VoLTE - VoLTE, non-VoLTE - VoLTE, and fixed VoIP phone - VoLTE cases. The real-life samples were collected based on randomly selected time intervals.

After the traffic collection step, I selected randomly different SIP Call-ID parameters, which belonged to different transactions and dialogs. Based on the selected Call-IDs, I filtered and collected manually the coherent message sequences into different files. Next to the random selection method, I also searched for special cases to cover the record generation method represented by Thesis 2.1 - 2.3.

I created the following categories from the test samples:

1. BYE or Cancel ended calls,

2. Failure code ended calls,

3. Redirected calls,

4. Failure code ended transactions (within a call session, or not call-related transactions),

5. Subscribe dialogs, including Notify transactions (including failure code ended transactions),

6. MSISDN content in special cases,

7. Authentication-related message sequences (re-authentication events or Register transactions),

8. Extended Service Call Continuity message sequences,

9. Missing messages from the expected sequence (missing opening or closing messages),

10. Retransmission events and special routing cases (e.g., I-CSCF behavior, see Thesis 2.3).

Since there were no reference CDRs for the record generation as expected results, I manually created the expected output content as a reference for each input test files, based on the feedback of the telecommunication operators. Figure 3.12 represents a part from the expected results of a session setup message sequence.

```
=====================================================
TimeStamp:      2014.08.29 10:00:14.640
From MSISDN:    36[redacted]
To MSISDN:      36[redacted]
Call-ID:        bjf20b2g8gfpbpkx8y080qxy0mx9gmy0@ims.telekom.hu
From Tag:       bkg9b8km-CC-107
To Tag:         m2xgyy9b-CC-103
=====================================================
Messages:
        MsgType: INVITE
        TimeStamp: 2014.08.29 10:00:14.640
        LinkID: [redacted]
        IPv4_SrcAddr: [redacted].69
        IPv4_DstAddr: [redacted].70
        IPv4_TTL: 58
        CSEQ: 1
        CSEQ_Method: INVITE
-----------------------------------------------------
        MsgType: 100 Trying
        TimeStamp: 2014.08.29 10:00:14.653
        LinkID: [redacted]
        IPv4_SrcAddr: [redacted].70
        IPv4_DstAddr: [redacted].69
        IPv4_TTL: 64
        CSEQ: 1
        CSEQ_Method: INVITE
-----------------------------------------------------
        MsgType: 180 Ringing
        TimeStamp: 2014.08.29 10:00:14.737
        LinkID: [redacted]
        IPv4_SrcAddr: [redacted].70
        IPv4_DstAddr: [redacted].69
        IPv4_TTL: 64
        CSEQ: 1
        CSEQ_Method: INVITE
-----------------------------------------------------
        MsgType: 200 OK
        TimeStamp: 2014.08.29 10:00:17.584
        LinkID: [redacted]
        IPv4_SrcAddr: [redacted].70
        IPv4_DstAddr: [redacted].69
        IPv4_TTL: 64
        CSEQ: 1
        CSEQ_Method: INVITE
-----------------------------------------------------
```

Figure 3.12: Example content of a manually created reference for a session setup validation test case

After the required output content definition, I applied the implemented version of the thesis on the test input files, and generated the compact record form of each test sample (see Figure 3.13). I manually compared each result one by one with the expected one, and I found that the record collection methods, which is defined by Thesis 2.1 - 2.3, are equivalent with the claim of the telecommunication operators.

```
SIP.CallOpen:
        id: 1
        reason: normal
Common.TimeStamp:
        value: 2014.08.29 10:00:14.640

SIP.TransactionOpen:
        id: 1
        type: INVITE
        callRecordId: 1
        reason: normal
        source: [redacted].69
        destination: [redacted].70
SIP.Phrase:
        type: callId
        data: bjf20b2g8gfpbpkx8y080qxy0mx9gmy0@ims.telekom.hu

SIP.MessageInfo:
        linkName: [redacted]
        method: INVITE
        cseqNum: 1
        cseqMethod: INVITE
        type: request
        statusCode: 0
        messageId: 0
        transactionId: 1
SIP.Phrase:
        type: toURI
        data: 36[redacted]
SIP.Phrase:
        type: fromTag
        data: bkg9b8km-CC-107
SIP.Phrase:
        type: fromURI
        data: 36[redacted]
SIP.UID:
        type: MSISDN
        origin: toField
        inherited: false
        MSISDN: 36[redacted]
SIP.UID:
        type: MSISDN
        origin: fromField
        inherited: false
        MSISDN: 36[redacted]

SIP.MessageInfo:
        linkName: [redacted]
        method: INVITE
        cseqNum: 1
        cseqMethod: INVITE
        type: response
        statusCode: 100

            .
            .
            .
```

Figure 3.13: Part of an example output file content, created from a validation test sample

Naturally, during the operation, new cases for sessions appear due to the nature of the interworking telecommunications equipment both internal and external for the given operator's domain. The thesis coverage for these special cases has been validated manually along the operation and then included in the automatic validation process. The most convincing type of validation is of course when the implementation based on the thesis operates successfully under real-life conditions. This is the actual case for the given thesis, since the CDR compilation works at Magyar Telekom based on this thesis for over three years now.

3.4.3 Thesis 2.2

Thesis 2.2 - I have introduced a unique set of key parameters to cross-correlate the collected Call Data Records between SIP-related logical interfaces. I have identified the protocol fields to recognize the user-related identifiers and label the records with user information. [J3]

Associating transactions and dialogs by taking their interface-ID (link-ID) into account is a basic CDR-generation technique. It is based on the assumption that the communication between two network nodes is direct (through one link). In practice, this is not a proper attitude; although provides the expected result in many cases. In reality, the IMS domain has many functional nodes, and the SIP messages of a dialog are routed within the domain in many times. I investigated the key parameters of the dialog identification procedure, and I explored that special nodes (e.g., AS - Application Server, IN SDP - Intelligent network Service Data Point) often regenerate some dialog-related parameters (e.g., Call-ID, From tag and To tag parameters). These nodes start a new SIP dialog, which continues the original call setup process. I found that the Call-ID is a dynamic information and new key parameters are needed to notice the changes. In order to properly identify a complete call setup procedure - and monitor the routing within a domain or between IMS domains - there are extra information required beside the basic CDR keys (Dialog-ID and IP Addresses).

I defined two record types based on the complexity of the record:

- bCDR (basic Call Data Record) to express the collected information on one interface,
- cCDR (concatenated bCDRs) to represent the whole SIP call (the message sequences through all related interfaces).

time	ip	tt	bCDR ID	cCDR ID	from msisdn	to msisdn	icid
08:40:05.281	10.210.124.2 & 212.51.95.68	invite	5812420914	91347391	+365******1	061******2	828d6c88e
08:40:05.342	10.210.124.2 & 212.51.95.68	invite	5812421042	91347391	+365******1	061******2	828d6c88e
08:40:05.356	192.168.4.20 & 192.168.3.5	invite	5812421078	91347391	+365******1	061******2	828d6c88e
08:40:05.371	192.168.3.5 & 192.168.4.20	invite	5812421107	91347391	+365******1	061******2	828d6c88e
08:40:05.522	192.168.4.19 & 84.1.238.70	invite	5812421463	91347391	+365******1	061******2	828d6c88e
08:40:05.966	192.168.4.20 & 84.1.238.70	invite	5812422591	91347391	+365******1	061******2	828d6c88e

Figure 3.14: Example of a concatenated INVITE CDR from basic CDRs

The cCDR includes all CDRs from the various links, over which the call control messages traversed. In order to concatenate bCDRs into one cCDR, we need some higher-level information, which allows to join the independent call segments properly. During my research, I found that a good candidate for this is the ICID (IMS Charging ID) parameter. This is a unique value in the INVITE-dialog messages, and it is used for gathering the billing information. The ICID value is located in the P-Charging-Vector field, as a parameter. Not every dialog message contains it, but it is mandatory in the initial-Invite request message. However, ICID appears only within the IMS domain, over the trusted IMS links. This means that this value is hidden outside the trusted domain by the edge node. Since the Dialog-ID is the same as in the trusted links, a well-maintained Dialog-ID - ICID database provides great support for cCDR generation. An example of such a concatenated CDR is depicted by Figure 3.14, and Figure 3.15 represents a call setup procedure, segmented into bCDRs.

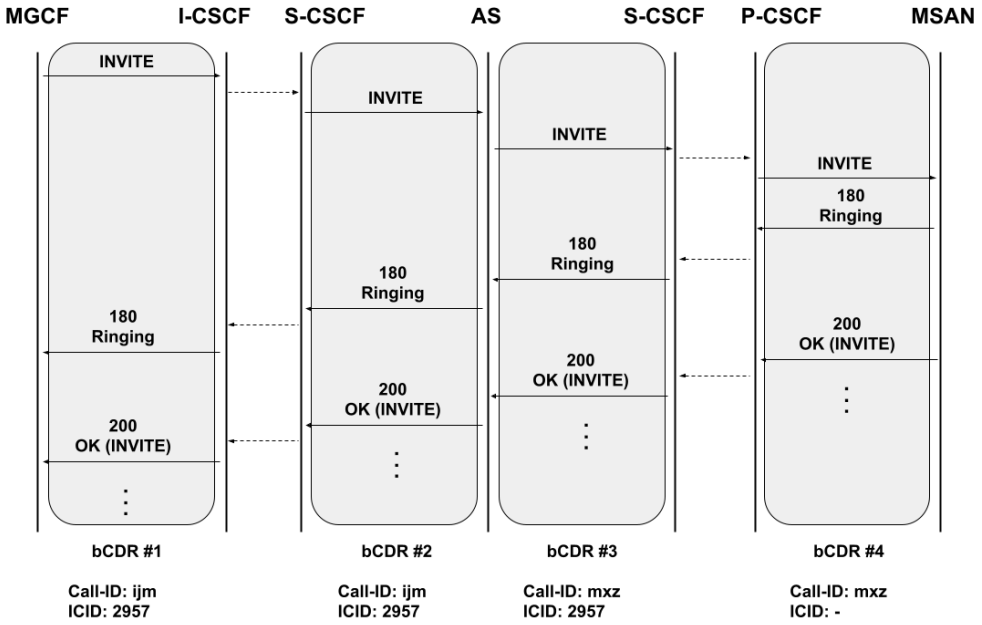

Figure 3.15: ICID-, and Call-ID-based call recognition in IMS. Note, that the two S-CSCFs are the same equipment, but the bCDR's have different Call-IDs.

Figure 3.16 depicts an example for a very simple Call-ID - ICID database, which can be used to concatenate the basic bCDRs into cCDR. For the dialogs not containing any ICID - but have the same Call-ID as another dialog with ICID: we can group those, and assume the same ICID for all. Figure 3.16 demonstrates, how this pairing works with the help of the Call-ID - ICID database. By following the bCDRs shown in Figure 3.15, we can see how they belong together. Let Call-ID #1 be the key of bCDR #1 and #2, and Call-ID #2 be the key of bCDR #3 and #4. Note, that the ICID value is empty in bCDR #4. Since bCDR #3 contains an ICID and a Call-ID as well, we can associate this ICID to bCDR #4 too, since they have the same Call-ID. This procedure can also be applied for bCDRs #1 and #2.

Call-ID #1	ICID #1
Call-ID #2	ICID #1
Call-ID #3	ICID #2
Call-ID #4	ICID #2
Call-ID #5	ICID #2

Figure 3.16: Basic Call-ID - ICID database for Call-CDR collection

There are three important user-related identifiers that support the operator's IMS administration tasks. These are:

- MSISDN - in short: the called/calling party number;
- IMSI - the SIM card's identifier, and
- IMPI - a global, private identifier within IMS - which could also be a concatenation of MSISDN, IMSI, fixed equipment port numbers, and others.

As previously detailed, the order of the fields and parameters can be varied within a message. To collect user-related information, the CDR-assembling algorithm has to search in many fields and parameters. To recognize the caller, the From field could be the starting point. However, I found that in some cases the From field hides the user, and any user identifiers (such as MSISDN, IMSI or IMPI) can be included in the Contact, P-Asserted-Identity, P-Preferred-Identity or in Remote-Party-ID field, in which the format is also variable. Recognizing the callee in a SIP dialog could be a quite complex task. First of all, in most cases the called-MSISDN is located in the Request line, To field or in P-Called-Party-ID field. Beside these fields, the called phone number can be sent digit-by-digit in an INFO transaction, within the INVITE-dialog. Call forwarding is also applicable in IMS: the History-Info field implies this event. An Invite message often includes an SDP or XML layer. SDP is for the media session parameters, and it contains codec-related information. In contrast, the XML part may contain caller and callee MSISDNs (or IMSI or IMPI) and even ICID values. Furthermore, while ICID is a unique identifier within an IMS domain, it is possible to capture different calls with same ICIDs, when the border between IMS domains is monitored. To cover these scenarios as well, the globally unique IOI (Inter Operator Identifier) can be stored beside the ICID. In some cases, a bCDR does not contain the MSISDN numbers, because the IMS hides the user information (e.g., anonymous caller). Using the ICID - Call-ID database, the missing user information could be derived from other bCDRs.

The validation of this thesis is the same as the validation section of Thesis 2.1.

3.4.4 Thesis 2.3

Thesis 2.3 - I have investigated the IP-address key parameter on real-life core network traffic to trace the complete control sequence between two adjacent IMS nodes. I have pointed out that there are cases when the IP-addresses of the session opening and closing transactions are not identical. Accordingly, I have determined that the message collection should be extended to more than one logical interface. [J3]

From the network operator's point of view, a Call Data Record shall contain the complete control sequence, from the setup to the release procedure. There is a practical issue though, which is not usually visible in the logical network architecture diagrams: an IMS domain is usually realized by more than one node - and more than one IP address - in the routing path. On the other hand, using the IP address in the CDR keys raises a question: do we see the end of the dialog always at the same link, as the initial message? To answer this question, we have to examine the functions of the IMS nodes. The problem space can be reduced to routing: how does the IMS look for a called user before eventually finds it?

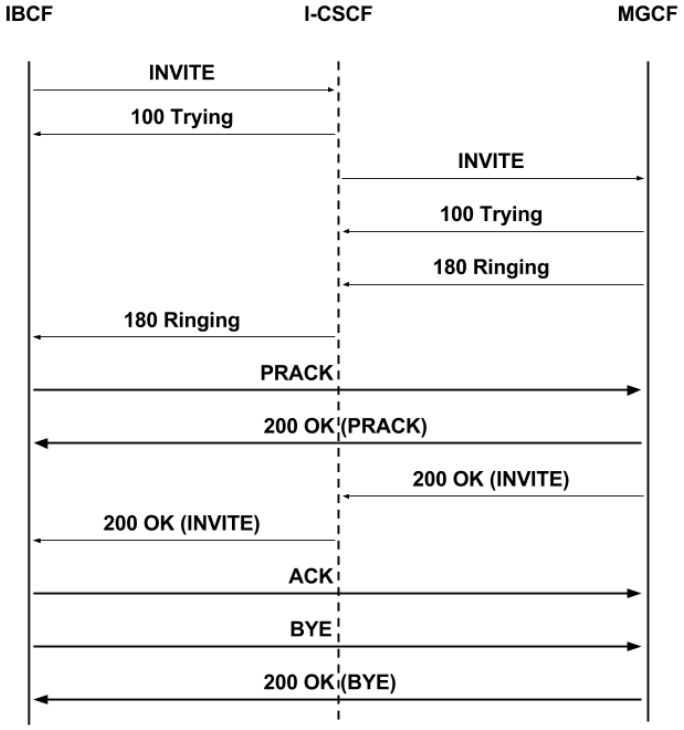

Figure 3.17: I-CSCF operation after receiving a final response message. Note, that I-CSCF disappears from the call-chain when 200 OK arrives.

In general, the entry point to the IMS domain is the P-CSCF module. P-CSCF always forwards the incoming messages to the S-CSCF node. The link between P and S is often non-monitorable if they reside within physically the same equipment. The I-part has a different task: it allows the communication setup between other domains. Unfortunately, after the UAC receives the final response to the initial-Invite message, the I-CSCF leaves the

routing path, hence the edge nodes communicate directly with each other. The result of this property is that the IP addresses of the dialog's release messages are not the same as the IP addresses of the initial message. This hidden property cannot be inferred from RFC-3261, while it must be considered for the session tracking. To assemble the dialog opening and closing transactions, more sampling points (more interfaces) should be analyzed.

Figure 3.17 depicts a typical IBCF - I-CSCF - MGCF communication flow. The initial-Invite message gets forwarded from the IBCF to the I-CSCF, but the call-completion messages (e.g., BYE transaction) are directly sent between IBCF - MGCF, using the same Call-ID and tag parameters, as the first message of the dialog. When the final response to the initial-Invite message arrives with a failure code (e.g., a 487 Response, triggered by a Cancel request), it raises another issue: collecting the special, provisional messages of the dialog. During my research work, I found that those provisional messages (e.g., PRACK, ACK) are routed between IBCF and MGCF, so these are bypassing I-CSCF as well. Figure 3.18 depicts a Cancel procedure, in which the call release messages are routed at the same path as the initial message, except those special, provisional messages.

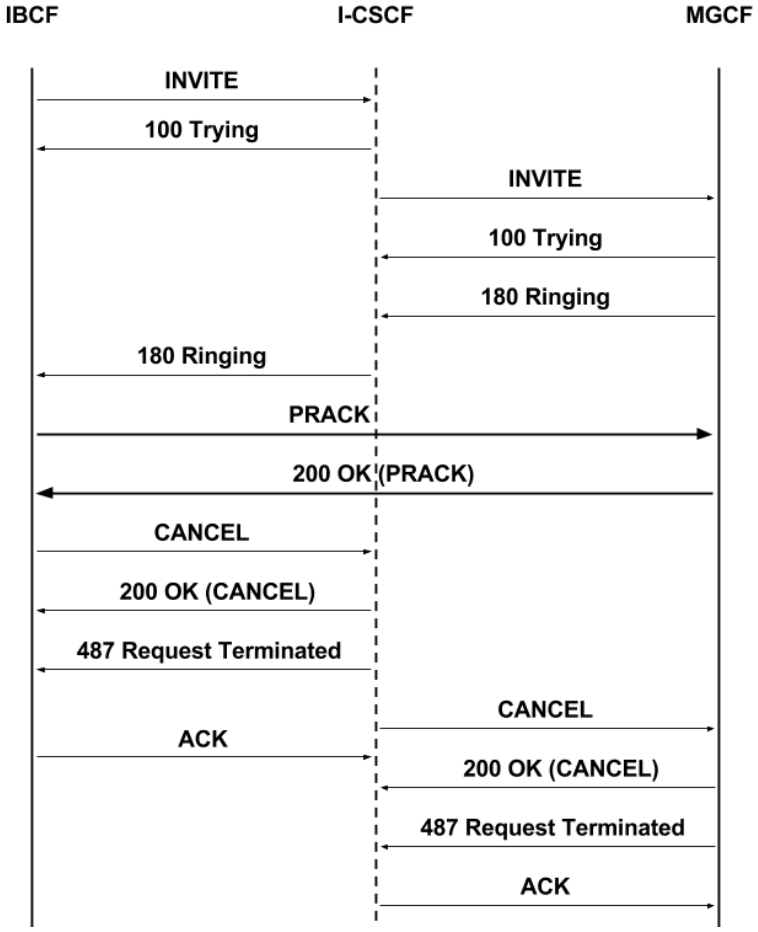

Figure 3.18: I-CSCF message forwarding in a case of CANCEL Request.

The validation of this thesis is the same as the validation section of Thesis 2.1.

3.4.5 Conclusion about SIP protocol-based records and statistical results

The availability of Call Data Records is fundamental for service and network operators because a continuous network assessment is one of the key tools for providing feedback about quality. Regarding VoLTE calls, a passive monitoring-based CDR assembly is crucial for the operators in order to be able to trace what is working well, and what is not. CDRs are also important during a new IMS domain installation procedure, during a resource expansion task, or even while connecting the new generation of telecommunication architecture (e.g., 5G core network elements). To perform proper root cause analysis on VoLTE calls, the record assembly mechanism should handle complex message sequences. The operators need new methods for collect and correlate SIP message sequences between the IMS-related interfaces and to trace the call setup sequences during the routing.

In my research work, I focused on new SIP CDR generation methods, and provided a complete solution for the SIP and IMS node-related challenges. I defined, what messages and parameters can label the records and what are the key parameters for the correlation mechanism. I specified record opening and closing events, record types, and identifiers for the message assembly mechanism. In contrast of the RFC-offered Dialog-ID, I found an extra parameter (IMS Charing ID) to trace the whole call setup procedure.

One of the suggestions is to compose basic (link-level) CDRs into more general CDRs, to allow the operator to easily see the bigger picture, e.g., calls together with registrations. The starting point of the CDR assembling algorithm is the dialog recognition process of the IMS nodes: Dialog-ID-based identification. Since the tag parameter of the To field is not fixed until the final response message, it could be skipped from the CDR keyset. Network monitoring results show that without the To tag parameter, the Call-ID, From tag pair is also unique, and the extension with IP addresses gives a strong identification key.

CDR closing reasons provide useful information for the operators while analyzing the CDRs. This information should be as precise as possible, otherwise it would be misleading. The monitoring system should take into account the mentioned special features and exception-handling mechanisms of the IMS systems and the SIP protocol, otherwise operators would get false status and statistical data about their VoLTE service. The record closing algorithm has to use extra logic for the CDR closing events for providing proper statistics. It is clear that not every 4XX-type response releases a dialog or a transaction, and the algorithm should handle these exceptions (e.g., 403-Forbidden, 408-Not Acceptable Here, 491-Request Pending). In a case of REGISTER-type records, the authentication procedure (e.g., 401-Unauthorized) could be also concatenated into the same CDR. It is important to distinguish cases of rejects during registration, and cases suggesting network error: this helps in proper classification of 4XX-type responses.

Another suggestion is that the monitoring system should be equipped with methods of handling and cross-correlating key parameters (IMSI, MSISDN, IMPI, ICID) on-the-fly. Such an up-to-date database with multi-parameter indices can also help in building further, monitoring-based services, as well. Taking into account the methods and corrective actions for CDR-assembling lead to recognizing the really unusual call scenarios and even to recognizing downgraded quality within the VoLTE service.

In order to show a higher-level abstraction of the SIP calls, the link (IP address) related

CDRs can be collected into one call record, using billing information as the global user-activity identifier (ICID). A Call-ID - ICID database is a strong assignment for higher-level CDRs. In case the CDRs related to the same call would not get connected, statistics about current or overall number of call can be falsely interpreted, too.

The algorithm should support the I-CSCF behavior, and handle the half dialogs, or even just a PRACK transaction in a case of a CANCEL procedure. Using Call-ID - ICID database, the opening and closing of the insufficient records can be summarized into the normal-operation statistics, instead of indicating a failure event. Although the above-described scenarios are seemingly easy-to-cover by software, the distributed manner of user-related identifiers makes them harder to handle.

If the specialties of the IMS architecture and the SIP protocol are handled by the monitoring system, its measurement result can be considered as a reference. Any significant deviation from that reference would indicate potential problems. When the operator trusts the result of the monitoring system, its personnel is kept interested in clearing out all the really erroneous situations that such a system points out.

Statistical results

In order to handle all sorts of exceptions, extra logic has to be added to the CDR assembling mechanism. A complex logic, which handles various (sometimes very peculiar) exceptions, does have a significant drawback. Experts must put in greater and greater efforts into figuring out solutions for those exceptions that have smaller and smaller probability to occur. While this gives a technically perfect result for the CDR assembler, it may not be economical, given the impact of the network monitoring system and its overall purpose. For the monitoring system, it is the operator's management decision whether further exceptions are worth covering, when e.g., over 99% of the cases are handled properly.

Based on a 5-day long monitoring period at a major Hungarian network operator, when over 60 million INVITE CDRs and over 530 million REGISTER transactions were captured, I got the following, rough proportions of exceptional cases:

- 20% of the final responses in the INVITE CDRs were 407 Proxy Authentication Required;
- <0.8% of the answers to Invite messages were 403 Forbidden, <0.1% were 488 Not Acceptable Here, and <0.02% were 491 Request Pending responses;
- 10% of the bCDRs would have handled incorrectly without taking into account the behavior of I-SCSF;
- 1:6 is the proportion of the cases when the caller releases the call before it gets fully established (Cancel);
- 1:2 is the ratio of unsuccessful (4XX) and successful (2XX) response codes (see the distribution below);
- 1:4 is the ratio of unsuccessful and successful response codes when not counting the 487 response code (which should be accounted as normal for canceled calls);

- 2:5 is the proportion of registrations requiring authentication out of all the registration requests.

The distribution of INVITE dialog response codes in this period were the following:

- 1XX: 66.0%,
- 2XX: 21.2%,
- 3XX: 0.13%,
- 4XX: 12.0%,
- 5XX: 0.45%,
- 6XX: 0.20%.

Table 3.3 compares the results of Thesis 2.1 - 2.3 to the RFC-defined session identification method.

Event	Previous methods	Thesis 2.1 - 2.3
Not justified record closing	21%	0%
The success of session assembling on one interface	83%	100%
Not recognized session opening/closing transaction based on the IP addresses	10%	0%
Proportion of 4XX and 2XX response classes	1:2	1:4

Table 3.3: Statistical results of Thesis 2.1 - 2.3

The architecture of a telecommunication core network (and the operated network functions) often changes. Newer versions of the standards are released frequently, which result in new message parameters, new contents and even new message sequences. These changes could result in previously unknown or undefined traffic cases. The results based on Thesis 2.1 - 2.3 represent a momentary state of a Voice over LTE service.

The CDR generation method for the 5-day long measurement was based on Thesis 2.1 - 2.3. This measurement showed that indeed, not covering the "exceptional" cases within CDR generation (see Thesis 2.1 - 2.3) would significantly mislead the operators when analyzing the CDRs without such detailed knowledge of the complex and exceptional cases.

Chapter 4

Service quality analysis of IP-based voice services

4.1 Introduction to voice quality inspection

Looking back on recent years, voice-based mobile communication evolved radically, i.e., Voice over LTE (VoLTE) and Voice over IP (VoIP) opened the way towards high definition real-time voice services. This technological evolution raised users' expectations against interactive voice services. Transmitting live voice data over IP networks, which became a popular service of the near past (VoLTE/VoIP mobile calls, Viber, Skype, etc.), defines strict real-time criteria, including the control of end-to-end delay, delay variation and packet reordering.

The real-time voice packets travel as user data through the VoLTE network, which is often a higher throughput capable interface than the interfaces for the control traffic. The voice quality (or real-time user data) analysis and failure detection is another type of management tasks. To get a high-level overview about the packet-level QoS (e.g., in a predefined tracking area) is an exciting and challenging area. It can be a very powerful tool in the operator's hand to detect network anomalies or even predict the failures. To get a high-level overview, the operators require a detailed analysis of each call, which is a resource-intensive task in the data plane. User privacy and a requirement of a continuous, no-reference measurement environment further increase the complexity factor.

This chapter presents novel methods for the network and service operators' tasks. It introduces a novel packet-level measurement method (including a novel metric set) to measure network and service quality online. It is done without inspecting the voice payload and needing the reference voice source.

The current chapter contributes the following novelties:

- It introduces new performance metrics that enable to measure and describe the time-domain behavior of the service from the viewpoint of the voice application.

- Based on the proposed metrics, it introduces a no-reference voice quality estimation model.

- Additionally, this chapter proposes a new method to identify the pace of the speech (dynamic or slow) as long as voice activity detection (VAD) is present between the endpoints. This identification supports the introduced quality model to estimate the perceptive quality with higher accuracy. The performance of the proposed model is validated against a full reference voice quality estimation model called AQuA, using real VoIP traffic (generated from various voice samples) in controlled network scenarios.

Chapter 4 is organized as follows. Chapter 4.2 provides a theoretical overview covering the service quality estimation models and Chapter 4.3 presents a technical recap for the main material of Chapter 4, focusing on the RTP (Real-Time Protocol) specialities. Finally, Chapter 4.4 introduces the new results based on the specific challenges.

4.2 Theoretical background for service quality analysis

To analyze the performance of the data plane, the application-level packetized media and the jitter buffer properties should be also considered. This viewpoint needs specific models, metrics and performance indicators to estimate perceptive service quality.

4.2.1 IP Performance Measurement

Since the VoLTE service is based on IP network components, the aim of a first level analysis should be based on packet-level metrics. The IETF working group [72] distinguishes the following essential impairment indicators:

- one-way packet loss [73],
- one-way delay [74],
- delay variation [75] and
- packet reordering [76].

Based on [77], the performance metrics could be differentiated as active, passive or hybrid methods. An active method depends on a dedicated traffic. The packet stream of interest has field values that are defined for the measurement.

Since active methods are often applied on multiple measurement points [78], the sequence number combined with a precise timestamp is a general input information. Generally, the source and the destination of the flow are also predefined. The measurement point could imply a logical location (e.g., a telecommunication interface) or a physical location (e.g., the external link of a network node) where the observation is performed.

In contrast to the active methods, the passive method operates solely on the observation of an unmodified packet flow. The passive measurement must not change the content of the packets and the protocol field values. Passive methods operate on an existing network traffic to observe and collect information, and often operate on more logical observation points. Protocol filters often applied on the observation points to prefilter the dedicated streams. The communication for a general information collector module is a basic aspect of passive measurement. The passive performance metrics are applied separated from the packet stream.

The combination of the active and passive methods are called as hybrid metrics. The first type of hybrid methods generates packet stream and also observes the network load passively. The second type of hybrid methods applies active and passive measurements in parallel and analyzes the characteristics and the differences.

Packet loss

Packet loss is an essential network impairment indicator [73] which implies the current status of the network. To understand and recognize packet loss is profitable from the network operators' viewpoint, because loss events could eventuate that the applications or services do not perform well. When the packet loss is relatively large to some threshold value, it

is more difficult for transport-layer protocols to deliver control or user-related information (e.g., the transport-layer protocol tries periodically to re-send the missing information, which results bandwidth degradation). Since the real-time applications and services are sensitive to excessive packet loss, it could result quality degradation in voice communication or failure events during a call setup procedure.

Measuring one-way loss instead of round-trip loss is more advantageous from the perspective of the core network: the network architecture is distributed and there could be several alternative paths for the control traffic. The route could be asymmetric from the source node to the destination node and backward. Besides that, quality of service guarantees may radically differ in the reverse direction.

The RFC 7680 [73] defined loss metric parameters are: source IP address, destination IP address, time (T) and a threshold as a waiting time. Loss event occurs, when a packet is sent from the source to the destination at wire time T, and the destination did not receive the packet within the waiting time. It should be noted that packet loss should be differentiated from large network delay. The theoretical upper bound of the waiting time should be calculated from the packet lifetimes of the current service, or even could be 255-second as the upper bound of IP packet lifetime [79].

When the packet is received by the destination, but the payload or the header information is corrupted, it could be also considered as loss from the perspective of the applications. Another type of loss is the incorrect reassembly of fragmented information.

Based on [73], loss distance and loss period metrics [80] could be derived from packet loss events. Bursty packet loss (or consecutive packet loss of a given stream) is a loss pattern measured by the loss period metric, which network behavior can degrade the quality of the provided services. To determine the distribution of the loss events is a key parameter for performance assessment. Different loss distribution could result different user observations based on the type of the distribution (bursty or consistent).

Packet delay and packet delay variation

The end-to-end delay could be also a service quality degradation factor. Large delay results similar use-cases than packet loss. Providing real-time services is impossible in erratic networks. The large value of delay could result packet loss from the application's viewpoint, when the delay exceeds the waiting time threshold. It is also recommended to measure one-way delay in core networks instead of round-trip delay, which can be also traced back to asymmetric network routes, as [73] defines. The one-way delay metric, introduced in [74] is defined as follows. The metric parameters are: source IP address, destination IP address, time (T) and loss threshold waiting time. When a source sends a packet to a destination at wire time T, the destination receives the packet at $T + d_T$, where d_T is the one-way delay metric.

Analyzing predefined message sequences or packet streams, the simple delay metric could be used to derive another metric: delay variation [75]. The delay variation metric is the difference of the packet's one-way delay. This metric could be used to define the queue sizes in network nodes, or the buffer size of the real-time applications (e.g., de-jitter buffer [81] [82] size for voice). Delay variation has two industry-specific categories [81]: Inter-Packet Delay Variation (IPDV) and Packet Delay Variation (PDV). The difference between IPDV

and PDV is the reference for the delta calculation. In the IPDV formula, the reference is the previous packet in the message sequence. Calculating the PDV metric, the criteria for the reference is predefined (e.g., the minimum delay in the measurement interval). Next to the buffer occupancy calculation, delay variation could refer to frequent path changes, frequent loss and load balancing problems in a core network context.

Packet reordering

Since IP-based networks do not ensure ordered packet delivery, every packet contains a unique sequence number (e.g., a randomly initialized counter value). This unique identifier is used to sort the received packets at the destination side. To measure the orderliness, the packet reorder metric [76] is a basic packet-by-packet characterization method.

The packet reordering metric is purely based on the sequence numbers, which is used to analyze the arrived sequence. For example, if we consider the 3,4,5,7,8,6 sequence, then packet 6 is out-of-order. We could also classify packet 6 as late packet or reordered packet. From real-time applications' viewpoint (e.g., VoLTE services and voice calls), the packet reordering is a relevant feedback. A late-arriving packet could be dropped, since it loses its relevance at application-level. Reorder events could be traced back to retransmission, buffer queuing algorithms, multi-processor served network interfaces or load balancing.

The basic reorder metric is calculated from sequence number (s) and next expected sequence number (NextExp) for a given stream, where the stream is identified by the source and destination IP addresses. The packet is reordered, if $s < NextExp$. When $s \geq NextExp$, the packet is in order. However, the case of $s > NextExp$ should be also differentiated, which is called as sequence discontinuity, based on [76]. The $s - NextExp$ difference defines the size of the discontinuity, which is more precisely the number of missing packets. The RFC 4737 [76] also defines the late time offset metric for reordered packets, which equals to: $dstTime(i) - dstTime(j)$, where $dstTime(i)$ is the time when the i^{th} packet arrives at the destination, and index j represents the discontinuity associated to i. The $dstTime(j)$ value reflects the expected arrival time. De-jitter buffers [82] are used in real-time applications to smooth out the reordering events, and sort the arrived packets before the payload processing phase. Calculating the late time metric determines that the arrived packet is still useful from the perspective of the application.

4.2.2 Voice quality assessment standards

Mean Opinion Score (MOS) [83] [84] is a standardized scale for communication quality estimation. The value is calculated based on different network models and protocol parameters in several conversational and application-specific situations. [85] defines an objective measurement model, and [86] introduces a 5-point scale for subjective results. The MOS score is a generally used opinion value, which is assigned to a call conversation, based on a predefined scale. The scope of the audio MOS calculation methods could be categorized into three categories:

- Listening quality,
- Conversational quality, and
- Talking quality.

Each category could be further splitted into subjective, objective or estimated cases.

Predicting the listening-quality is a method for listening-only situations. The measurement could be performed at electrical or at acoustical interfaces. In a case of electrical interface-based scenarios, there is a sealed condition between the handset receiver and the user's ear, while in the acoustical case there is a leaky condition. In the latter case the measured MOS value is maybe degraded during the test scenarios. The talking quality of a voice call describes the perceived quality at the talking party side. To estimate the aspect of the talking party, the methods take echo signal, background noise switching and double talk into consideration.

The MOS scores could be also applied to estimate the call quality in conversational scenarios using e.g., the E-model [87]. The E-model is a transmission planning and rating method which calculates a transmission quality value from impairment factors. The model has several versions, which applies the combination of the following parameters:

- room noise,
- weighted echo path loss,
- round-trip delay,
- equipment impairment factor,
- circuit noise,
- quantizing distortion,
- talker echo loudness rating,
- random packet loss based on codec type,
- low talker sidetone levels,
- packet loss probability,

- delay impairments,
- delay sensitivity parameters, and
- system interactivity requirements (e.g., low or very low).

The E-model results a scalar quality rating value, the so-called R value, which can be transformed into the MOS scale as an estimated customer opinion.

There are several standardized test methodologies to measure perceptual quality degradation in a voice material. One of the well-known algorithms is the Perceptual Evaluation of Speech Quality (PESQ) [88], which is standardized as ITU-T P.862 recommendation. The model could be used to measure the performance of narrow-band speech codecs, or end-to-end communication. PESQ does not include delay, sidetone, echo and other impairments related to a two-way communication, which could result a high PESQ score value combined with a poor connection quality. PESQ uses e.g., audio signals, transcodings, bit rates, environmental noise as input parameters. This model applies a full reference (FR) solution (see Figure 4.1), which means that the algorithm compares the original speech signal with the degraded waveform (PESQ uses 64 ms frames for the cross-correlation). The original signal is passed through the communication system, and the output is recorded as the degraded form. The PESQ method compares the delay, silence and speech intervals as the first step, and also compares the audio signals. It computes two error parameters, which is used to calculate an objective listening quality MOS score (the output range will be between 1.0 and 4.5).

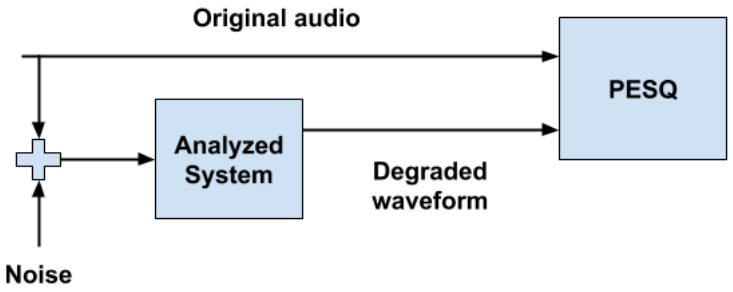

Figure 4.1: Reference-based voice quality estimation model using PESQ [88]

Since PESQ is primarily focused on the quality impact of codecs, there was a claim to create a method, which rather focuses on network impairments. Perceptual Objective Listening Quality Analysis (POLQA) [89] is a newer standard for objective voice quality analysis. The method measures one-way speech distortion and noise but also ignores delay, sidetone, echo and other impairments of a two-way communication. POLQA is also a full reference-based voice quality assessment method, which supports narrowband, wideband and super-wideband signal bandwidth. The algorithm splits the reference and degraded signals into very small time frames and produces a set of delays. These delays are used to re-determine the estimated sampling rate. As the next step, the algorithm transforms the reference and the degraded signal into an internal representation based on perceptual frequency and loudness. POLQA also eliminates the low levels of noise and analyzes the change of the signal characteristics.

As a result, it computes five indicators: frequency response indicator, noise indicator, and three other indicators which describe the difference from the internal representation. The combination of the five indicators estimates an objective listening quality MOS value (the output range will be between 1.0 and 4.75).

AQuA [90] is an industrial, easy-to-use voice quality tester from Sevana. It also implements an FR model, which provides quality analysis between audio files. The software calculates an estimated MOS value (the output range will be between 1.0 and 5.0), as a classic indication of perception, and a quality percentage value, both derived from the waveform. The latter, i.e., the quality percentage calculation procedure, is based on the analysis of spectrum vibration, energy distortion and other waveform-specific parameters. AQuA uses a different perceptual model than PESQ to reveal more information about the loss of voice quality. This is because there are cases of degradation that PESQ fails to detect (see Table 4.1).

Feature	PESQ	POLQA	AQuA
Measurements in wideband audio	No	Yes	Yes
VAD detection in AMR coded	No	Yes	Yes
Variable delay up to 1 sec in VoIP	No	Yes	Yes
Works in SWB (above 16 kHz sample rate) bandwidth	No	Yes	Yes
Works with audio sampling rate higher than 48 kHz	No	No	Yes
MOS score for long sequences of speech (more than 30 seconds)	No	No	Yes
Language independent	Yes	No	Yes
Multi-dimensional audio (works with stereo)	No	No	Yes
Able of working with long audio (1 hour and longer)	No	No	Yes

Table 4.1: A comparative summary: PESQ, POLQA and AQUA [90]

Since the standards and previous solutions are mostly FR designs, applying it on real-time user traffic is not relevant in a real-life situation. The standards could be used as an ad-hoc, predefined measurement to get the current status of the network, but a continuous FR management tool could not be applied because of user privacy issues. No-Reference models could be an appropriate choice, but the packet-level metrics, which are used as input parameters are essential counters, and do not separate well the reorder/loss concept at application-level. There is a need for a new packet-level metric calculation method, which emulates the buffer occupancy to recognize the loss events from the application's viewpoint.

4.2.3 State-of-the-art analysis of service quality estimation models

As previously mentioned in subsection 4.2.2, a widely used standardized opinion model for subjective voice quality estimation is the E-model [91]. Several papers examined the effectiveness of the E-model for voice quality estimation, and found, that it is not accurate for estimating subjective quality [92]. It underestimates the voice quality in the case of transmission delay [93] and does not take the burst packet loss into account [94].

A. Takahashi et al. [92] proposed a modified E-model for quality estimation. The authors found relationship between network delay and speech distortion. The paper also investigates the problem space of terminals' loss rate. Since the jitter buffer implementation can be different in each terminal type, the packet loss rate can be also different and a calibration file (describing the characteristics of the jitter buffer) should be used for each type [95].

A. Raake et al. [93] presented an improved E-model and extended the original parameters with two new inputs: random loss and burst ratio of the loss. Authors also introduced two packet loss behavioral models: macroscopic loss (speech quality changes over time) and microscopic loss (the effect of packet loss at the decoder side). The prediction model from the average packet loss has been adopted in the E-model.

A. Takahashi et al. [20] proposed an E-model-based method that describes the relationship between delay effects and speech distortion, when they occur at the same time. The authors estimated a subjective MOS from the R-value by transformation. The new MOS estimation model uses the one-way delay, echo-path loudness rating, noise level, and equipment impairment factor as input parameters. The proposal assumes the speech coding technology and the packet loss rate to be previously known.

Y. Jung and C. Manzano [94] presented a study about the effects of burst packet loss. The authors also found that the Emodel does not calculate with the burst packet loss. Accordingly, the proposal defines novel burst-related metrics: burst duration, burst density, gap duration and gap density. Their experimental results showed that the PESQ scores correlate with the above loss parameters. The authors defined multiple loss-range categories for MOS estimation and found that under 7 percent of packet loss, there is no any benefit, in terms of accuracy, to include the burst in the estimation formula.

S. R. Broom [96] found that the system characteristics should be calculated as a degradation value for MOS estimation. The paper concludes that the network loss and jitter values are insufficient for estimating the voice quality. The author introduced a calibration method to measure the characteristics of the VoIP equipment. Using this device specific parameter, a higher correlation rate is achievable.

Y. Ouyang et al. [97] proposed an Android App implementation for VoLTE quality estimation. The system examines the wireless network part of the communication path and uses the POLQA standard for initial calibration. The applied model uses two phases: a training phase and a testing phase. In the training phase, the mobile phone is directly connected to a box to measure MOS values based on reference voice signals. In the testing phase, the client runs in the background and periodically sends network indicators to a database server for further analysis.

D. Luksa et al. [98] showed that the choice of the codec (G.711, GSM, Speex, iLBC 20, iLBC 30, and G.729) has no impact in the intelligibility of voice performance and does not influence the objective voice quality.

W. Zou et al. [99] proposed a machine learning algorithm for voice quality estimation. The authors used a random forest-based training and assessment method that applies 9 input parameters for the assessment (e.g., UDP length, bitrate, and average packet loss). PESQ scores were used as a reference for the analysis of 2,400 degraded voice samples. Test results showed that the correlation between the proposed model and PESQ is higher than between PESQ and the E-model.

Y. Han and G. Muntean [100] present a hybrid call quality assessment model, namely HCQ. The authors collected the benefits of the intrusive (e.g., POLQA, PESQ) and non-intrusive (e.g., E-model) methods and combined them into one solution. The system calculates two MOS values, which are invoked in the final calculation phase. One is the result of the online model and the other one is estimated by an offline reference-based model. The offline MOS calculation is based on small voice segments, where the system records a few seconds and estimates its quality with PESQ. The drawback of this solution is that the recorded voice is sent through the network for analysis, which is not a viable option in many scenarios. Authors recommend a reduced reference method for the offline calculation.

N. Majed et al. [101] analyzed delay-based measurement methods in VoLTE environment. The authors found that 3GPP test methods (e.g., IPDV) do not model the behavior of jitter buffers in real VoLTE systems and they presented a set of improvements for a higher correlation result.

M. Abareghi et al. [102] proposed an improved ITU-T standard, namely P.563 [103]. This standard incorporates a nonintrusive model for voice quality assessment, but it is not appropriate for VoIP calls. The authors presented a new distortion class for proper network condition detection.

D. E. Conway [104] proposed a passive measurement method that incorporates the offline quality estimation model called PESQ and a media payload injection technique to replace the original user media data with a test media source. Using payload replacement, the author avoided any privacy issues raised from decoding the user's media payload. However, the proposed method does not reflect to the properties of the application's playout buffer and therefore the real data loss present at the input of the voice decoder is not incorporated in the result of the quality assessment.

L. Sun and E. C. Ifeachor [105] presented a new passive (non-intrusive) methodology for developing accurate models to predict voice quality for IP-based voice services. Based on the methodology, authors introduced regression models for predicting conversational voice quality for four common voice codecs: G.729, G.723.1, AMR, and iLBC. By using real VoIP traces, authors showed that the prediction accuracy of the generated models is close to the one of the combined ITU PESQ/E-model method.

J. C. w. Lin et al. [106] introduced a new parametrical neural network-based model to estimate the voice quality in Voice over IP systems. The work enhances the model presented in [105]. The main contribution of their work is the statistics-based packet loss evaluation. The proposed method is more efficient computationally since it does not require Markov model mapping.

It should be noted that both [105] and [106] lack considering the effect of packet reordering and packet losses derived from the overrun of the playout buffer (i.e., loss due to a traffic burst).

4.3 Technical background for data plane analysis

Section 3.3 implies a detailed overview about the Voice over LTE architecture (the LTE architecture and the IMS domain), including the signalling interfaces and monitoring points. The current section completes Section 3.3 with the RTP protocol specialities, to get a whole overview about the signaling- and user plane.

4.3.1 RTP protocol

Transmitting live voice data over IP networks, which became a popular service of the near past (e.g., VoLTE-based mobile calls), defines strict real-time criteria, including the control of end-to-end delay and packet reordering. The VoLTE media services operate over the Real-Time Transport Protocol (RTP) [107], which is an UDP-based protocol carrying the media flow end-to-end. RTP and SIP supplement each other in the VoLTE service: SIP is used for the control part and RTP is used for the media (e.g., voice, video) delivery part of a multimedia session.

RTP operates with a small, fixed-size, compact header, and codec-defined payload information. It carries compressed audio data in small chunks (e.g., a 20 ms voice chunk). The header also defines the type of the encoded audio part, which is included in every packet. The source of the multimedia stream is identified by the Synchronization Source (SSRC) ID, which is unique for each VoLTE audio or video source. The receiver side uses the SSRC to group the RTP packets and reconstruct the audio or video flow based on the individual timing and sequence number information.

Since RTP is not a connection-oriented protocol, it does not use methods to ensure timely delivery. It uses sequence numbering and timestamping to relay the time-sensitive media frames over packet-switched networks. The main feature of the protocol is the ability to detect packet losses and to recover orderliness and timing at the receiver side. The latter feature is based on a special timestamp format derived from the voice codec's internal sampling frequency. When a sequence numbered RTP packet arrives at the receiver far from its expected position, the encapsulated media data loses its relevance in time, and thus the RTP payload will be ignored. Figure 4.2 represents a simple reordering example, where packet $p2$ arrives later than it is expected. In order to maintain low end-to-end delay, RTP operates with a small-sized application-level receiver buffer (e.g., 100 ms), namely a playout or (de-) jitter buffer, to smooth out delay variation and perform reordering, when needed. The recovery processes (jitter elimination and packet reordering) are limited by the effective size of this buffer.

While the sequence number is incremented by one for each packet and used to detect loss or reordering, the timestamp carries codec-related temporal information, and is used to schedule the arrived packets in time. For voice services, the timestamp in the RTP header is incremented in every packet by the number of audio samples it carries. Typically, real-time voice codecs release 10-30 ms voice frames at their outputs. 20 ms frame size is the most common choice for VoLTE and VoIP codecs such as Adaptive Multi-Rate Wideband (AMR-WB) [108], Opus [109] and Speex [110]. Accordingly, the average extent of the jitter buffer is usually in the range of 60-140 milliseconds, which provides temporal storage for a few consecutive media frames only. For example, when a packet contains 20 ms of audio

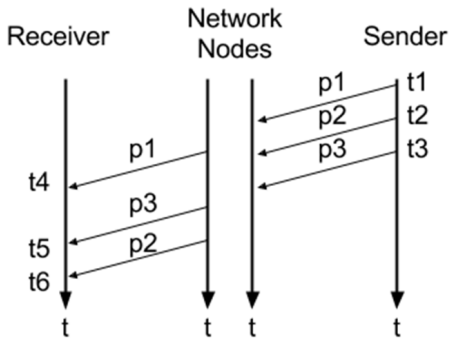

Figure 4.2: Packet Reordering

sampled at 8 kHz, the timestamp is incremented by 160 in each RTP packet. The initial value of the sequence number is random within the 16-bit range.

4.4 New results in service quality assessment

Thesis group 3 - I have provided management methods for the user plane of the VoLTE ecosystems. To assess the data plane, I have defined a new packet-level metric set for IP-based voice services. I have introduced a new method to recognize packet loss events and created two new application-specific loss categories for the arrived packets. I have shown that the calculated loss metrics could be used to derive a new quality model that correlates with the result of a reference-based MOS estimation model. [J1] [C1]

4.4.1 Challenges about voice quality assessment

The growing scale of VoLTE deployments as well as the widespread of Internet voice services increase the demand for their user-centric service quality monitoring. The primary question in this domain is how a telecommunication provider can monitor the quality of user experience without executing subjective assessments or applying the common full reference objective quality models. While the former is time and resource-consuming and typically performed ad-hoc, the latter requires a reference source, and by processing the voice data it may offend user privacy. The VoLTE services operate over the Real-Time Transport Protocol (RTP) [107], which is a UDP-based protocol carrying the media flow end-to-end.

The available service quality measurement models are typically:

- packet-level QoS-based,
- objective QoE estimation-based, or
- subjective evaluations from volunteer users.

Existing voice quality assessment methods incorporate a single model or a combination of the models above.

A continuous voice traffic monitoring reveals basic transmission impairments, which does not give a result that is based on the viewpoint of the voice application. Packet-level analysis of QoS parameters omits the presence and the effect of application-level media buffering (i.e., jitter buffer) that directly influences the perceptive call quality. Alternatively, a provider can analyze the voice waveform using a reference-based objective quality model (e.g., PESQ [111], POLQA [89]). The main drawbacks of the latter algorithms are:

1. offline operation (i.e., post-processing),
2. requirement for a reference source,
3. processing user data (i.e., decoding the voice conversation for quality analysis), which is a fundamental privacy issue.

However, constraint 3. can be eliminated by executing active measurement sessions, i.e., a group of mobile terminals can perform periodic or ad-hoc test calls over the network.

Real-time traffic enables using offline post-processing methods only with strict limitations, or with extended algorithms. The latter require extra resource allocation for pre-buffering and delayed computing. On the other hand, an operator can collect ad-hoc feedbacks about the service quality by applying subjective assessments. This direct method gives a realistic description of the user experience at a given time. However, it requires unreasonable effort from both sides, i.e., from the customer as well as the provider.

The service operators need new traffic measurement methods for RTP-based real-time flows, to measure the transmission properties from the application's perspective. To grant user privacy is an essential and challenging task, the new assessment methods should analyze the traffic without inspecting the voice payload (i.e., the private media data) and needing for the corresponding reference voice material.

It is a common technique in service management to use traffic analysis for estimating the transmission quality at the receiver side. Typically, packet-level QoS parameters are measured on the basis of the corresponding IETF RFC proposals [112] [113] [114] [115] [116] [117]. A common property of the RFC-based methods is that they separate the reorder and loss concepts. In the case of real-time media transmission, it may have several benefits to handle these parameters side by side.

Recognizing and differentiating reorder events from packet loss is not a trivial task. From the application's viewpoint, both packet-level reordering and packet loss have various cases and each of them has a unique impact-level on the perceptive quality. Additionally, the state of the buffer that is available for the RTP flows to eliminate reordering is unknown by the independent assessment tools.

Considering the common QoS-based quality analysis models, the lack of application-level performance indicators (e.g., a buffer-state descriptor) and the separation of the reorder and loss concepts together do not enable to estimate perceptive quality at a high confidence level in many transmission scenarios. When a sequence numbered RTP packet arrives to the receiver far from its expected position, the encapsulated media data loses its relevance in time, and thus the RTP payload will be ignored. If we follow this thought, only a stateful algorithm could squarely classify a packet as reordered or lost.

The identification, differentiation and calculation of reorder and loss events is a challenging task in itself. However, these performance indicators can signal network failures, operators need an easy-to-understand quality index for each multimedia session. Mean Opinion Score (MOS) index is a generally accepted quality indicator, which generation from pure packet delay information is also a challenging task.

4.4.2 Thesis 3.1

Thesis 3.1 - I have provided a new method to calculate packet loss metrics on real-time IP-based voice traffic. I have defined two loss categories for the arrived packets from the applications' perspective. The method is able to operate in real-time, without reference waveform and user privacy violation. [J1] [C1]

To measure the characteristics of real-time flows from the application's viewpoint, we have to examine the operation of the RTP protocol (see section 4.3.1). As mentioned in section 4.3.1, RTP uses a sequence number for packet orderliness, and also uses a timestamp to represent the playout time at the receiver side. The timestamp is incremented by a codec-defined value, which is counted from the sampling rate and audio chunk size (e.g., the incrementum is 160 in the case of 8kHz sampling rate and 20 ms audio chunk size).

Let S_i be the sequence number of the i^{th} arrived packet, and let P_i be the packetizing period of the currently used codec.

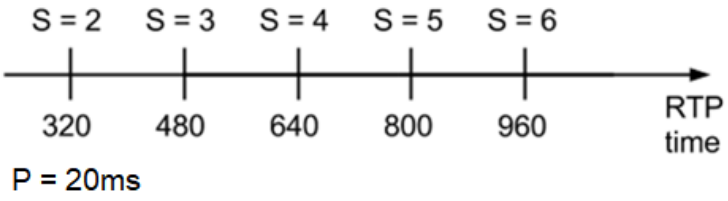

Figure 4.3: Timestamp increment in RTP packets.

Based on this scheduling scheme, the method applies a reference timeline, on which current time is incremented by the delta arrival time between two consecutive packets. A local time-based timestamping mechanism is used for the delta time calculation. The reference timeline will be used to classify an incoming packet based on its time domain properties. Let t_i be the arrival time and let d_i be the delta time of the i^{th} packet from the previously arrived packet. Let Ct be the current time that represents a time relative to the beginning of the measurement. When a packet arrives, the delta time is calculated and added to current time Ct (see Figure 4.4).

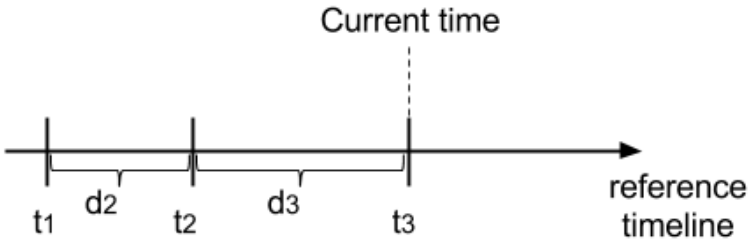

Figure 4.4: Reference time incremented by delta arrival times

Besides the calculation of the reference time, the new method applies windowing to

perform classification and to calculate the metrics. The proposed model handles a time-window (as an expected arrival interval) that is defined by two reference points in time. The reference points (and thereby the time-window) are re-calculated for each arrived IP packet in real-time and fitted on the reference timeline to determine the performance metrics (see Figure 4.5). The size of the expected arrival interval accommodates the size of the jitter puffer (100 ms, typically). Let Rp_{ij} be the j^{th} reference point of the i^{th} arrived packet, where j = 1, 2. The method assumes P_i to be known in each calculation cycle. When the RTP stream involves multiple payload types (i.e., multiple codecs and sampling period combinations), P_i should be properly synchronized to the arrival events. The rate could be preset, if it is previously known from the codec properties, or could be indirectly measured in the initial phase of the algorithm if it is unknown. The resolution of P_i should be equal to the resolution of Ct.

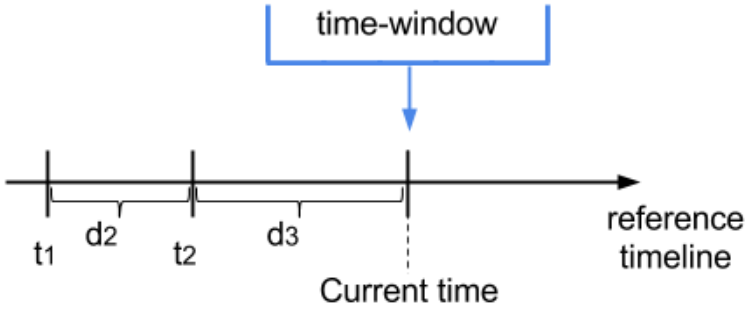

Figure 4.5: Time-window fitting on the reference timeline

Let T be a threshold value that defines a global time interval for the packet classification. Its value is determined by the playout buffering scheme of the voice application. T is practically an integer value, and T $\geq$ 2. The T threshold makes the method scalable for any playout buffer size. When P_i is a 20 ms value (which is a typical packetization time in VoLTE services), to determine a 100 ms jitter buffer size the T parameter should be 2. Using the previously defined parameters, Rp_{ik} reference points are calculated by formulas (4.1 - 4.2), respectively:

$$Rp_{i1} = (S_i \times P_i) - ((T + 0.5) \times P_i) \tag{4.1}$$

$$Rp_{i2} = (S_i \times P_i) + ((T + 0.5) \times P_i) \tag{4.2}$$

The time-window represents an application-specific interval as the expected window that could be increased by the T threshold value. Rp_{ik} reference points are updated for each arrived packet and the time-window is slided accordingly. Then it is fitted on the reference timeline to perform classification and to recognize packet loss events. Figure 4.6 - 4.7 present example calculations.

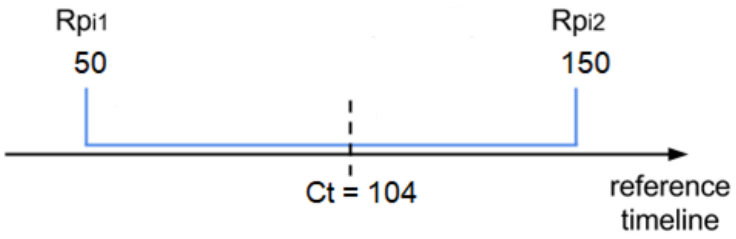

Figure 4.6: Time-window fitting in the case of i=5, P_i=20, Ct=104 and T=2

The model takes the arrived sequence number S_i and the packetizing period P_i to determine the expected arrival interval for the current voice frame (see Formulas 4.1 - 4.2).

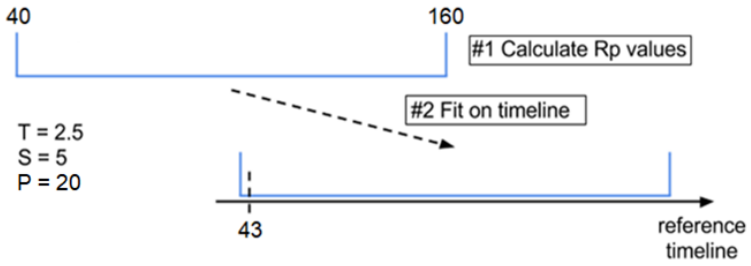

Figure 4.7: Time-window fitting in the case of i=5, P=20, Ct=43, T=2.5

The method has an initialization phase followed by an assessment phase. During the initial phase, the reference timeline and the packetizing period should be properly adjusted. A reference timer is typically a common local clock source, which is used to timestamp the arrived packets. However, the reference timeline is initialized by the $S_0 \times P_0$ product. If reordering or loss occurs in the init phase, the reference timeline must be re-initialized by the next in-order packet. The method maintains the initialization phase until the packetizing period is fixed and no out-of-order packet arrives for at least a $(4 + 2T) \times P_i$ period. This criterion grants that the values of the RP_{ik} reference points will be higher than the initial value of the timeline. As soon as the above conditions are met, the method steps into the assessment phase. The assessment phase relies on the previously introduced time-window.

My model introduces two loss categories for the arrived packets, which are differentiated by the time-window. Furthermore, it specifies an additional category for packets that are not arrived and therefore considered lost (namely, not-arrived-loss).

I differentiated the following loss categories:

- Early arrived loss (EAL)
- Late arrived loss (LAL)
- Not-arrived loss (NAL)

Since voice codecs and thus RTP operate with time frames, the decoding algorithm always has to process a sequence of voice frames. When a packet comes earlier than it is expected,

the media application stores its payload in the playout buffer. When a packet with an early arrival faces a filled buffer, the application may discard its payload (i.e., the voice frame) despite the packet arrived to the endpoint. This is because its playout time is far from the current time window. The method defines the early arrived loss metric for the representation of media frames arrived too early. Using the windowing technique, the packet is counted as early arrived loss when $Ct < Rp_{i1}$. Similarly, we can differentiate the late arrived loss event. When a packet arrives too late from its expected interval, the application may handle the missing time information with different techniques, because the received audio information is irrelevant in time. The method counts the i^{th} packet as late arrived loss when $Ct > Rp_{i2}$.

Figure 4.8 represents the loss categories related the expected arrival interval.

Figure 4.8: Loss categories based on the time-window

It should be noted that the not-arrived loss metric is also determined in the measurement phase, but it is calculated only from the sequence number and the T threshold. Based on the time-window, Ct exactly defines the relationship between the packet and the playout buffer usage.

Based on the experience of related researches [94], the burst packet loss is also a quality-reducing factor. During my research work, I found that 8-10 consecutive packet loss events, which span about a 160-200 ms time interval, resulting in a perceptible crackling sound. The number of crackling events is an annoying factor, which reduces the quality of the speech. It should be handled for a proper assessment algorithm. Let LDC be the loss discontinuity counter and LDT be a loss discontinuity threshold. LDC is incremented when a loss event occurs (early-, late- or not-arrived loss), and is reset to zero when it equals LDT or when the packet arrived in time. Let LDE be the loss discontinuity event, which is incremented by any LDC = LDT event.

4.4.3 Thesis 3.2

Thesis 3.2 - I have defined the slow and dynamic speech types as new categories in voice calls. I have provided a new method to distinguish them in voice calls. I have demonstrated that the weighted form of the new loss metrics, as defined in Thesis 3.1, correlates with the reference-based MOS estimation results. I have identified that the differentiation of the slow and dynamic speech types improves the accuracy of the proposed model. [J1] [C1]

During the quality analysis research steps, I recognized two speech types:

- slow speech, and
- dynamic speech.

To differentiate the slow and the dynamic speech categories, I briefly introduce the voice activity detection (VAD) method, which is commonly used in IP-based voice services. VAD is an important technique to reduce the bit rate of speech in VoLTE and VoIP applications. The method detects the silence periods of the speech and indicates the start and the end of the silence in the RTP flow. During a silence period, the sender side does not transmit voice packets, meanwhile the decoder at the receiver side generates comfort noise for the listener. This is a frequent scenario when the caller or the callee listens and does not say anything. These silence periods should be recognized for quality estimation and the slow or dynamic types should be differentiated based on the VAD properties.

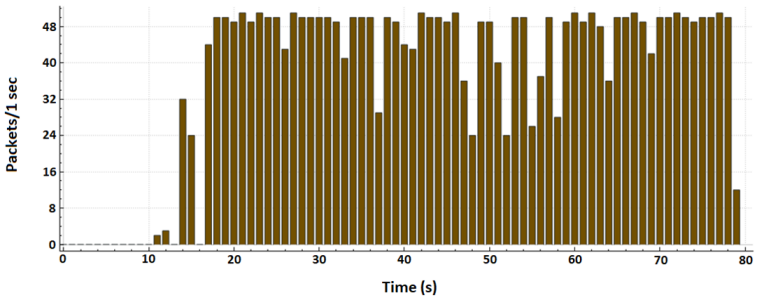

Figure 4.9: Packet rate pattern for dynamic speech

The primary question about differentiating slow and dynamic speeches is how the speech type can be identified without inspecting the voice payload. To solve this question, this thesis proposes a new real-time method that relies on the threefold relationship between speech type, non-VAD/VAD ratio and alteration of voice packet rate. The packet rate is constant (50 pps, typically) as long as the voice coder is not in VAD state, we can detect the ratio of non-VAD and VAD states as per time unit. Since in VAD mode the packet rate is significantly reduced by the codec, the speech type is based on the effective packet rate (corrected by the measured loss) versus the maximum packet rate (i.e., the constant 50 pps packet rate of the non-VAD mode). Real-life speech examples are shown in Figure 4.9 and 4.10.

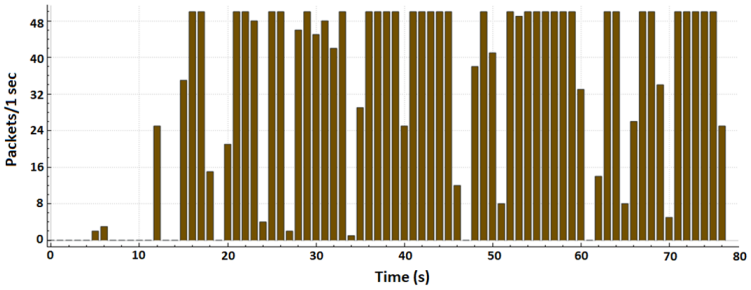

Figure 4.10: Packet rate pattern for slow speech

According to the analysis of 12 voice samples (see Table 4.2), the average of the effective packet rate is 0.883 x 50 pps for the dynamics speeches and 0.703 x 50 pps for the slow ones. In the next research steps, I applied a 0.8 threshold to automatically separate the two speech types.

Type of speech	#1	#2	#3	Average
Male - Dynamic	0.877	0.872	0.904	0.884
Male - Slow	0.726	0.767	0.653	0.715
Female - Dynamic	0.883	0.869	0.893	0.882
Female - Slow	0.722	0.619	0.732	0.691

Table 4.2: Speech type versus effective packet rate (pps)

By introducing a category quality indicator (CQI) for each loss-category as a category-specific weight, we can calculate a new quality estimation index on the common MOS-scale. This quality index represents the overall performance of the call session. It is important to note that by calculating a quality index the primary aim is to provide an indication of how a network path complies with the requirement of delivering voice traffic end-to-end. Typically, a low-level degradation of the transmission, in terms of timing, can be eliminated by the receiver side jitter buffer, and therefore the perceived service quality may be unaffected. Nevertheless, such low-level impairments can be forerunners for a negative trend within the network path (i.e., evolving congestion) that will degrade user experience in the very near future.

To determine the weights for each loss category (as the CQI weights), I generated several audio files with various impairment parameters. The measurement setup contained two laptops (laptop-A as caller and laptop-B as callee, see Figure 4.11) and an OpenWrt [118] router to emulate network behavior. I applied the NetEm (Network Emulator) Linux kernel module on the router to drop, delay and reorder the transmitted RTP packets. I used Ekiga [119] as a VoIP client on both laptops to perform IP-based voice calls. While the RTP packets captured losslessly, the speech waveform with Audacity [120] was also recorded. As voice references, I applied four independent samples from audiobooks: a slow (calm) male speech, a slow female speech, a dynamic male speech, and a dynamic female speech. I replayed the audio files and streamed it into the VoIP call. In the Ekiga client, the default codec type

was AMR-WB and the size of the jitter buffer was 100 ms.

Since the input jitter buffer eliminates the effect of a reordered packet with a relatively low sequence offset and grants an ordered playout, a reordered packet with a lower delay variation than the buffer capacity does not result voice quality degradation. However, a larger sequence offset drives to receiver-side packet elimination, i.e., data loss. I checked this statement with replayed audio files.

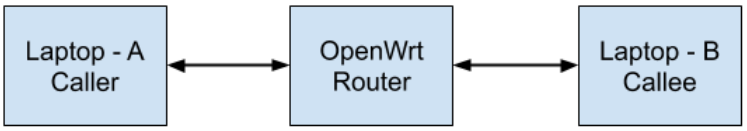

Figure 4.11: Laboratory setup for audio test file generation

In contrast to profile reordering and packet loss independently, I analyzed both side by side to provide a higher accuracy estimation of the voice quality degradation. I generated test audio files using the emulator system configured with 2, 5, 10, 20, 30 and 40 percent of packet loss, respectively. I applied the related network QoS scenarios on each of the four reference speeches. I calculated the MOS value for each output voice file using the AQuA [90] analyzer tool, and also measured its packet-level loss metrics according to the proposed method in Thesis 3.1. During my research, I profiled the early loss, late loss and not-arrived loss categories. I observed that a late arrived voice packet results in the same data loss scenario as a not-arrived loss packet. Both cases imply the same type of data loss at the input of the voice decoder. Based on the measurement results, I found that the characteristics of the early-loss differs from that of the not-arrived loss (see Figure 4.12 and 4.13). Accordingly, the MOS index should be determined by unique functions.

Using the calculated CQI indicators, I found correlation between the weighted loss metrics and AQuA MOS values. To define an appropriate function for calculating the CQIs, I profiled the loss categories based on the AQuA results.

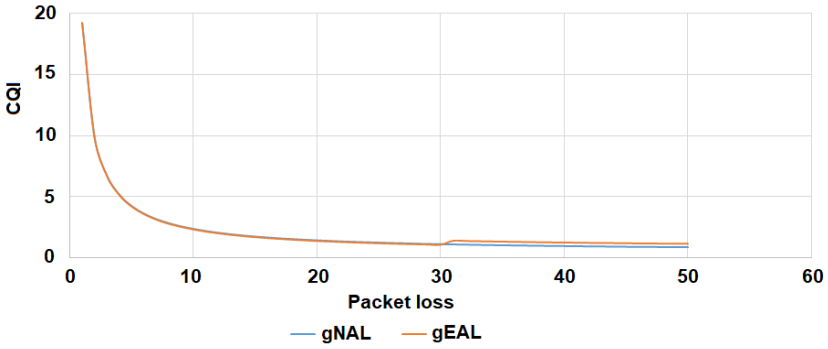

Figure 4.12: CQI weight functions of dynamic speech

To get general weight functions for the quality estimation, I applied regression analysis using curve estimation in IBM SPSS [121]. I used the inverse model for both slow and dynamic speeches. The results of the linear regression on the not-arrived loss weights show a correlation of 99.8% in the case of dynamic speech, and 100% in the case of slow speech.

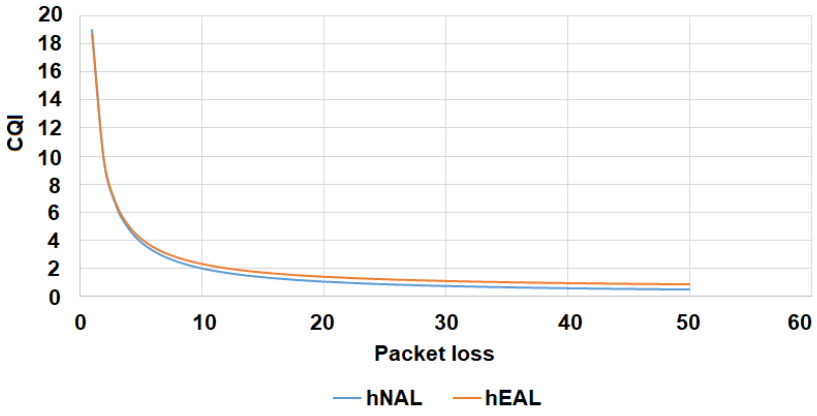

Figure 4.13: CQI weight functions of slow speech

Considering the early arrived loss, the regression analysis reveals a correlation of 98.2% for the dynamic speech and 99.6% for the slow one. Using the measured loss value as an explanatory variable, the weights for different speech types can be determined by unique predictor functions. Based on the outcomes of the inverse model-based regression analysis, I defined two weight (predictor) functions (4.3, 4.4) for the slow speech segments ($hNAL(p)$ for not-arrived loss and $hEAL(p)$ for early loss), and two others (4.5, 4.6) for the dynamic speech segments ($gNAL(p)$ for not-arrived loss and $gEAL(p)$ for early loss). The p explanatory variable is the packet loss rate for the given time period.

$$h_{NAL}(p) = 0.093 + \frac{0.18923}{p} \tag{4.3}$$

$$h_{EAL}(p) = 0.48 + \frac{0.182}{p} \tag{4.4}$$

$$g_{NAL}(p) = 0.453 + \frac{0.18796}{p} \tag{4.5}$$

$$g_{EAL}(p) = \begin{cases} 0.453 + \frac{0.18796}{p} & , p \leq 0.3 \\ 0.747 + \frac{0.206}{p} & , p > 0.3 \end{cases} \tag{4.6}$$

In the case of late arrived loss, I used the predictor functions (4.3) and (4.5) for determining the appropriate CQI weights. This difference suggests that the punishment of the early- and the late- or not-arrived loss should not be equal. I note that each of the weight functions (4.3)-(4.6) assumes a non-zero p value. When the p loss value is zero, CQI calculation is omitted.

4.4.4 Thesis 3.3

Thesis 3.3 - I have provided MOS estimation formulas for slow and dynamic speech types based on the new metric set, defined in Thesis 3.1, and based on the new weight calculation functions, defined in thesis 3.2. I have validated the estimation method with laboratory measurements. [J1] [C1]

To estimate the MOS index from the loss values presented in Thesis 3.1, the three loss categories (i.e., early loss, late loss, not-arrived loss) have to be weighted by the previously defined CQI functions (see Thesis 3.2). For a higher correlation, the burst loss events should be also recognized, and the LDE value (see Thesis 3.2) should be also weighted. Let w_{LDE} be the proper weight for the LDE counter based on the measurement samples (see Table 4.3). The w_{LDE} values are determined based on the highest correlation between my estimation formulas and the AQuA software results.

Type of speech	LDE $<$ 10	10 $<$ LDE $<$ 15	15 $<$ LDE
Dynamic	0.012	0.006	0.002
Slow	0.002	0.004	0.008

Table 4.3: Summary table for LDE weights

Let P_c be the rate of the given category counter versus the total RTP packets sent, where c means the category (NAL, EAL, and LAL) and $0 \leq P_c \leq 1$. The estimated QoE value is calculated by (4.7)-(4.10). Let P_{SL} be the sum of P_{NAL} and P_{LAL}. Let $f(p, st)$ be the function, used in (4.7), where p is the packet loss and st is the speech type, that returns the appropriate CQI function ($h_{NAL}(p)$, $h_{EAL}(p)$, $g_{NAL}(p)$ or $g_{EAL}(p)$, respectively) based on the speech type and the corresponding loss categories.

$$M_1 = \frac{(1 - P_{SL} \times f(P_{SL}, st)) + (1 - P_{EAL} \times f(P_{EAL}, st))}{2} \tag{4.7}$$

$$M_2 = w_{LDE} \times LDE \tag{4.8}$$

$$M_3 = M_1 - M_2 \tag{4.9}$$

$$MOS = 5 \times M_3 \tag{4.10}$$

To validate the voice quality assessment model (4.10) in a laboratory environment, I implemented the proposed model in $C++$, and applied it on sample capture files. The measurement setup for the validated sample generation was similar to the applied measurement setup in Thesis 3.2. I used two x86-64 workstations with Ubuntu operating system as caller and callee, and the NetEM kernel module in both directions between them to emulate network impairments. To validate the accuracy of the MOS estimation model, and also the robustness of the applied parameters (T = 2, and 0.8 threshold value for the speech type

recognition), I generated 480 test cases. These traffic samples are derived from slow and dynamic reference speeches. I applied Ekiga [119] to perform the VoIP calls, and arecord to redirect and record the audio files. I selected four different new voice samples (not overlapping with ones used for the regression analysis) as reference from audiobooks. For each test case generation, I chose one of the four reference speeches and messed up the network properties. To induce early loss, not-arrived loss and late loss events, I activated and deactivated the following NetEm kernel module parameters (network delay, reordering and loss) during the call:

- loss i% (where i = 0 ... 40) to induce NAL,
- delay 200ms reorder j% (where j = 100 ... 60) to induce LAL,
- delay 200ms reorder i% (where i = 0 ... 40) to induce EAL.

I applied the AQuA analyzer tool to calculate the MOS values for each speech sample. To determine the accuracy of the presented quality assessment model, I calculated the different loss category counters per speech sample and applied the MOS formula (4.10).

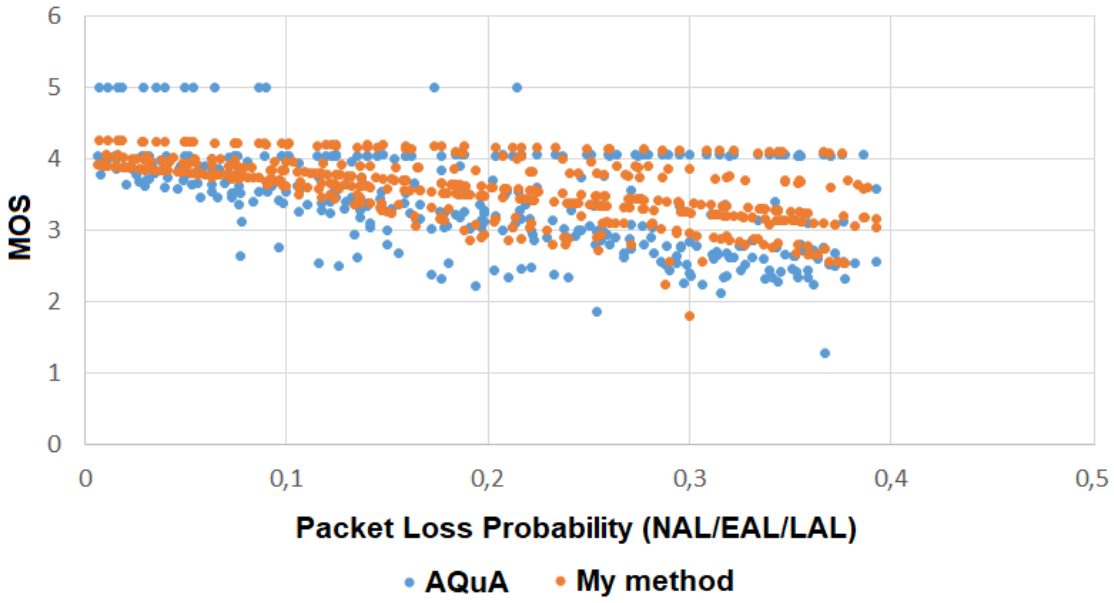

Figure 4.14: Validation results: correlation between AQuA MOS and my estimated MOS values

Figure 4.14 represents the correlation between AQuA MOS and the estimated MOS values by formula (4.10). The validation results reveal that the absolute error range is between 0.002 and 1.4 MOS, and the average delta MOS is 0.285. To conclude the observations, the difference between AQuA and my estimated MOS in 82% of test cases is less or equal to 10% (0.5 MOS), in 74% of the cases is less or equal to 8%, and in 48% of the cases is less or equal to 4% (0.2 MOS).

The results show that the average delta percent (see Table 4.4), which comes from the difference between AQuA [90] MOS and the proposed model-based MOS values, is less than

Impairment	Speech type	Average delta MOS
Not-arrived loss	dynamic	0.30
Not-arrived loss	slow	0.29
Early arrived loss	dynamic	0.26
Early arrived loss	slow	0.32
Late arrived loss	dynamic	0.34
Late arrived loss	slow	0.20
Overall		0.285

Table 4.4: Summary of MOS assessment validation

6.8% in the case of dynamic speech, and less than 6.4% in the case of slow speech. The slow speech test cases frequently contain 1-3 seconds long silence blocks. Table 4.4 shows the summary of the validation results.

4.4.5 Conclusion about service quality analysis

The evolution of HD voice-based mobile communication expects several transmission parameters to be kept under tight control to grant reliable and high-quality voice services. It necessitates a user-centric service quality monitoring of VoLTE and VoIP applications in real-time.

Since the essential packet-level metrics are not appropriate for application-level loss calculation, they cannot be used for a precise voice quality estimation. In contrast to previously published works, which are based on reference voice materials or pure packet-level counters, I presented a new metric calculation method. As a new metric set, I introduced 3 loss categories (early arrived loss, late arrived loss and not-arrived loss), which are especially calculated to serve voice quality assessment models. My approach to determining service quality is based on the analysis of media data availability inside the endpoint's playout buffer that is a key element of the media delivery chain. However, instead of analyzing media data directly at the endpoint's playout buffer, I modeled its operation in an independent measurement system using a time windowing method.

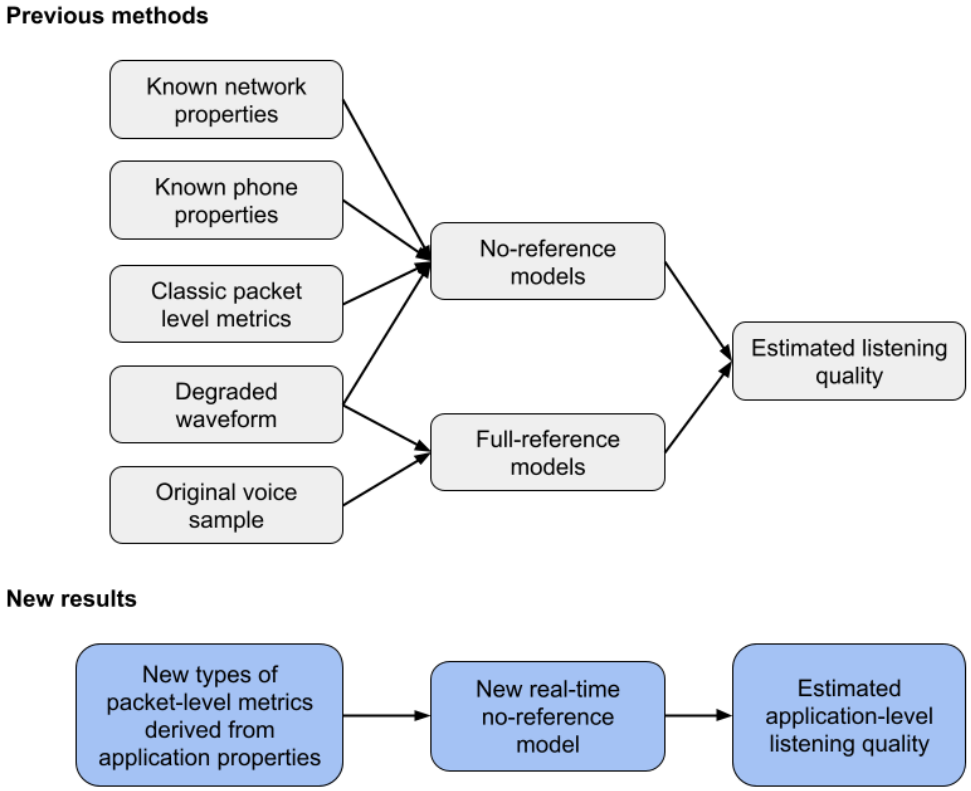

Figure 4.15: Novelties (colored in blue) in contrast to the previous models

The main advantage of Thesis 3.1 (i.e., an integrated reorder-loss calculation) is the ability to determine the status of the arrived IP/RTP packet from the application's viewpoint. Based on these statistics, we get a cross-layer overview that can be applied for estimating the quality of user experience. I also introduced a No-Reference quality estimation model, which operates on the new metric set and produces a MOS index. Figure 4.15 represents the

new results (colored in blue) in relation to the previous quality estimation models. Next to a new metric calculation and quality assessment model definition I proposed a new method to identify the pace of the speech (dynamic or slow) as long as voice activity detection (VAD) is present between the endpoints. This identification supports the introduced quality model to estimate the perceptive quality with higher accuracy.

I validated my quality estimation method against a full-reference voice quality estimation model called AQuA with 8 different voice materials (4 male and 4 female samples) in many different network scenarios. I found high correlation between formula 4.10-defined MOS and AQuA MOS in all scenarios. The validation results showed that applying my research results (i.e., CQI weights, three types of loss metrics, loss burst measurement, and speech type detection), the MOS index could be estimated with an average of 5.7% error ratio.

Chapter 5

Results Applied in Practice

The hardware-based packet parsing method, which is presented by Thesis group 1, was applied and implemented as a prototype for the national project C-GEP, GOP-1.1.1-11-2012-0031, in Hungary. After the project ends, the method was further generalized and applied by AITIA International Inc., and Magyar Telekom Nyrt. for VoLTE signalling analysis.

The SIP protocol-based new Call Data Record generation method, described by Thesis group 2, was applied and implemented for Magyar Telekom Nyrt. as a part of a VoLTE traffic analysis system. The implemented form of the method is currently operating on the core network traffic, and I am in touch with the operators, who send me continuous feedback about the management system.

The RTP protocol-based new metric calculation model, which is detailed by Thesis 3.1, was applied and implemented as a prototype for FIRST (Future Internet Research, Services, Technology) project, TÁMOP-4.2.2.C-11/1/KONV-2012-0001, in Hungary. After the project ends, the metric set was further analyzed and used to create a new No-Reference model, which is presented by Thesis 3.1 - 3.3.

To convey the collected knowledge for the university students, I created a new educational material and started a new course, which title is "Haladó LTE hálózati vizsgálatok".

Summarizing the results achieved, my new methods are applied by the telecommunication operators for everyday tasks, and also help to understand and analyze complex message sequences.

Regarding the latest 5G network [122] and IMS domain standard [33], the IMS remains the central element for call control. Since the main protocols will do not change within the IMS network, my research results are also applicable to analyze the 5G call services.

Chapter 6

Summary

Although its first commercial deployments happened in 2014, Voice over LTE is still not yet a mature and reliable service. There is a reason behind this delay: it is hard to meet the expected quality of service when the infrastructure and the service logic reached such a high complexity. The evolution of HD voice-based mobile communication expects several transmission parameters to be kept under tight control to grant reliable and high-quality voice services.

Network and service analysis is one of the key tools for providing feedback about quality. Regarding calls, passive monitoring-based CDR assembly is crucial for the operators in order to be able to trace what is working well - and what is not. The Voice over LTE service is an ideal candidate for showing the complexity of telecommunication core network analysis. The LTE EPC and the IMS domain are built upon different principles, and key parameters have to be identified for cross-interface correlation. The situation becomes even more complicated when the identifiers are hided or changed by the network. Understanding these cases and providing methods for fault and performance management is important, but also challenging.

Next to the control traffic analysis, the data plane, which carries the audio traffic, needs to be also processed for a global view. Since the signalling traffic and also the user traffic changed to an IP-based model, the root cause analysis and failure detection methods should also handle new situations. The core network operators encounter new problems during the everyday work, and need new management aspects to handle these special problem spaces.

In my research work, I focused on new methods and models for the new challenges and message sequence scenarios in VoLTE-related calls. Starting from the hardware-accelerated, complex protocol stack parsing, I proposed a new CDR generation method for SIP protocol-based interfaces. Besides the signalling traffic analysis, my dissertation presents a new measurement method for RTP-based call quality estimation.

In Chapter 2 I summarized the theoretical and technological background of hardware-based packet processing solutions. Next to the challenges, my dissertation reveals the possible trade-offs of the hardware-accelerated packet parsing approaches. The presented thesis group helps to design a complex protocol parsing engine, and shows the viable options in worst-case scenario.

Besides covering the generic problems of SIP protocol-based monitoring, I presented CDR collection methods in Chapter 3 for complex message sequences. Starting from simple use cases, and detailing more complex ones, my research work presents complete methods for SIP

CDR assembly that works in current LTE scenarios. After the definition of the challenges, I introduced what messages and parameters can define the CDRs and how. I also suggest what interfaces to monitor and what are the key parameters to use for cross-correlation.

Chapter 4 examines another problem space: RTP protocol-based call quality assessment. The presented method operates on the data plane traffic to measure the packet loss properties in real-time. I introduced a new, classification-based metric calculation method to differentiate the reorder-loss concept in RTP traffic. Using this metric set, I also introduced a new, no-reference MOS estimation method.

Regarding my scientific work, I comprehensively examined the entire VoLTE system, from the low-level packet processing to the higher-level message sequence analysis, including the control plane and also the user plane.

Acknowledgements

I would like to express my gratitude for all people who have supported my research. My special thanks go:

- to Péter Orosz, who provided me with endless technical ideas together with motivation during my studies and work,
- to Pál Varga, who supported me during my work and gave me enough "free time",
- to all the colleagues at AITIA, Telecom Hungary and BME, for the challenges they let me face with,
- to last but not least, my family

Thank you.

Bibliography

[1] "Mean Opinion Score (MOS) terminology." [Online]. Available: https://www.itu.int/rec/T-REC-P.800.1-200303-S/en

[2] Napatech, "Building Intelligent Mobile Data Services Using Deep Packet Instpection," 2011.

[3] Xilinx, "What is an fpga?" 2018. [Online]. Available: https://www.xilinx.com/products/silicondevices/fpga/what-is-an-fpga.html

[4] NXP, "C-5 network processor - silicon revision d0," 2018. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/C5NPD0-DS.pdf

[5] Nvidia, "What is gpu-accelerated computing?" 2018. [Online]. Available: http://www.nvidia.com/object/what-is-gpu-computing.html

[6] K. Septinus, P. Pirsch, H. Blume, and U. Mayer, "A Fully Programmable FSM-based Processing Engine for Gigabytes/s Header Parsing," in *International Conference on Embedded Computer Systems*, 2010.

[7] M. Avalle, F. Risso, and R. Sisto, "Scalable Algorithms for NFA Multi-Striding and NFA-Based Deep Packet Inspection on GPUs," *IEEE/ACM Transaction on Networking*, vol. 24, pp. 1704–1717, 2016.

[8] P. Benácek, V. Pus, and H. Kubátová, "P4-to-VHDL: Automatic Generation of 100 Gbps Packet Parsers," in *IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines*, 2016.

[9] P. Kobiersky, J. Korenek, and L. Polack, "Packet Header Analysis and Field Extraction for Multigigabit Networks," in *12th International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2009., pp. 96–101.

[10] M. Attig and G. Brebner, "400 Gb/s Programmable Packet Parsing on a Single FPGA," in *ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*, 2011., pp. 12–23.

[11] G. Brebner and W. Jiang, "High-Speed Packet Processing using Reconfigurable Computing," *IEEE Micro*, vol. 34, pp. 8–18, 2014.

[12] C. Kozanitis, J. Huber, S. Singh, and G. Varghese, "Leaping multiple headers in a single bound: wire-speed parsing using the kangaroo system," in *29th IEEE Conference on Computer Communications*, 2010., pp. 830–838.

[13] "NetFPGA project." [Online]. Available: http://www.netfpga.org

[14] G. Gibb, G. Varghese, M. Horowitz, and N. McKeown, "Design Principles for Packet Parser," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2013., pp. 13–24.

[15] T. Duan, J. Shen, P. Wang, and S. Liu, "A Self-Adaptive Programming Mechanism for Reconfigurable Parsing and Processing," *China Communications*, vol. 13, pp. 87–97, 2016.

[16] V. Pus, L. Kekely, and J. Korenek, "Low-Latency Modular Packet Header Parser for FPGA," in *9th ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, 2012., pp. 77–78.

[17] P. Benácek, V. Pus, J. Korenak, and M. Kekely, "Line Rate Programmable Packet Processing in 100Gb Networks," in *27th International Conference on Field Programmable Logic and Applications*, 2017.

[18] X. L. Hu, X. R. Wang, L. Wang, and H. Li, "Design of Extensible Forwarding Element Architecture and its Key Technology Verification," in *International Conference on Integrated Circuits and Microsystems*, 2016.

[19] J. Li and Z. S. B. Han, "P5: Programmable Parsers with Packet-level Parallel Processing for FPGA-based Switches," in *ACM/IEEE Symposium on Architectures for Network and Communications Systems*, 2017.

[20] V. Pus, L. Kekely, and J. Korenek, "Design Methodology of Configurable High Performance Packet Parser for FPGA," in *17th International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2014.

[21] J. S. da Silva, F.-R. Boyer, and J. M. P. Langlois, "P4-compatible High-level Synthesis of Low Latency 100 Gb/s Streaming Packet Parsers in FPGAs," in *26th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'18)*, 2018.

[22] P. Orosz, T. Tóthfalusi, and P. Varga, "C-GEP: Adaptive Network Management with Reconfigurable Hardware," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.

[23] P. Varga, L. Kovács, T. Tóthfalusi, and P. Orosz, "C-GEP: 100 Gbit/s Capable, FPGA-based, Reconfigurable Networking Equipment," in *IEEE 16th International Conference on High Performance Switching and Rouging (HPSR)*, 2015.

[24] X. Inc., "Introduction to fpga design with vivado high-level synthesis," 2018. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/ug998-vivado-intro-fpga-design-hls.pdf

[25] M. B. Anwer, M. Motiwala, M. B. Tariq, and N. Feamster, "SwitchBlade: A Platform for Rapid Deployment of Network Protocols on Programmable Hardware," in *ACM SIGCOMM*, 2010., pp. 183–194.

[26] V. Moreno, J. Ramos, P. M. S. del Río, J. L. García-Dorado, F. J. Gomez-Arribas, and J. Aracil, "Commodity Packet Capture Engines: Tutorial, Cookbook and Applicability," *IEEE Communications Surveys and Tutorials,*, vol. 17, pp. 1364–1390, 2015.

[27] HiTech Global, "100 Gbps Ethernet MAC IP Core," 2018. [Online]. Available: http://www.hitechglobal.com/IPCores/100GigEthernet-MAC-PCS.htm

[28] Mantaro, "100 Gbps Ethernet MAC IP Core," 2018. [Online]. Available: http://www.mantaro.com/products/fpga-ip-cores.htm

[29] Hiteksys, "100 Gbps Ethernet MAC IP Core," 2018. [Online]. Available: http://hiteksys.com/products/fpga-ip-cores/100g-ethernet

[30] IEEE, "Csma/cd access method and physical layer specifications amendment 4: Media access control parameters, physical layers, and management parameters for 40 gb/s and 100 gb/s operation," 2010.

[31] Wireshark. [Online]. Available: https://www.wireshark.org/

[32] CISCO, "Cisco visual networking index: Global mobile data traffic forecast update. 2015-2020 white paper," 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[33] 3GPP, "IP Multimedia Subsystem (IMS). TS 23.228," 2016. [Online]. Available: http://www.3gpp.org/DynaReport/23228.htm

[34] D. Kozma, G. Soos, and P. Varga, "Supporting LTE Network and Service Management through Session Data Record Analysis," *Infocommunications Journal*, vol. 71(2), pp. 11–16, 2016.

[35] 3GPP, "Key Performance Indicators (KPI) for the Evolved Packet Core (EPC), 3GPP TS 32.455."

[36] ——, "Key Performance Indicators (KPI) for the IP Multimedia Subsystem (IMS), 3GPP TS 32.454."

[37] P. Varga, G. Szelindi, G. Sey, and E. Cseszko, "Az LTE maghálózat monitorozásának kihívásai és megoldásai," *Híradástechnika*, pp. 36–42, 2012.

[38] G. Breda and L. Mendes, "QoS Monitoring and Failure Detection," in *International Telecommunications Symposium*, 2006.

[39] P. Lutiis and D. Lombardo, "An Innovative Way to Analyze Large ISP Data for IMS Security and Monitoring," in *13th International Conference on Intelligence in Next Generation Networks*, 2009.

[40] M. Nassar, R. State, and O. Festor, "A Framework for Monitoring SIP Enterprise Networks," in *4th International Conference on Network and System Security*, 2010.

[41] B. Raouyane, M. Bellafkih, M. Errais, D. Leghroudi, D. Ranc, and M. Ramdani, "eTOM Business Processes Conception in NGN Monitoring," *Springer Berlin Heidelberg*, pp. 133–143, 2011.

[42] D. Hoffstadt, S. Monhof, and E. Rathgeb, "SIP Trace Recorder: Monitor and analysis tool for threats in SIP-based networks," in *8th International Wireless Communications and Mobile Computing Conference*, 2012.

[43] L. Zhang, Z. Zhang, X. Cui, and D. Liu, "Research on the Monitoring and Controlling Model of SIP Network," in *IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013.

[44] J. Balcerzak, G. Tyszka, and S. Senecal, "QoS Monitoring Model of Registration Procedure for IMS Platform," in *16th International Telecommunications Network Strategy and Planning Symposium*, 2014.

[45] J. Hyun, J. Li, C. T. Im, J. Yoo, and J. W. Hong, "A VoLTE Traffic Classification Method in LTE Network," in *16th Asia-Pacific Network Operations and Management Symposium*, 2014.

[46] ——, "A High Performance VoLTE Traffic Classification Method using HTCondor," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.

[47] C. Li, G. Tu, C. Peng, Z. Yuan, Y. Li, S. Lu, and X. Wang, "Insecurity of Voice Solution VoLTE in LTE Mobile Networks," in *22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[48] M. Olsson, S. Sultana, S. Rommer, L. Frid, and C. Mulligan, *SAE and the Evolved Packet Core*. Oxford, UK: Academic Press, 2009.

[49] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, 2002.

[50] J. Rosenberg and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers," RFC 3263, 2002.

[51] ITU-T, "Specifications of Signalling System No. 7 - ISDN user part," International Telecommunication Union, Recommendation Q.763, Dec. 1999.

[52] 3GPP, "Application of the Base Station System Application Part (BSSAP) on the E interface," 3rd Generation Partnership Project (3GPP), TS 49.008, Dec. 2015. [Online]. Available: http://www.3gpp.org/DynaReport/49008.htm

[53] 3GPP, "Media Gateway Control Function (MGCF) - IM Media Gateway," Mar. 2016. [Online]. Available: http://www.3gpp.org/DynaReport/29332.htm

[54] M. Poikselka and G. Mayer, *The IMS: IP Multimedia Concepts and Services.* Chichester, UK: Wiley, 2009.

[55] M. Poikselka, H. Holma, J. Hongisto, J. Kallio, and A. Toskala, *Voice over LTE (VoLTE).* Wiley, 2012.

[56] A. B. Roach, "Session Initiation Protocol (SIP)-Specific Event Notification," RFC 3265, 2002.

[57] K. Sultan, H. Ali, and Z. Zhang, "Call Detail Records Driven Anomaly Detection and Traffic Prediction in Mobile Cellular Networks," *IEEE Access*, vol. 6, pp. 41 728–41 737, 2018.

[58] M. Kumar, M. Hanumanthappa, and T. V. S. Kumar, "Crime Investigation and Criminal Network Analysis using Archive Call Detail Records," in *Eighth International Conference on Advanced Computing (ICoAC)*, 2017.

[59] O. Jukic and I. Hedi, "The use of Call Detail Records and Data Mart Dimensioning for Telecommunication Companies," in *IEEE 20th Telecommunications Forum (TELFOR)*, 2012.

[60] R. Sikder, M. J. Uddin, and S. Halder, "An Efficient Approach of Identifying Tourist by Call Detail Record Analysis," in *IEEE International Workshop on Computational Intelligence*, 2016.

[61] Q. Lin and Y. Wan, "Mobile Customer Clustering Based on Call Detail Records for Marketing Campaigns," in *IEEE International Conference on Management and Service Science*, 2009.

[62] J. Holub, M. Wallbaum, N. Smith, and H. Avetisyan, "Analysis of the Dependency of Call Duration on the Quality of VoIP Calls," *IEEE Wireless Communications Letters*, vol. 7, pp. 638–641, 2018.

[63] J. Rosenberg, "The Session Initiation Protocol (SIP) UPDATE Method," RFC 3311, 2002.

[64] S. Donovan, "The SIP INFO Method," RFC 2976, 2000.

[65] G. Camarillo, C. Holmberg, and Y. Gao, "Re-INVITE and Target-Refresh Request Handling in the Session Initiation Protocol (SIP)," RFC 6141, 2011.

[66] C. Holmberg, "Number Portability Parameters for the "tel" URI," RFC 6228, May 2011.

[67] D. Worley, M. Huelsemann, R. Jesske, and D. Alexeitsev, "Completion of Calls for the Session Initiation Protocol (SIP)," RFC 6910, 2013.

[68] 3GPP, "Circuit switched (cs) fallback in evolved packet sysetm (eps), 3gpp ts 23.272."

[69] F. Janky and P. Varga, "Time Synchronization Solution for FPGA-based Distributed Network Monitoring," *Infocommunications Journal*, vol. 10, 2018.

[70] D. Kozma and P. Varga, "Traffic Analysis Methods for the Evolved Packet Core," in *5th Mesterproba at BME*, 2006.

[71] P. Varga, D. Kozma, and T. Tothfalusi, "Assembling VoLTE CDRs based on Network Monitoring – Challenges with Fragmented Information," in *IFIP/IEEE IM*, 2017.

[72] IETF, "IETF IP Performance Measurement Work Group." [Online]. Available: https://tools.ietf.org/wg/ippm/

[73] G. Almes, S. Kalidindi, M. Zekauskas, and A. Morton, "A One-Way Loss Metric for IP Performance Metrics (IPPM)," IETF, RFC 7680, January 2016. [Online]. Available: https://tools.ietf.org/html/rfc7680

[74] ——, "A One-Way Delay Metric for IP Performance Metrics (IPPM)," IETF, RFC 7679, January 2016. [Online]. Available: https://tools.ietf.org/html/rfc7679

[75] C. Demichelis and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)," IETF, RFC 3393, November 2002. [Online]. Available: https://tools.ietf.org/html/rfc3393

[76] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser, "Packet Reordering Metrics," IETF, RFC 4737, November 2006. [Online]. Available: https://tools.ietf.org/html/rfc4737

[77] A. Morton, "Active and Passive Metrics and Methods (with Hybrid Types In-Between)," IETF, RFC 7799, May 2016. [Online]. Available: https://tools.ietf.org/html/rfc7799

[78] A. Morton and S. V. den Berghe, "Framework for Metric Composition," IETF, RFC 5835, April 2010. [Online]. Available: https://tools.ietf.org/html/rfc5835

[79] J. Mahdavi and V. Paxson, "IPPM Metrics for Measuring Connectivity," IETF, RFC 2678, September 1999. [Online]. Available: https://tools.ietf.org/html/rfc2678

[80] R. Koodli and R. Ravikanth, "One-way Loss Pattern Sample Metrics," IETF, RFC 3357, August 2002. [Online]. Available: https://tools.ietf.org/html/rfc3357

[81] A. Morton and B. Claise, "Packet Delay Variation Applicability Statement," IETF, RFC 5481, March 2009. [Online]. Available: https://tools.ietf.org/html/rfc5481

[82] ITU-T, "Performance parameter definitions for quality of speech and other voiceband applications utilizing IP networks," 2003.

[83] ——, "P.10/G.100, Vocabulary for performance, quality of service and quality of experience," 2017.

[84] ——, "P.800.1, Mean opinion score (MOS) terminology," 2016.

[85] ——, "P.562, Analysis and interpretation of INMD voice-service measurements," 2004.

[86] ——, "P.80, Methods for subjective determination of transmission quality," 1993.

[87] ——, "G.107, The E-model: a computational model for use in transmission planning," 2015.

[88] ——, "P.862, Perceptual Evaluation of Speech Quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," 2001.

[89] ——, "Perceptual Objective Listening Quality Assessment," 2018. [Online]. Available: https://www.itu.int/rec/T-REC-P.863

[90] Sevana, "AQuA - Audio Quality Analyzer." [Online]. Available: https://sevana.biz/products-aqua/

[91] "The e-model, a computational model for use in transmission planning." [Online]. Available: https://www.itu.int/rec/T-REC-G.107

[92] A. Takahashi, H. Yoshino, and N. Kitawaki, "Perceptual QoS Assessment Technologies for VoIP," *IEEE Communications Magazie*, vol. 42, pp. 28–34, 2004.

[93] A. Raake, "Short- and Long-Term Packet Loss Behavior: Towards Speech Quality Prediction for Arbitrary Loss Distributions," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1957–1968, 2006.

[94] Y. Jung and C. Manzano, "Burst Packet Loss and Enhanced Packet Loss based Quality Model for Mobile Voice-over Internet Protocol Applications," *IET Communications*, vol. 8, pp. 41–49, 2014.

[95] S. Broom and M. Hollier, "Speech Quality Measurement Tools for Dynamic Network Management," in *Measurement of Speech, Audio and Video Transmission Quality In Telecommunication Networks*, 2003.

[96] S. R. Broom, "VoIP Quality Assessment: Taking Account of the Edge-Device," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1977–1983, 2006.

[97] Y. Ouyang, T. Yan, and G. Wang, "CrowdMi: Scalable and Diagnosable Mobile Voice Quality Assessment Through Wireless Analytics," *IEEE Internet of Things*, vol. 2, pp. 287–294, 2015.

[98] D. Luksa, S. Faajt, and M. Krhen, "Sound Quality Assessment in VOIP Environment," in *37th International Convention on Information and Communication Technology*, 2014.

[99] W. Zou, F. yang, and X. Li, "A packet-layer quality assessment system for voip using random forest," in *IEEE International Conference on Computer and Information Technology*, 2014.

[100] Y. Han and G. Muntean, "Hybrid Real-time Quality Assessment Model for Voice over IP," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2015.

[101] N. Majed, S. Ragot, X. Lagrange, and A. Blanc, "Delay and Quality Metrics in Voice over LTE (VoLTE) Networks: an End-Terminal Perspective," in *International Conference on Computing, Networking and Communications*, 2017.

[102] M. Abareghi, M. M. Homayounpour, M. Dehghan, and A. Davoodi, "Improved ITU-P.563 Non-Intrusive Speech Quality Assessment Method for Covering VOIP Conditions," in *10th International Conference on Advanced Communication Technology*, 2008.

[103] ITU-T, "P.563, Single-ended method for objective speech quality assessment in narrow-band telephony applications," 2004.

[104] A. E. Conway, "A Passive Method for Monitoring Voice-over-IP Call Quality with ITU-T Objective Speech Quality Measurement Methods," in *IEEE International Conference on Communications*, vol. 4, 2002., pp. 2583–2586.

[105] L. Sun and E. C. Ifeachor, "Voice Quality Prediction Models and their Application in VoIP Networks," *IEEE Transactions on Multimedia*, vol. 8, pp. 809–820, 2006.

[106] J. C. w. Lin and P. Fournier-Viger, "Employing the Neural Networks to Parametrically Assess the Quality of a Voice Call," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, 2016., pp. 1–5.

[107] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF, RFC 3550, July 2003. [Online]. Available: https://tools.ietf.org/html/rfc3550

[108] "Wideband Coding of Speech at Around 16 kbit/s using Adaptive Multi-Rate Wideband (AMR-WB)." [Online]. Available: https://www.itu.int/rec/T-REC-G.722.2/en

[109] J. Valin, K. Vos, and T. Terriberry, "Definition of the Opus Audio Codec," IETF, RFC 6716, September 2012. [Online]. Available: https://tools.ietf.org/html/rfc6716

[110] "Speex: A Free Codec for Free Speech." [Online]. Available: https://www.speex.org

[111] "Perceptual Evaluation of Speech Quality (PESQ)." [Online]. Available: http://www.itu.int/rec/T-REC-P.862

[112] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser, "Packet Reordering Metrics," IETF, RFC 4737, November 2006. [Online]. Available: https://tools.ietf.org/html/rfc4737

[113] A. Jayasumana, N. Piratla, T. Banka, A. Bare, and R. Whitner, "Packet Reordering Metrics," IETF, RFC 5236, June 2008. [Online]. Available: https://tools.ietf.org/html/rfc5236

[114] A. Morton and B. Claise, "Packet Delay Variation Applicability Statement," IETF, RFC 5481, March 2009. [Online]. Available: https://tools.ietf.org/html/rfc5481

[115] C. Demichelis and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)," IETF, RFC 3393, November 2002. [Online]. Available: https://tools.ietf.org/html/rfc3393

[116] V. Raisanen, G. Grotefeld, and A. Morton, "Network Performace Measurement with Periodic Streams," IETF, RFC 3432, November 2002. [Online]. Available: https://tools.ietf.org/html/rfc3432

[117] A. Morton, G. Ramachandran, and G. Maguluri, "Reporting IP Network Performance Metrics: Different Points of View," IETF, RFC 6703, August 2012. [Online]. Available: https://tools.ietf.org/html/rfc6703

[118] OpenWrt, "Openwrt." [Online]. Available: https://openwrt.org/

[119] Ekiga, "Ekiga software." [Online]. Available: http://www.ekiga.org/

[120] Audacity, "Audacity software." [Online]. Available: http://www.audacityteam.org/

[121] IBM Analytics, "IBM SPSS Software." [Online]. Available: https://www.ibm.com/analytics/us/en/technology/spss/

[122] 3GPP, "System architecture for the 5G System (5GS). TS 23.501," 2020.

Publications

[J] Journal Papers

[J1] P. Orosz, T. Tothfalusi, *A New Application-aware No-reference Quality Assessment Method for IP Voice Services*, Journal of Network and Systems Management (JONS), 2021, 10.1007/s10922-021-09595-8

[J2] P. Orosz, T. Tothfalusi, P. Varga, *FPGA-Assisted DPI Systems: 100 Gbit/s and Beyond*, IEEE Communications Surveys and Tutorials, pp. 1–27, 27 p., 2018

[J3] T. Tothfalusi, P. Varga, *Assembling SIP-based VoLTE Call Data Records based on Network Monitoring*, Telecommunication Systems 68, pp. 393–407., 15 p., 2018

[C] Conference and Workshop Papers

[C1] P. Orosz, T. Tothfalusi, *VoicePerf: A Quality Estimation Approach for No-reference IP Voice Traffic*, IEEE/IFIP Network Operations and Management Symposium (NOMS), Budapest, Hungary, 2020

[C2] B. Gyires-Toth, P. Varga, T. Tothfalusi, *Utilizing Deep Learning for Mobile Telecommunications Network Management*, IEEE/IFIP International Symposium on Integrated Network Management, 2019

[C3] B. Nagy, P. Orosz, T. Tothfalusi, L. Kovacs, P. Varga, *Detecting DDoS Attacks within Milliseconds by Using FPGA-based Hardware Acceleration*, IEEE/IFIP Network Operations and Management Symposium (NOMS), Taipei, Taiwan, 2018

[C4] P. Varga, T. Tothfalusi, B. Zsolt, S. Gabor, *Complex Solution for VoLTE Monitoring and Cross-protocol Data Analysis*, IEEE/IFIP Network Operations and Management Symposium (NOMS), Taipei, Taiwan, 2018

[C5] P. Varga, G. Kathareios, A. Mate, R. Clauberg, A. Anghel, P. Orosz, B. Nagy, T. Tothfalusi, L. Kovacs, M. Gusat, *Real-Time Security Services for SDN-based Datacenters*, IEEE 13th International Conference on Network and Service Management (CNSM 2017), Tokyo, Japan, 2017

[C6] P. Varga, D. Kozma, T. Tothfalusi, *Assembling VoLTE CDRs based on Network Monitoring - Challenges with Fragmented Information*, IFIP/IEEE International Symposium on Integrated Network Management, 2017

[C7] P. Varga, L. Kovacs, T. Tothfalusi, P. Orosz, *C-GEP: 100 Gbit/s Capable, FPGA-based, Reconfigurable Networking Equipment*, IEEE 16th International Conference on High Performance Switching and Routing (HPSR), Budapest, Hungary, 2015

[C8] T. Tothfalusi, L. Kovacs, P. Orosz, P. Varga, *100 Gbit/s Network Monitoring with on-the-fly Reconfigurable Rules for Multi-encapsulated Packets*, IEEE 16th International Conference on High Performance Switching and Routing (HPSR), Budapest, Hungary, 2015

[C9] P. Orosz, T. Tothfalusi, P. Varga, *C-GEP: Adaptive Network Management with Reconfigurable Hardware*, IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, Kanada, 2015

[C10] T. Tothfalusi, P. Orosz, P. Varga, *C-GEP: Platform Demo for 100 Gbit/s Network Monitoring*, IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, Kanada, 2015

[C11] T. Tothfalusi, P. Orosz, *Line-rate Packet Processing in Hardware: The Evolution towards 400 Gbit/s*, 9th International Conference on Applied Informatics, Eger, Magyarország, 2014