Voice over New Radio (VoNR) on a Private 5G Network Using SDRs and Open Source Software

Effects of varying the QoS metrics on QoE

Prepared by:
Rasekoai Mokose
MKSRAS001 Department of Electrical Engineering
University of Cape Town

Prepared for:
Joyce Mwangama
Department of Electrical Engineering
University of Cape Town

October 29, 2025

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in **Electrical and Computer Engineering**

Key Words: Voice over New Radio (VoNR), Private 5G Network, Open-Source 5G Core, IP Multimedia Subsystem (IMS), Quality of Experience (QoE), Network Impairments

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report **Voice over New Radio (VoNR) on a Private 5G Network Using SDRs and Open Source Software,** from the work(s) of other people has been attributed and has been cited and referenced. Any section taken from an internet source has been referenced to that source.
3. This report**Voice over New Radio (VoNR) on a Private 5G Network Using SDRs and Open Source Software** is my own work and is in my own words (except where I have attributed it to others).
4. **I have not paid a third party to complete my work on my behalf**. My use of artificial intelligence software has been limited to troubleshoot the errors during the configuration of the files for Core network, IMS and SRSRAN. (**specify precisely how you used AI to assist with this assignment, and then give examples of the prompts you used in your first appendix**).
5. I have not allowed and will not allow anyone to copy my work with the intention of passing it off as his or her own work.
6. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Word count: .....

October 29, 2025

Rasekoai Mokose

Date

Acknowledgments

I would like to extend my heartfelt gratitude to my supervisor, Associate Professor Joyce Mwangama, for her unwavering support and guidance throughout the duration of this project. Her insightful suggestions, clear explanations of technical concepts, and recommendations of relevant materials were invaluable in navigating the complexities of implementing Voice over New Radio (VoNR). Although the IMS component was not successfully implemented due to time constraints and configuration challenges, the process provided a rich learning experience that deepened my understanding of 5G architecture and SIP-based voice services. Professor Mwangama's assistance in troubleshooting IMS, as well as her provision of the necessary equipment for setting up the testbed, played a crucial role in advancing the project and overcoming practical hurdles. Her patience, expertise, and encouragement were instrumental in shaping both the technical and academic aspects of this research, and I am sincerely grateful for her mentorship.

Terms of Reference

1. **Project Title**: Effects of varying the QoS metrics on QoE of Voice over New Radio (VoNR) on a Private 5G Network Using SDRs and Open-Source Software
2. **Background**: With the evolution of mobile networks to 5G and the adoption of IMS (IP Multimedia Subsystem) for voice services, understanding VoNR quality under various network conditions is critical for service providers and network engineers
3. **Objectives**:
    - Deploy a complete 5G testbed, including core network, IMS layer, and radio access network
    - Implement VoNR calling capabilities through IMS integration
    - Assess voice quality under controlled network impairments (latency, jitter, packet loss)
    - Analyze results against ITU-T quality standards and recommendations
    - Document deployment procedures and configuration challenges
4. **Scope**:
    - Build and configure Open5GS 5G core network
    - Deploy Kamailio IMS for VoIP services
    - Implement srsRAN radio access network
    - Configure PyHSS for subscriber management
    - Provision USIM cards with subscriber credentials
    - Conduct VoNR quality testing using ViSQOL algorithm
    - Analyze MOS scores under varying network conditions
    - Document findings and provide recommendations
5. **Resources required**:
    - Hardware: USIM cards, card reader, compatible UE devices,URSP:B210 SDR, antennae
    - Software: Open5GS, Kamailio, srsRAN, PyHSS, SIP clients
    - Tools: Network impairment simulation tools, ViSQOL, WireShark, UHD spectrum analyzer

Abstract

In the field of mobile networks, the most recent technology changing global communications is fifth-generation (5G) wireless. This technology provides great support for services requiring vast data rates, ultra-low latency, and connectivity to many smart devices. One such service is Voice over New Radio (VoNR), which allows voice calls to be made via 5G New Radio access technology. Despite various efforts to improve VoNR, maintaining a high Quality of Experience (QoE) is still a major challenge due to variables such as latency, jitter, packet loss, and others that impair both clarity and reliability.

This study seeks to develop a testbed to investigate VoNR performance utilizing Software Defined Radio (SDR), specifically srsRAN, an open-source 5G core, and Kamailio as the IP Multimedia Subsystem (IMS) server. Wireshark and other similar programs will be used to inspect network traffic and analyze performance. The key metrics under evaluation will be jitter, packet loss, call setup time, and Mean Opinion Score (MOS), among others. The project will also look into ways to improve these parameters, eventually improving the QoE. The findings are expected to apply to private 5G networks, improving QoE on VoNR performance in real-world deployments.

Contents

Chapter 1

Introduction

1.1 Background to the study

This research aims to develop a testbed for transmitting voice calls over the Internet Protocol (IP) with a focus on private 5G networks. In this context, the delivery of voice services is known as Voice Over New Radio (VoNR), which transmits voice as data packets rather than through circuit-switched channels as is done in traditional cellular networks. The experiments are carried out in a private 5G network, a localized system deployed within a campus environment that operates independently of public mobile operators and the wider internet [1]. The project utilizes Software-Defined Radios (SDRs), specifically the USRP B210, to enable transmission and reception of radio signals. The network stack is built on open-source software platforms, including srsRAN and open5GS, which provide freely available and modifiable implementations of 5G RAN and core functions. Within this setup, the study specifically investigates how impairments in network performance metrics, namely, latency, jitter, and packet loss, affect the Quality of Experience (QoE) of VoNR, measured through ViSQOL and Wireshark.

1.2 Objective and Problem statement

1.2.1 Problem Statement

Voice over New Radio (VoNR), the designated voice service for 5G Standalone (SA) architecture, is expected to deliver high-quality, low-latency voice communication across 5G networks. While commercial network operators are actively deploying VoNR in public systems, its performance and behavior within private 5G networks, which are increasingly adopted by enterprises, universities, and research facilities, remain underexplored. Private networks fundamentally differ from their public counterparts in scale, hardware and software configurations, and traffic characteristics. These differences can significantly influence the resulting service quality.

Crucially, VoNR is highly sensitive to network impairments in key performance metrics such as latency, jitter, and packet loss. Any degradation in these metrics directly affects the Quality of Experience (QoE) perceived by the end-users.

1.2.2 Objectives

1. To design and implement a private 5G testbed using srsRAN, open5GS Core, Kamailio IMS, and USRP B210 hardware.

2. To establish VoNR calls over the testbed and obtain baseline QoE measurements under ideal network conditions.

3. To systematically vary latency, jitter, and packet loss individually and in combinations.
4. To measure the impact of these impairments on voice quality using ViSQOL MOS.
5. To analyze and interpret the relationship between KPIs (latency, jitter, packet loss) and QoE outcomes for VoNR.

1.2.3 Purpose of the study

This research aims to investigate the impact of varying metrics, including latency, packet loss, and jitter, on the perceived quality of VoNR in private 5G networks. With systematic variation of these metrics and measuring their respective effect on the QoE with the aid of ViSQOL and Wireshark, this study aims to provide quantitative evidence on the limitations and resilience of VoNR under the various network conditions.

The outcome of this research will show the relationship between varying QoS parameters and the impact the changes have on the QoE, thus allowing the network administrators to troubleshoot and manage the network with ease. They will not have to wait for users to complain about bad call quality; they can monitor QoS metrics and predict when QoE is likely to degrade. The research will also identify the specific technical thresholds where VoNR QoE begins to degrade noticeably, thus providing target values for the metrics, enabling the design and optimization of the network for voice services. As this research uses the open-source software, its success will validate the use of these tools in conducting rigorous QoE testing and, consequently, provide confidence to researchers looking to experiment with 5G private networks and businesses willing to deploy them.

1.3 Scope, limitations and assumptions

1.3.1 Scope

The scope of this study was limited to implementing and evaluating Voice over New Radio (VoNR) on a private 5G Standalone network using Software-Defined Radios (SDRs) and open-source software. The research focused on:

1. Setting up a functional 5G SA testbed with IMS integration.
2. Implementing voice services using SIP signaling as an alternative when IMS integration proved unstable.
3. Using the G.711 μ-law codec for voice transmission.
4. Evaluating voice quality under controlled network impairments (latency, jitter, packet loss) using objective MOS scores.

The study did not include:

1. Wideband codecs such as EVS or Opus
2. Full-scale IMS integration for native VoNR (due to configuration complexity and time constraints).
3. Large-scale performance testing under concurrent call scenarios.

4. Subjective user experience assessments.

.

1.3.2 Limitation

1. The use of a real mobile phone for initiating and terminating calls prevented full automation of the testing process and introduced potential inconsistencies between test runs as it forced continuous play of the audio.

2. Ensuring complete independence between successive tests was challenging, as overlapping audio content may have occurred despite PCAP clearing procedures.

3. The use of Zoiper in testing led to confinement to G.711 μ-law, which does not represent modern wideband voice services

1.3.3 Assumptions

1. The testbed operates in a controlled lab setting, meaning external network traffic, interference, and environmental factors are minimal or negligible.

2. A single SDR and UE setup is assumed to be sufficient for demonstrating VoNR functionality and evaluating basic voice quality metrics.

3. Zoiper is assumed to reliably handle SIP signaling and G.711 μ-law codec without introducing significant artifacts or distortions that would affect MOS scoring.

4. The tools used to introduce latency, jitter, and packet loss are assumed to accurately simulate real-world network conditions and their impact on voice quality.

5. The MOS scores derived from objective tools like ViSQOL are assumed to be valid indicators of perceived voice quality, despite the absence of subjective user feedback.

6. Clearing PCAPs between tests is assumed to remove residual data that could affect subsequent test results, even though some audio overlap may still occur.

1.4 Plan of development

This report is organized into seven chapters, each addressing a specific aspect of the research:

Chapter 1: Introduction Provides the background and motivation for the study, outlines the research problem, objectives, scope, and limitations, and explains the significance of implementing VoNR on a private 5G network.

Chapter 2: Literature Review Examines existing knowledge and technologies relevant to the study. It begins with the evolution of mobile communication, followed by an overview of fifth-generation (5G) networks. The chapter then explores Voice over New Radio (VoNR) and IMS Integration, including IMS call setup flow over 5G/4G networks. It also reviews Objective Speech Quality Assessment methods and related work, concluding with identified research gaps and relevance to this study.

Chapter 3: Methodology Describes the design and implementation of the VoNR testbed, including hardware and software components, network configuration, impairment simulation techniques, and the automated testing framework. It also details the tools used for RTP extraction, MOS calculation, and workflow automation.

Chapter 4: Results and Analysis Presents the findings from single-parameter and combined impairment tests, including latency, jitter, packet loss, and codec behavior. The results are analyzed against ITU-T standards and interpreted to identify key performance thresholds and anomalies.

Chapter 5: Discussion Interprets the results in the context of practical deployment challenges, including IMS integration issues, codec limitations, and methodological constraints. It also compares the findings with industry practices and highlights lessons learned from the testbed approach.

Chapter 6: Conclusions: Summarizes the key findings of the study, highlighting the feasibility of VoNR implementation, critical performance thresholds, and insights gained from the experimental approach.

Chapter 7: Recommendation: Outlines actionable steps for future work, including codec diversity, improved automation, enhanced IMS integration, and scalability testing.

Chapter 2

Literature Review

2.1 Evolution of mobile communication

Mobile communication networks have evolved from the first to the fifth generation, and with the sixth just around the corner, driven by technological advancement and user demand for faster and more reliable services. Earlier generations, from the first to the third, relied on circuit switching to transmit information. This required reserving a dedicated physical channel, also known as a fixed bandwidth, for the duration of the communication. This approach ensured reliable message delivery with minimal, if any, latency, as the channel could not carry any other data until the transmission was complete[2]. Ensuring minimal latency keeps the network highly responsive, thereby providing an excellent user experience and a high Quality of Service (QoS).

However, despite this significant advantage, circuit-switched networks also faced challenges. A key limitation was poor link utilization: while circuit switching achieves very low latency by bypassing routing and arbitration once a circuit is established, the dedicated channel remains reserved for the entire communication session. Even during periods of silence, no other data can use the channel, resulting in inefficient use of network resources[3]. This inefficient bandwidth utilization wasted resources that could have supported higher capacity for users, especially in light of the rapidly growing demand for network services.

This challenge made it difficult to cope with the scaling up of network utilization and therefore prompted the transition to packet-switched networks. In this approach, data is divided into packets, transmitted through multiple routes, and reassembled at the destination node [4]. Packet switching effectively mitigated the issue of poor bandwidth utilization while enabling much higher network capacity. [5]. Packet switching was first introduced in the third generation, where it operated alongside circuit switching, and later became the sole communication method in the fourth and fifth generations. This approach proved highly effective in meeting the growing demand for internet services, particularly by supporting high data rates.

However, challenges arose in the delivery of voice services, which, unlike data, require low latency and minimal jitter to be perceived as high-quality. While early attempts to deliver voice over IP (VoIP) were more cost-effective, their reliance on a shared, best-effort network raised challenges in guaranteeing the Quality of Service (QoS) needed for a reliable and high-quality voice experience. This led to the development of dedicated packet-switched solutions like Voice over LTE (VoLTE), which uses the IP Multimedia Subsystem (IMS) to ensure the strict QoS requirements for voice calls. The final step in this evolution is Voice over New Radio (VoNR), which brings this native packet-switched voice technology entirely into the 5G Standalone network, eliminating the final need for legacy network reliance.

2.2 Fifth Generation (5G) Networks

Currently, mobile communication is primarily driven by the fourth (4G) and fifth (5G) generations, both of which provide significantly improved services, particularly high mobile broadband. 4G succeeded 3G and was the first generation to operate entirely on a packet-switched network. While 4G effectively resolved many issues of earlier generations, its capacity and latency limitations were significant factors, alongside the emergence of new service demands, leading to the development of 5G, which was fundamentally established to address the diverse requirements of various network services, which is reflected in its core design principle, the Service-Based Architecture (SBA).

5G is designed to serve three main categories of use cases through network slicing:

1. Ultra-Reliable Low-Latency Communication (URLLC): For applications requiring extremely high reliability and minimal latency like remote surgery and industrial automation.
2. Enhanced Mobile Broadband (eMBB): For high-speed data access and capacity, like video streaming.
3. Massive Machine Type Communication (mMTC): For connecting a vast number of low-power, low-cost devices like IoT sensors [6]

This dedicated partitioning of the 5G network into distinct slices allows users to leverage the full capability of technology. However, this robust network slicing is only fully realizable when 5G is deployed in the Standalone (SA) architecture[7]. In contrast, the Non-Standalone (NSA) architecture proposed by the 3GPP in Release 15 enabled mobile operators to rapidly deploy 5G by anchoring the control plane to the existing 4G network, thereby mitigating the initial high cost of a full SA deployment.

For a better understanding of these two architectures, it is crucial to recognize that a mobile network fundamentally comprises two main parts: the Radio Access Network (RAN) and the Core Network (CN). The RAN connects the user equipment (the edge) to the network, while the CN manages the services, user sessions, and overall mobility within the network. In the Non-Standalone (NSA) mode, the 3GPP defined several options that represent different combinations of 4G and 5G RAN and CN elements [8] that can be deployed. Most of the existing literature has discussed these options, and they allowed operators to rapidly introduce 5G services.

One of the combinations, NSA Option 3 shown in figure 2.1, involves using the existing 4G Core Network (EPC) to handle the Control Plane functions like signaling and mobility management while the 5G RAN (gNB) is deployed in the Data Plane alongside the 4G RAN, providing enhanced data rates and 5G coverage[9]. Essentially, the 4G Core manages which UE needs the network services and determines its location, while the 5G RAN delivers the data at high-speed.

The subsequent deployment phase is the Standalone (SA) architecture. This rollout involves the 5G Core (5GC) and the 5G RAN operating together, which finally unleashes the full potential of the 5G network, including advanced features like comprehensive network slicing [10] mentioned above.

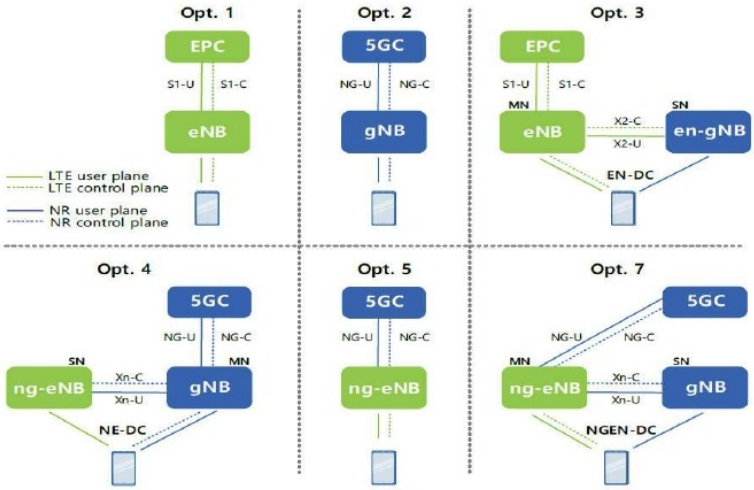


Figure 2.1: Options of 5G deployment[11]

2.3 Voice over New Radio (VoNR) and IMS Integration

While the first three generations prioritized circuit-switched voice services, the focus shifted to data in 3G and beyond, leading to the adoption of packet-switched networks. Voice over LTE (VoLTE) was introduced in 4G to support voice over IP, enabling traditional voice services without fallback to the legacy networks. In NSA 5G deployments, VoLTE remains the primary voice service since the 5G core is not fully utilized. However, VoLTE presents security concerns between the UE and eNodeB [12]. These include vulnerabilities such as denial-of-service (DoS), man-in-the-middle attacks, and poor encryption. Because NSA reuses the LTE EPC for voice, such weaknesses are inherited into the 5G environment, where higher data rates, lower latency, and new spectrum bands amplify the potential impact of attacks [13].

On the radio access side, NSA was enabled by 3GPP Rel-15 Dynamic Spectrum Sharing (DSS), allowing LTE and NR to share spectrum on the same base station. Rel-16 and Rel-17 further improved NR efficiency by enhancing downlink capacity and addressing control channel shortages [14]. While these RAN features strengthened NSA deployments, the reliance on LTE for voice underscores the need to migrate toward VoNR in Standalone 5G, where voice is natively anchored in the 5G core with improved security and reliability. 5G supports network slicing, virtual segmentation of the network into isolated services, which can enhance the security and quality of voice traffic by isolating it from other services[15]. Additionally, VoNR has less time delay and thus provides a better user experience [16].

The 5G Core Network (5GC) manages the data plane and the underlying control plane responsible for connection management. Its key functions include data provisioning, maintaining end-to-end connectivity, handling user mobility, and enforcing Quality of Service (QoS) policies. However, the core network does not handle application-layer signaling. This responsibility lies with the IP Multimedia Subsystem (IMS), which provides the control plane for application and session management. IMS uses SIP signaling for the setup, modification, and termination of voice, video, and multimedia sessions [17].

For user data management and authentication, IMS interfaces with a subscriber repository such as a PyHSS implementation, using the Diameter protocol to perform subscriber data handling and access verification. The cooperation of the 5G core and IMS is illustrated in Figure 2.2.

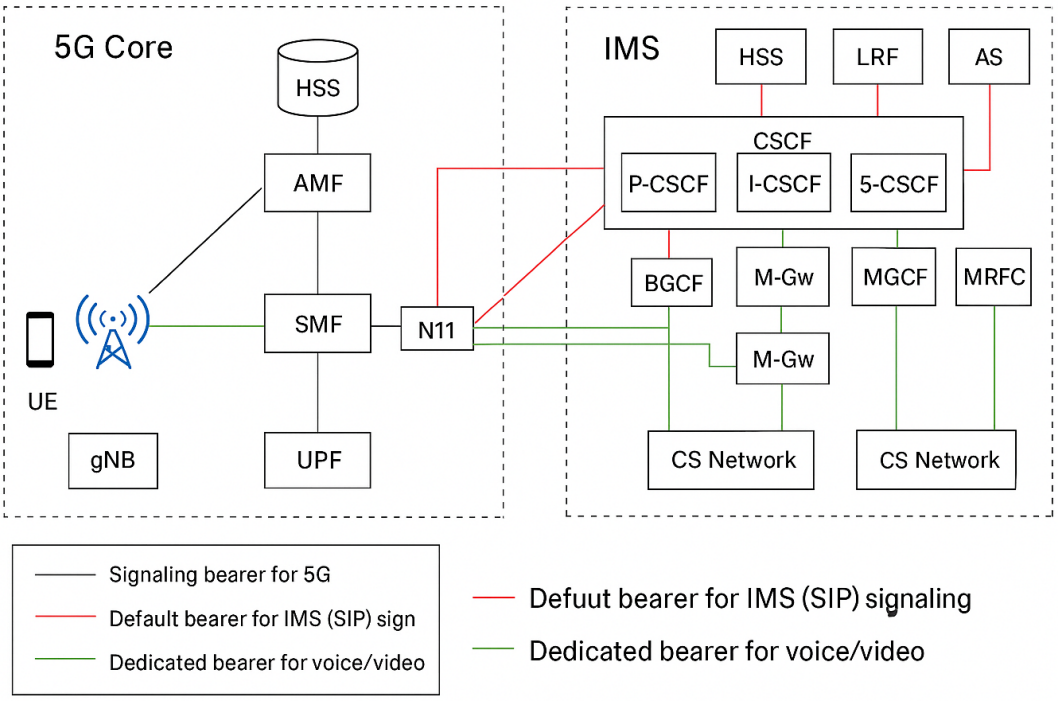


Figure 2.2: Simplified architecture illustrating the interaction between the 5G Core and the IMS during UE registration and voice session setup (image adopted from [18] and generated by ChatGPT)

When a UE powers on, it first connects to the 5G Core Network through the gNB (5G base station) using the Uu interface. The Access and Mobility Management Function (AMF) manages the signaling connection over the N1 interface and authenticates the UE with assistance from the Home Subscriber Server (HSS) via the N2 interface. Once registration and authentication are successful, the Session Management Function (SMF) establishes a PDU session to provide IP connectivity for the UE.

A default bearer is then created for IMS signaling traffic (shown in red in the diagram). Through this bearer, the UE reaches the Proxy-Call Session Control Function (P-CSCF), the entry point to the IP Multimedia Subsystem (IMS).

The IMS registration process begins with the UE sending a SIP REGISTER message to the P-CSCF, which forwards it to the Interrogating-CSCF (I-CSCF). The I-CSCF queries the IMS HSS to identify the appropriate Serving-CSCF (S-CSCF), which ultimately handles the user's SIP registration, authentication (via Diameter with the HSS), and session control.

After successful registration, the S-CSCF maintains the session context for the UE. When the UE initiates a voice call, a dedicated bearer (green in the diagram) is established between the UE and the IMS media gateway (M-Gw) to carry voice or video traffic. This ensures QoS-guaranteed, low-latency

transmission for the ongoing multimedia session.

IMS enables service composition for multimedia applications; however, it faces some challenges with orchestrating these services, which introduces complex feature interaction challenges, where combined services may conflict or behave unpredictably [19].

2.4 IMS call setup flow over 5G/4G networks

The communication flow is organized into three distinct phases that enable different types of IMS services. The process begins with IP connectivity establishment, where the caller first establishes basic IP connectivity with the 5G/4G core network. This foundational step enables data communication over the mobile network infrastructure and is a prerequisite for all subsequent IMS services.

Following connectivity establishment, the IMS service registration phase occurs. The caller initiates registration with the IMS server by sending a *SIP Register* message. The IMS server responds with a *401 Unauthorized* challenge, requesting authentication credentials. The caller then resends the *SIP Register* message with the proper credentials, and upon successful authentication, the IMS server responds with *200 OK*. This registration phase authenticates the user's identity and makes them discoverable within the IMS network for receiving incoming calls and messages. Once registered, the IMS service session establishment phase handles actual communication sessions. Figure 2.3 illustrates two distinct use cases for this phase.

In Case 1, which demonstrates text messaging over IMS, the caller sends a *SIP MESSAGE* request to the IMS server. The server immediately responds with *202 Accepted*, indicating it has received the message and taken responsibility for delivering it. The IMS server then forwards the *SIP MESSAGE* to the callee, who responds with *200 OK* to confirm successful receipt.

Case 2 demonstrates voice or video call establishment over IMS, which involves a more complex signaling exchange. The caller initiates the session by sending a *SIP INVITE* request to the IMS server, which forwards it to the callee. The IMS server responds with *100 Trying* as a provisional response to indicate the request has been received and is being processed. As the call setup progresses, Session Progress messages flow between the parties. When the callee's device begins alerting the user, *180 Ringing* responses are sent back through the IMS server to the caller, providing feedback that the destination is reachable and ringing. When the callee accepts the call, a *200 OK* response travels back through the same path. Once this signaling exchange completes successfully, the actual voice or video conversation begins using RTP and RTCP packets for real-time media transmission.

This architecture demonstrates how IMS enables rich communication services, including voice calls, video calls, and instant messaging over IP-based mobile networks using standardized SIP signaling protocols, with the IMS server acting as the central control point for session management and routing.

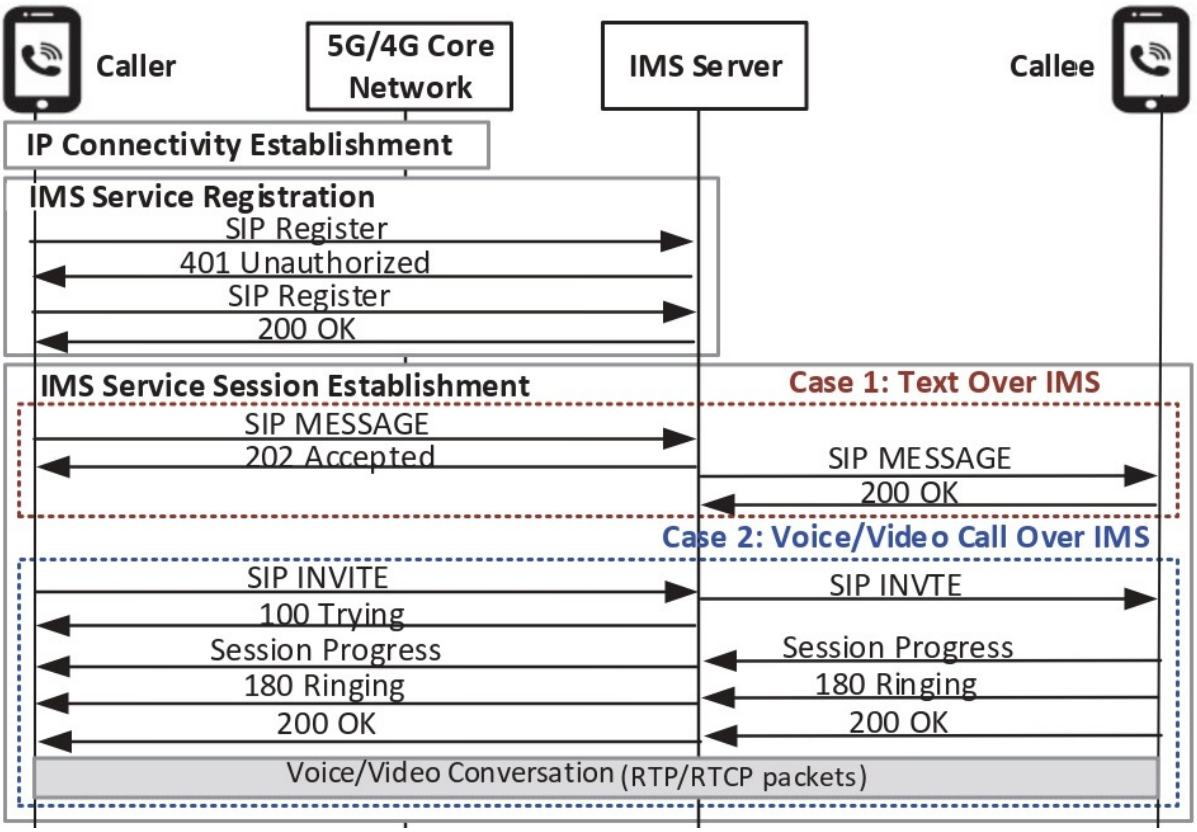


Figure 2.3: Illustrates the bidirectional exchange of control messages that enable voice communication between two UEs through the 5G Core and the IMS server[17].

2.5 Objective Speech Quality Assessment

For decades, objective speech quality assessment has relied on ITU-T standards such as PESQ (P.862) and its successor, POLQA (P.863). PESQ was originally developed for narrowband speech (300–3400 Hz), while POLQA extended this range to wideband (50–7000 Hz) and super-wideband (up to 14 kHz), addressing many of PESQ's limitations in modeling modern communication systems [20].Figure 2.4 illustrates the evolution of ITU-T's perceptual speech quality assessment standards from PSQM (P.861) to POLQA (P.863), showing how bandwidth support and network applicability expanded alongside telecommunications technologies.

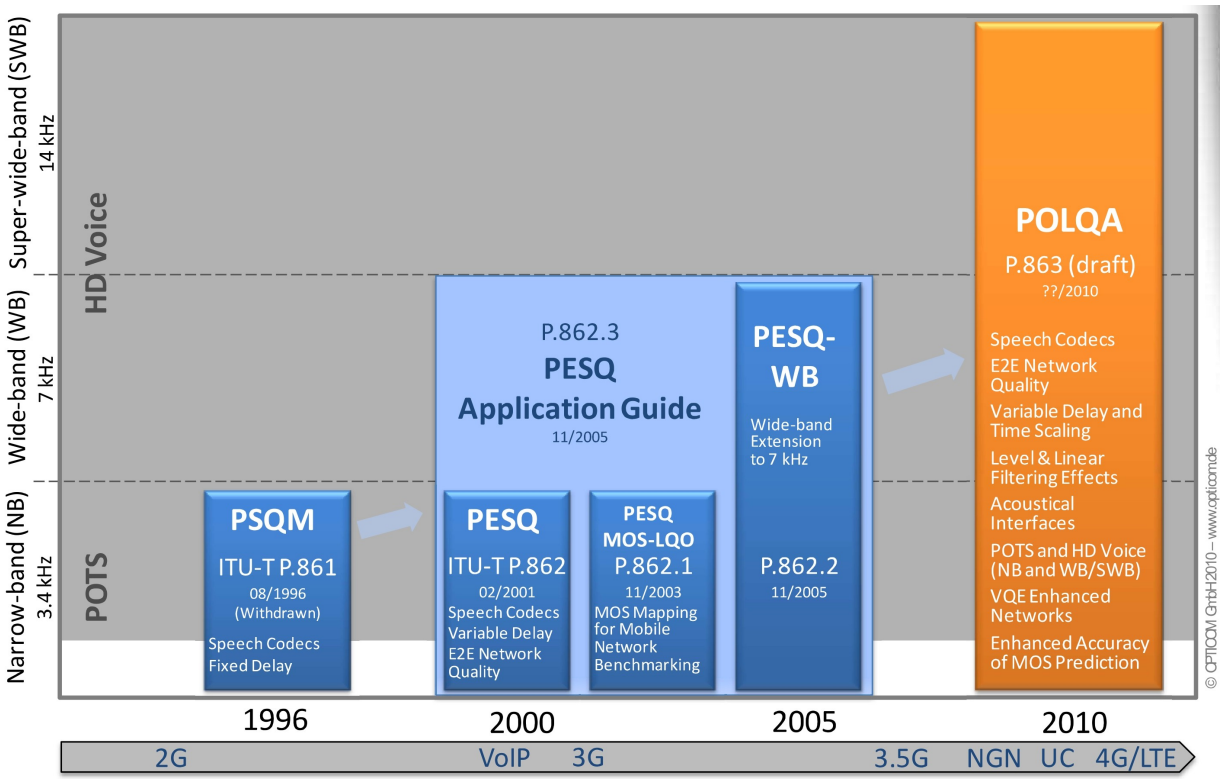


Figure 2.4: Evolution of ITU-T voice quality assessment standards (P.86x series) from PSQM to POLQA adopted from [21].

BESQ and POLQA have become the industry benchmarks for quantifying the perceptual impact of codecs, transmission impairments, and network conditions on speech quality. However, their proprietary and licensed nature often restricts use in open academic research.

In contrast, ViSQOL provides an open-source alternative that estimates perceptual quality using spectro-temporal similarity metrics. It has demonstrated performance comparable to PESQ and POLQA under a range of degradations, including noise, jitter, packet loss, and clock drift, while offering greater robustness to time-warping artifacts [22]. The latest version, ViSQOL v3, offers a refined C++ implementation and is used in real-world applications such as Google Meet and Opus codec evaluations [23].

2.6 Related work

2.6.1 Migration to VoNR

Previous studies on voice quality in mobile networks have primarily focused on VoIP and VoLTE technologies. VoLTE, which uses IMS and AMR/EVS codecs, has been widely evaluated for robustness under varying network conditions. For example, Cipressi [24] compared AMR-WB and AMR-NB using VQmon, while Prasad [25] proposed adaptive bitrate mechanisms to improve VoLTE quality. More recent work has shifted towards VoNR, the 5G-native voice solution. He et al. [26] conducted a comparative study of VoNR and VoLTE using POLQA-based MOS scoring under different network impairments and channel models. Their results showed that VoNR consistently outperformed VoLTE in

voice quality, especially under degraded conditions, due to its higher bandwidth, better fault tolerance, and advanced EVS codecs. Other studies have explored VoNR's advantages in power efficiency and latency, but few have focused on real-time quality assessment using open-source platforms, motivating the use of tools like Open5GS, srsRAN, and Kamailio to evaluate VoNR performance using a software-defined radio and commercial off-the-shelf device

2.6.2 Leveraging the open-source software

The softwarization and virtualization of mobile networks have spurred the development of open-source 5G cellular network components, enabling researchers to configure and deploy low-cost, flexible, and private 4G and 5G testbeds using Commercial-Off-The-Shelf (COTS) equipment. This approach helps overcome the financial and technical rigidity associated with proprietary vendor-specific infrastructure [7].

The open and modular 5G Standalone (SA) architecture has led to the emergence of popular open-source implementations across the key network domains. For the Radio Access Network (RAN), leading projects include srsRAN Project and OpenAirInterface 5G RAN (OAI), which implement the gNodeB functionality, often utilizing Software Defined Radios (SDRs). The comparative performance of these two stacks in end-to-end 5G SA platforms has been analyzed by Amini et al. [9] and Chepkoech et al. [27], with the latter evaluating OAI's good latency performance in NSA mode. Similarly, in the 5G Core Network (5GC), frameworks such as Open5GS and OAI5GC adhere to 3GPP standards.

A significant body of work has focused on studying the interoperability and comparative performance of various combinations of these open-source RAN and 5GC stacks in end-to-end testbeds, evaluating primarily data-centric metrics like throughput and latency. However, the initial focus of these open-source testbeds was largely limited to demonstrating basic data connectivity and non-voice applications, leaving the full integration and objective quality assessment of the complex Voice over New Radio (VoNR) and its required IP Multimedia Subsystem (IMS) layer relatively unexplored in these open-source environments.

2.6.3 Realisation of VoNR with open-source tools

Building upon this foundation of research, Modroiu et al. [28] took a crucial step in 2024 by presenting the world's first fully open-source VoNR prototype integrated into a 5G Standalone (SA) private network, which was demonstrated at the Open Source Mobile Network Technologies Conference 2024. Their implementation utilized srsRAN for the radio access network, Open5GS for the 5G core, and Kamailio with PyHSS for the IMS platform, successfully deploying IMS services over the standardized N5 interface. The initial work focused on prototype deployment and validation, measuring network-level KPIs including call setup time, jitter, and packet loss, while demonstrating the time required for QoS resource reservation in VoNR calls, approximately 825 ms for dedicated bearer establishment. Their extended study provided a comparative evaluation between VoNR and OTT voice services under congested and uncongested network conditions using a 40 MHz bandwidth deployment at 3.75 GHz. In their findings, both services remained stable with acceptable audio quality; VoNR exhibited slightly higher maximum jitter and unexpectedly higher packet loss in congested scenarios, contrary to expectations for QoS-enabled services. Additionally, VoNR call setup time increased significantly

under congestion. These results are shown in figure 2.5. The authors acknowledged limitations in their subjective evaluation of audio quality and identified the need for objective evaluation of QoE, investigation of unexpected packet loss behavior, and testing with various radio hardware configurations.

Setup	PacketLoss	MeanJitter	CallSetupTime
	%	ms	s
VoNR Congestion	2.59	11.24	1.66
VoNR No-congestion	0.03	9.24	0.95
OTT Congestion	2	12.11	0.16
OTT No-Congestion	0.04	11.06	0.10

Figure 2.5: Result of network performance metrics for open-source VoNR voice service obtained by [28].

2.7 Research Gaps

Despite extensive literature on VoNR's theoretical advantages and 5G network performance, critical gaps remain in understanding voice service quality in private 5G deployments using open-source platforms. While Modroiu et al.[28] demonstrated the first fully open-source VoNR prototype, their evaluation focused primarily on network-level KPIs under congested conditions. The authors acknowledged the need for objective QoE assessment using standardized MOS metrics and identified unexpected behaviors, such as higher VoNR packet loss compared to OTT services, requiring further investigation. Additionally, most VoNR quality assessments rely on proprietary tools like POLQA, limiting accessibility for academic research. The application of open-source alternatives like ViSQOL for systematic quality assessment under controlled impairments remains underexplored in private 5G contexts. Furthermore, empirical data on how individual network impairments—latency, jitter, and packet loss—affect voice quality when systematically varied in isolation and combination is limited. Understanding these relationships is essential for establishing practical thresholds for proactive network management. Finally, practical challenges of integrating IMS with open-source 5G cores are poorly documented, with most literature focusing on successful implementations rather than troubleshooting processes and alternative approaches.

2.8 Relevance of this Study

This research addresses these gaps through systematic, empirical VoNR quality evaluation using open-source tools in a private 5G SA testbed. Key contributions include:

- Validating ViSQOL as a cost-effective alternative to proprietary quality assessment tools.
- Establishing quantitative relationships between network impairments and voice quality through controlled testing.
- Enabling proactive network management by linking observable QoS metrics to predicted QoE degradation.
- Building confidence in open-source 5G platforms (srsRAN, Open5GS, Kamailio, PyHSS) for academic and enterprise deployments.

The study bridges the gap between theoretical VoNR capabilities and practical deployment realities in resource-constrained private network environments.

Chapter 3

Methodology

3.1 Research Design

The 5G testbed was implemented in a distributed fashion, where the gNB and the Core Network/IMS were deployed on separate PCs. The USRP B210 was incorporated to provide transmission and reception of the radio signals, and the OnePlus 11 5G and Google Pixel 7 COTS phones were used as the UEs. This topology was chosen because it closely resembles real-world network topologies, where the RAN and CN are physically and logically isolated. It enabled focused evaluation of Key Performance Indicators (KPIs), such as latency and jitter, introduced by network distribution.

3.2 System Model

The key network components for the testbed were User Equipment (UE), consisting of two 5G-compatible mobile phones and a programmable Subscriber Identity Module (SIM), the gNB, the base station that provided the radio communication between the UE and the 5G Core Network, Software Defined Radio (SDR) B210, which enabled the transmission and reception of radio signals, Kamailio, which functioned as a Session Initiation Protocol (SIP) server for Voice over New Radio (VoNR), and finally, the 5G Core Network (CN), which provided essential services including the Access and Mobility Management Function (AMF), Session Management Function (SMF), User Plane Function (UPF), and others.

The testbed was developed with two desktop PCs, one hosting the gNB, while the other PC hosted the Open5GS CN and IMS framework, which included the Kamailio SIP server as part of it. The COTS phones were used as the UEs, while URSP B210 was used as the radio front-end, connected via USB 3.0 to the PC hosting the gNB. The two PCs were connected to the same LAN: the gNB-hosting PC was on IP address 137.158.126.41, while the CN-hosting PC was on IP address 137.158.127.114. Figure 3.1 provides a visual overview of the testbed topology, while Table 3.2 details the specifications of the hardware used.

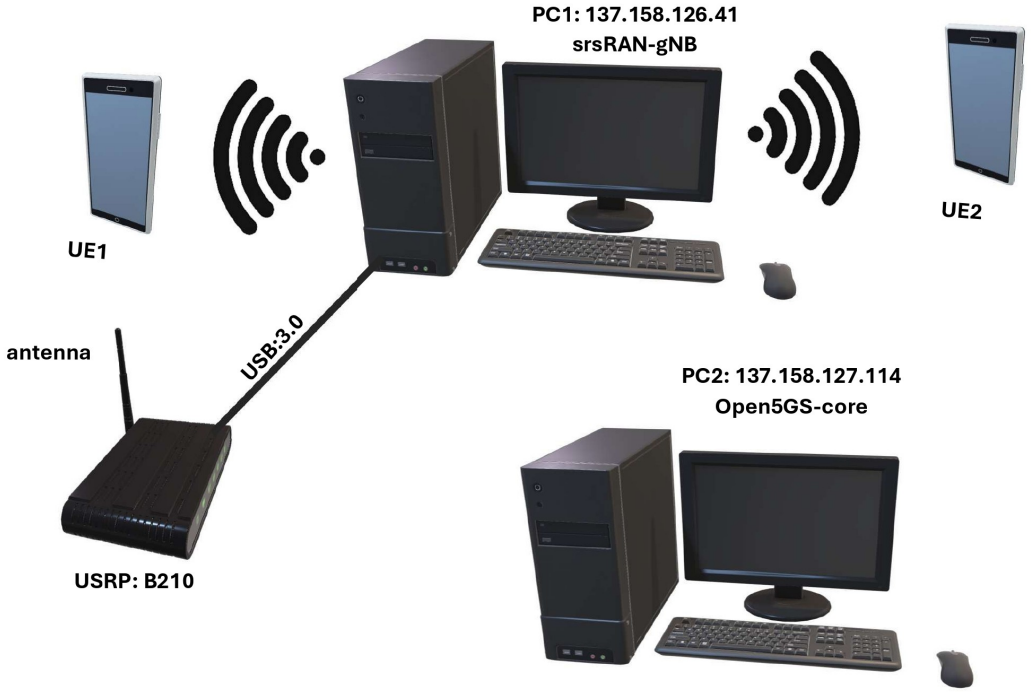


Figure 3.1: 5G Testbed using SDR and Open-source software

3.2.1 Device Specification

Table 3.1: Specification of the hardware used

Device Name	Description
PC1 and PC2	Hardware Model:ASUS, OS:ubuntu 22.04.5LTS, OS type: 64bit, Graphics:NV138, Disk Capacity:500.1GB, GNOME Version:42.9
Software Defined Radio	Brand: USRP B210, Frequency range:70MHz-6GHz, Frequency:3.4GHz Tx output Pmax:+20dBm, Rx input Pmax:-15dBm, Ports resistance:50Ω
User Equipment	Type: OnePlus-11 5G, Google Pixel7
Antennae	Brand: Poynting, impedandance:50Ω,Frequency range: 698MHz-3.8GHz

3.3 Deployment of testbed components

The testbed was constructed using three main components: the Open5GS core network, Kamailio IMS, and srsRAN gNB.

3.3.1 Open5GS Core Network

The 5G core network was built using the Open5GS project from the official GitHub repository open5gs. Following the project's documentation, the core network functions were deployed, including AMF, SMF, UPF, and other essential network elements. Configuration files were modified to define network parameters, PLMN settings, and subscriber information.

3.3.2 srsRAN gNB

The radio access network was built using srsRAN from its GitHub repository, srsRAN. The setup included configuring the base station, gNodeB, which involved careful configuration of files that were adjusted to match the core network parameters, including frequency bands, PLMN IDs, and network slicing parameters.

3.3.3 SIM Card Provisioning

SIM card provisioning was performed using a Rocketek smart card reader connected via USB, which supports ISO 7816 protocol for reading and writing SIM/USIM cards. The programming software used was pySim-shell pysim, an open-source tool for modifying sysmocom SIM card parameters such as IMSI, Ki, and OPC. The process involved inserting programmable USIM cards into the Rocketek reader and using pySim commands to write subscriber parameters. These programmed values were then registered in PyHSS to enable proper subscriber authentication across the network and IMS infrastructure. The ADM (Administrative) keys specific to each SIM card were required to unlock write access during the provisioning process.

3.3.4 Kamailio IMS

The IMS layer was implemented using Kamailio, an open-source SIP server. The deployment followed the Kamailio IMS configuration guidelines available on kamailio, integrating it with the Open5GS core to enable VoIP services. This involved configuring the P-CSCF, I-CSCF, and S-CSCF functions, along with setting up the appropriate SIP routing rules and database backends for subscriber management.

3.3.5 PyHSS

The Home Subscriber Server (HSS) was implemented using PyHSS, an open-source HSS/HLR solution written in Python. Deployed from the official GitHub repository pyhss.The setup involved configuring the database backend to store subscriber credentials, service profiles, and IMS-specific parameters such as authentication vectors and S-CSCF assignments. PyHSS was integrated with both Open5GS and Kamailio IMS through the Diameter protocol, enabling centralized subscriber management across the mobile network and VoIP services. Configuration files were modified to establish Diameter connections and define subscriber data, including IMSI, Ki values, and IMS public/private identities.

3.4 QoE Metrics and Measurement

The investigation focused on assessing the impact of three critical Network Performance Indicators (NPIs) on the end-user Quality of Experience (QoE): latency, jitter, and packet loss.

To determine the individual sensitivity and specific degradation thresholds for VoNR quality attributable to each metric independently, test scenarios were systematically conducted with the following varying parameters:

Table 3.2: Specification of the hardware used

Metric	**Range**	**Step size**
Latency(ms)	0 - 200	50
Jitter(ms)	0 - 100	25
Packet loss (%)	0 - 20	5

For each unique combination of these parameters (scenario), several trials were executed, resulting in the generation of corresponding audio files for subsequent objective analysis. This methodology ensured a robust dataset for determining the precise thresholds at which network distribution impairs VoNR service quality.

3.5 System Architecture and Test Environment

The VoNR quality testing framework was implemented on a private 5G SA network infrastructure consisting of:

1. **Core Network**: Open5GS providing 5G core network functions (AMF, SMF, UPF, etc.)
2. **RAN**: srsRAN Project implementing gNB functionality
3. **User Equipment**: Commercial 5G-capable mobile devices
4. **Media Processing**: RTPEngine for real-time RTP/RTCP packet processing and capture.

3.6 Test Scenarios

The research employed a systematic approach to investigate the impact of network impairments on VoNR quality:

3.6.1 Baseline Test

Here it was conducted under ideal network conditions with zero latency, jitter, and packet loss applied. Reference audio was transmitted over a VoNR call from the sending User Equipment (UE) to the receiving UE, with all RTP traffic captured. The audio captured during this test served as the unimpaired reference for all subsequent comparative analysis with the ViSQOL algorithm.

3.6.2 Single-Parameter Tests

The baseline test was followed by a single-parameter test where impairment scenarios involved isolating the effect of each degradation metric independently. Only one parameter (latency, jitter, or packet loss)

was varied across its defined range, while the remaining parameters were maintained at zero-impairment values.

3.6.3 Combined-parameter test

In this test, the parameters were varied within their respective defined ranges and combined such that no parameter remained the same.

3.6.4 Two-parameter Test

The last test was carried out by changing two parameters at the time while the third remained constant

3.7 Network Impairment Implementation

Network impairments were applied using Linux Traffic Control (tc) with the netem (Network Emulator) module at the User Plane Function (UPF) tunnel interface (ogstun) using the following commands to apply the impairments:

1. Packet loss: *tc qdisc add dev ogstun root netem loss X%*
2. Jitter: *tc qdisc add dev ogstun root netem delay 50ms Xms distribution normal.*
3. Latency: *tc qdisc add dev ogstun root netem delay Xms*

3.8 Extraction of the audio file

A reference audio file compliant with the ITU-T P.862 (1999) standard was played continuously from a phone into the microphone of the calling phone while the receiving phone accepted the call. An RTPEngine was configured to perform continuous packet capture (PCAP) of all RTP streams during active voice calls, storing PCAP files in /var/spool/rtpengine/pcaps/. This allowed a script to be run and automate the process. RTP audio was extracted from PCAP files using a multi-stage pipeline. First, tshark identified UDP ports carrying RTP traffic. Then, RTP streams were isolated using SSRC identifiers. The payloads were extracted and converted from hexadecimal to raw audio. Finally, G.711 codec-encoded audio was decoded at 8 kHz using FFmpeg and saved as a WAV file.

3.9 Data Collection and Analysis

A Python-based automated testing framework was developed to systematically simulate the defined scenarios. For each scenario, network impairment was applied via *tc netem* multiple times, and audio files were automatically extracted and stored.

3.10 Objective Quality Assessment and plotting

ViSQOL was used to measure perceptual voice quality using a TensorFlow Lite model in speech mode for narrowband audio (8 kHz). Each impaired audio sample was compared to a baseline audio to calculate a MOS-LQO score on a 1–5 scale, where higher scores indicate better perceived quality.

Each test produced individual WAV files, a CSV file with MOS scores and test conditions, and the corresponding plots.

Chapter 4

Results

4.1 Testbed Implementation

4.1.1 Base station

The 5G base station was implemented using the srsRAN gNB software on an Ubuntu 22.04 desktop, paired with an Ettus USRP B210 SDR as the RF front-end. The gNB provided the full 5G NR protocol stack, managing radio resources and UE connectivity, while the USRP handled transmission and reception at 3.5 GHz (n78 band) over a 20 MHz channel. The SDR connected via USB 3.0 and operated with Tx/Rx gains of 60 dB and 50 dB. Communication with the 5G Core occurred through the N2 (control plane) and N3 (user plane) interfaces.

Figure 4.1 shows the desktop PC running srsRAN to implement the gNB, alongside a screenshot of the gNB initialization details. Two antennas are connected to the USRP B210 SDR, positioned in vertical and horizontal orientations, one mounted on top of the PC chassis and the other on the side. The USRP B210 is placed next to the PC, serving as the RF front-end for 5G transmission.

Figure 4.1: 5G SA base station physical setup

When the gNB was initialized, the logs on the Core AMF were displayed as shown in Figure 4.2.

```
bluelabuser@bluelabuser-System-Product-Name:/etc/open5gs$ journalctl -u open5gs-amfd -f
Sep 25 14:58:57 bluelabuser-System-Product-Name open5gs-amfd[539853]: 09/25 14:58:57.636: [sbi] INFO: Setup NF EndPoint(addr) [137.158.127.114:7774] (.
./lib/sbi/nnrf-handler.c:955)
Sep 25 14:58:57 bluelabuser-System-Product-Name open5gs-amfd[539853]: 09/25 14:58:57.636: [sbi] INFO: [661559a8-9a0f-41f0-b7c1-db85bc121e5c] Subscripti
on created until 2025-09-26T14:58:57.634801+02:00 [duration:86400000000,validity:86400.000000,patch:43200.000000] (../lib/sbi/nnrf-handler.c:874)
Sep 25 14:58:57 bluelabuser-System-Product-Name open5gs-amfd[539853]: 09/25 14:58:57.637: [sbi] INFO: Setup NF EndPoint(addr) [137.158.127.114:7774] (.
./lib/sbi/nnrf-handler.c:955)
Sep 25 14:58:57 bluelabuser-System-Product-Name open5gs-amfd[539853]: 09/25 14:58:57.637: [sbi] INFO: [6615697a-9a0f-41f0-b7c1-db85bc121e5c] Subscripti
on created until 2025-09-26T14:58:57.635214+02:00 [duration:86400000000,validity:86400.000000,patch:43200.000000] (../lib/sbi/nnrf-handler.c:874)
Sep 25 15:07:52 bluelabuser-System-Product-Name open5gs-amfd[539853]: 09/25 15:07:52.395: [amf] INFO: gNB-N2 accepted[137.158.126.60]:38325 in ng-path
module (../src/amf/ngap-sctp.c:113)
Sep 25 15:07:52 bluelabuser-System-Product-Name open5gs-amfd[539853]: 09/25 15:07:52.395: [amf] INFO: gNB-N2 accepted[137.158.126.60] in master_sm modu
le (../src/amf/amf-sm.c:894)
Sep 25 15:07:52 bluelabuser-System-Product-Name open5gs-amfd[539853]: 09/25 15:07:52.403: [amf] INFO: [Added] Number of gNBs is now 1 (../src/amf/conte
xt.c:1277)
```

Figure 4.2: AMF logs after gnb initialization

The gNB signal was analyzed using UHD FFT on the USRP B210. The plot in figure 4.3 shows a carrier at 3.5 GHz (n78 band) with a peak near -65 dB with the sampling rate of 30.72 MS/s.

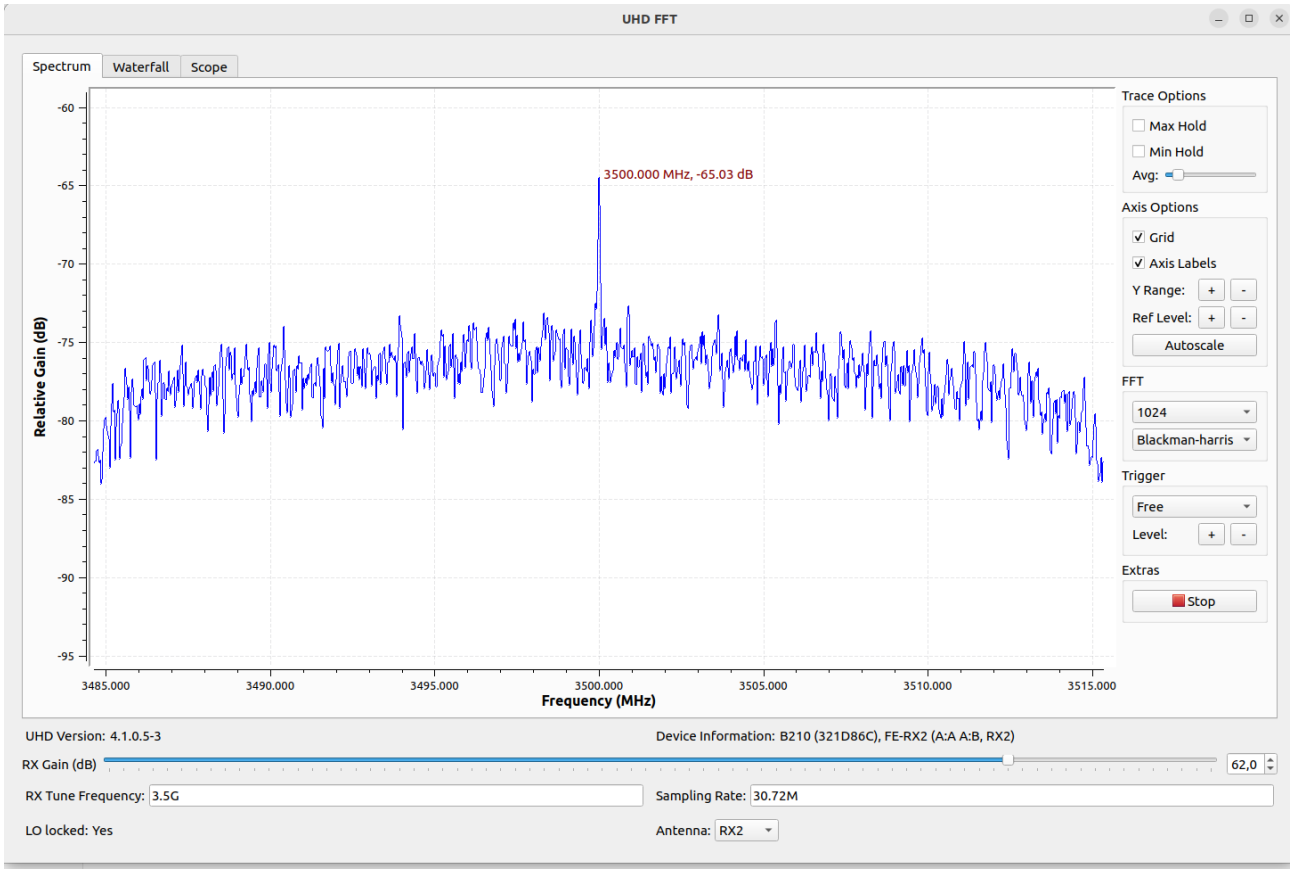


Figure 4.3: The UHD FFT plot for gNB

4.1.2 5G Core Network (5GC)

The core network was hosted on a separate desktop computer 4.4 with similar features to the one used for the base station. It was implemented using open5GS, an open-source software suite that provides all essential 5G Standalone (SA) core network functions. The statuses of the essential services of the core are shown in figure 4.7

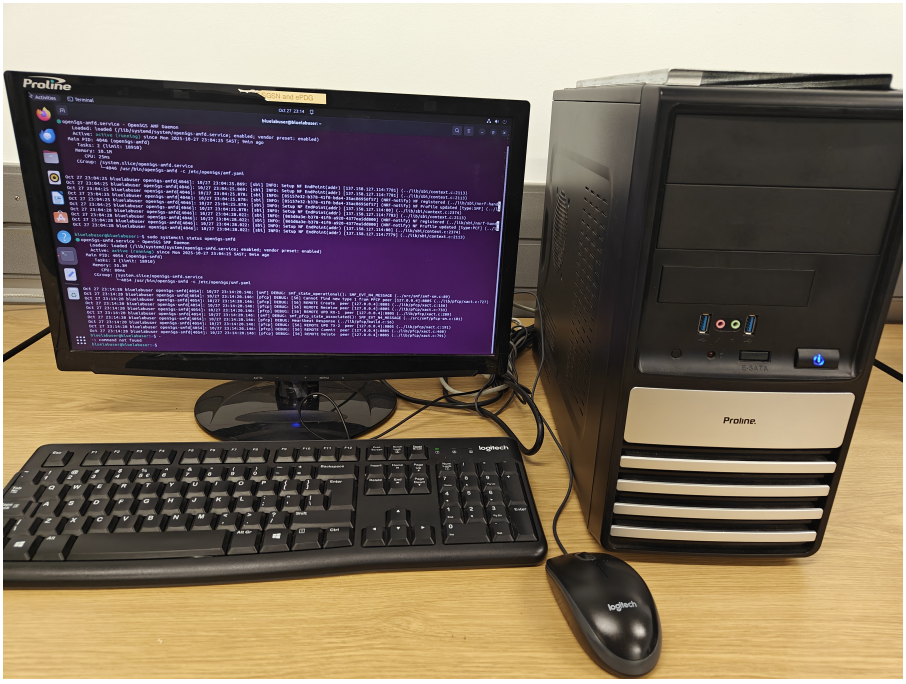
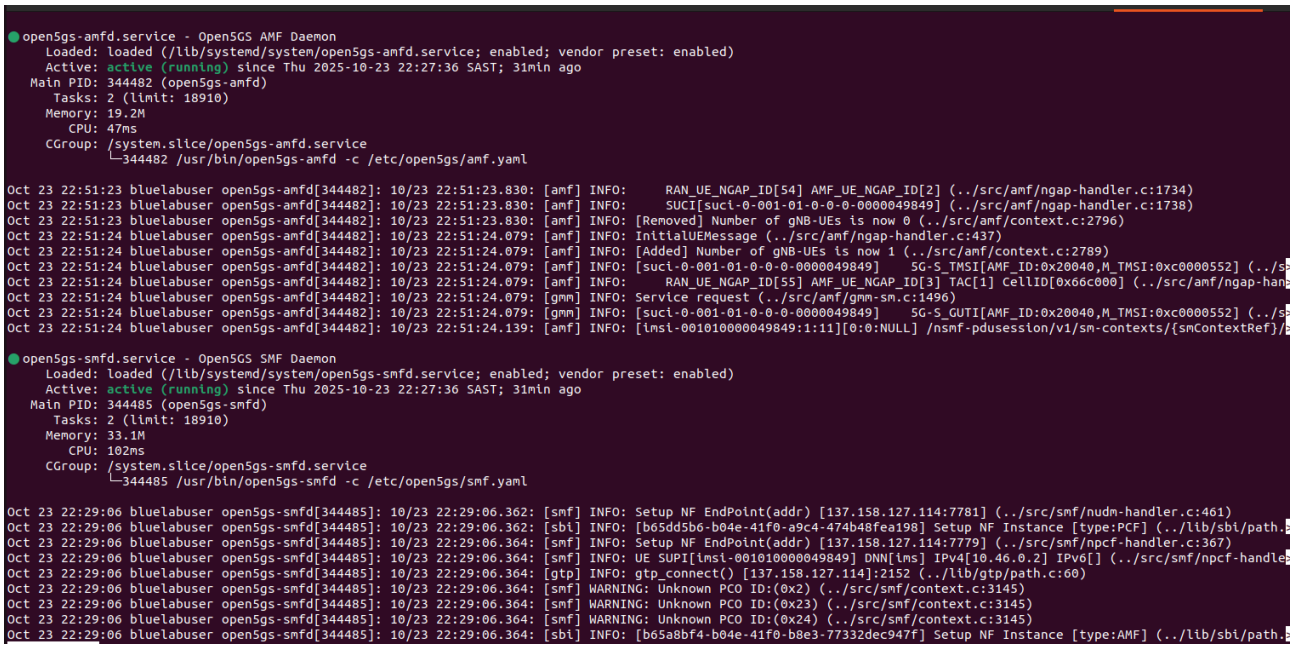
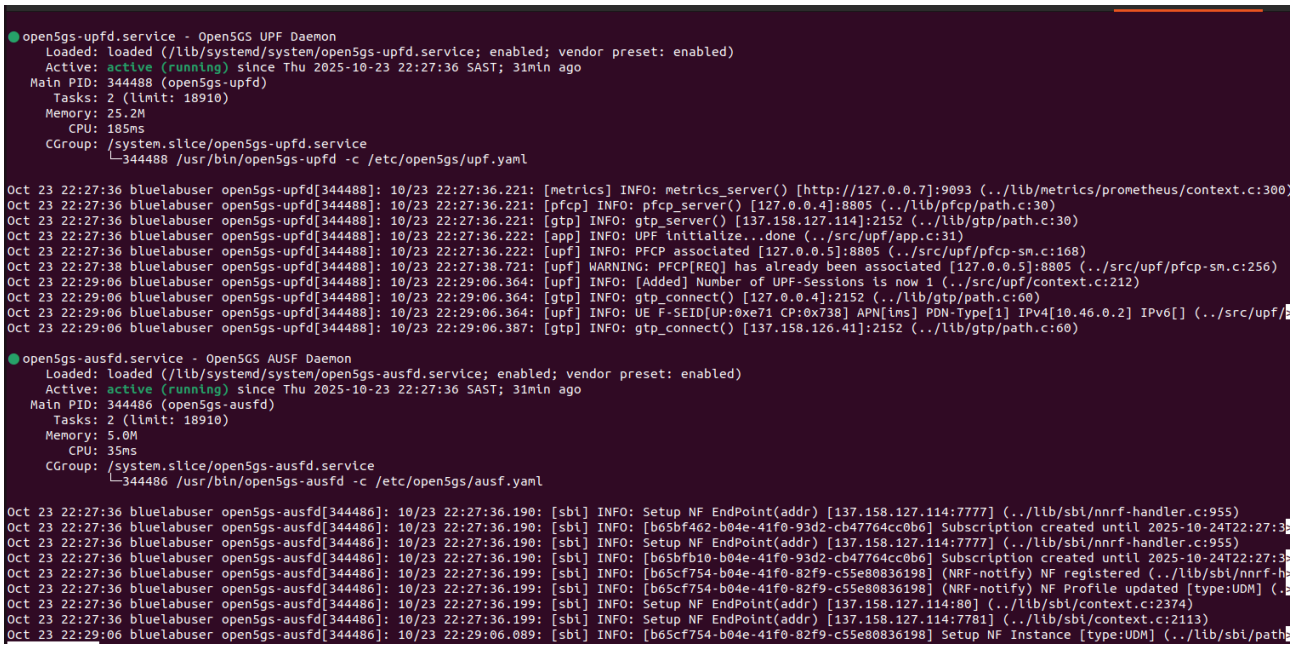


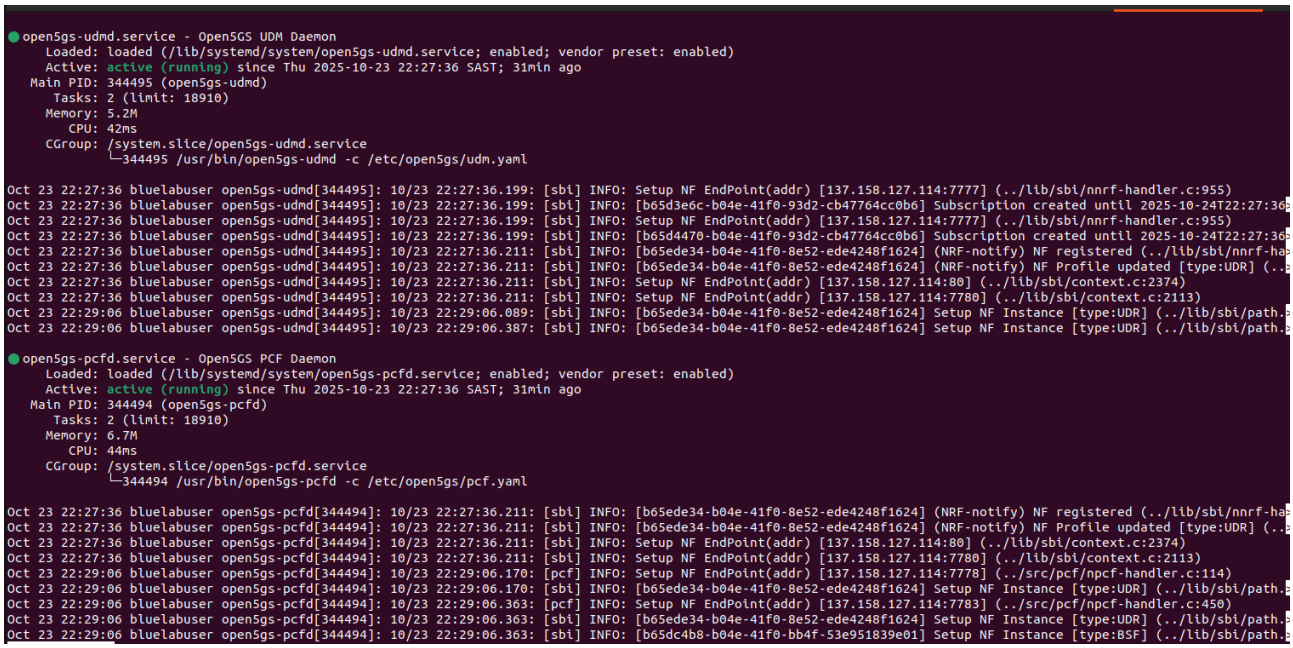
Figure 4.4: PC that hosted the open5gs core, IMS, and HSS database



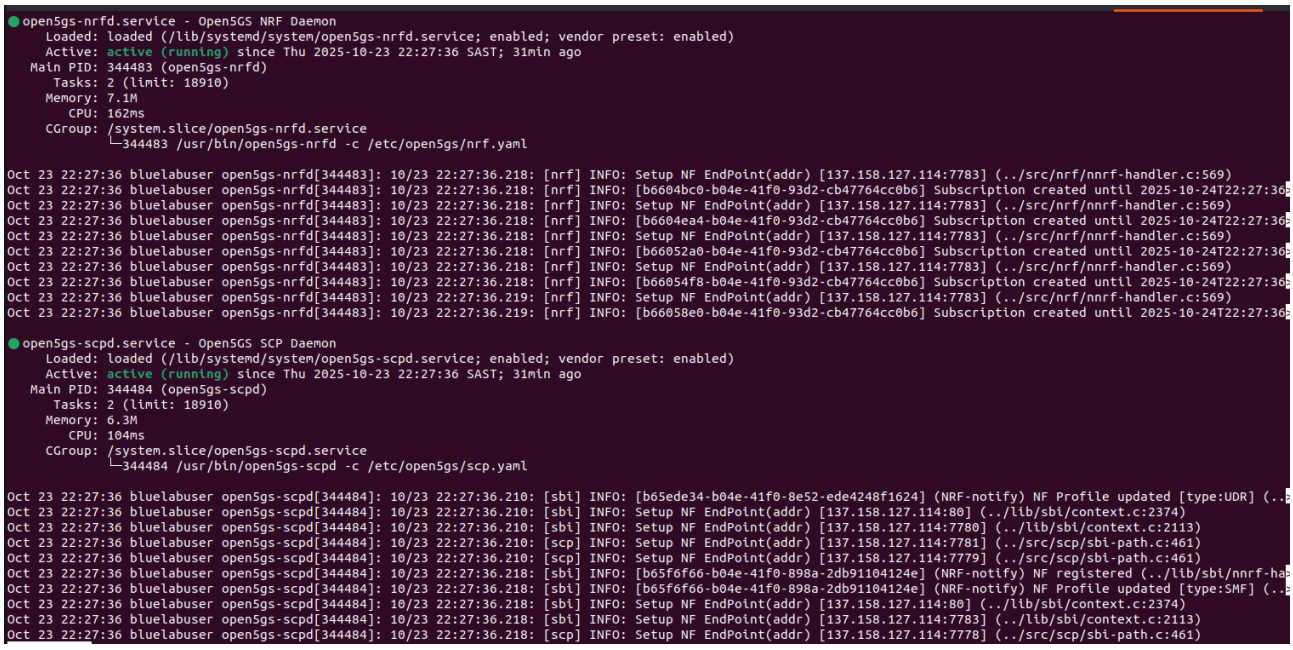
(a) Status of SMF and AMF core services



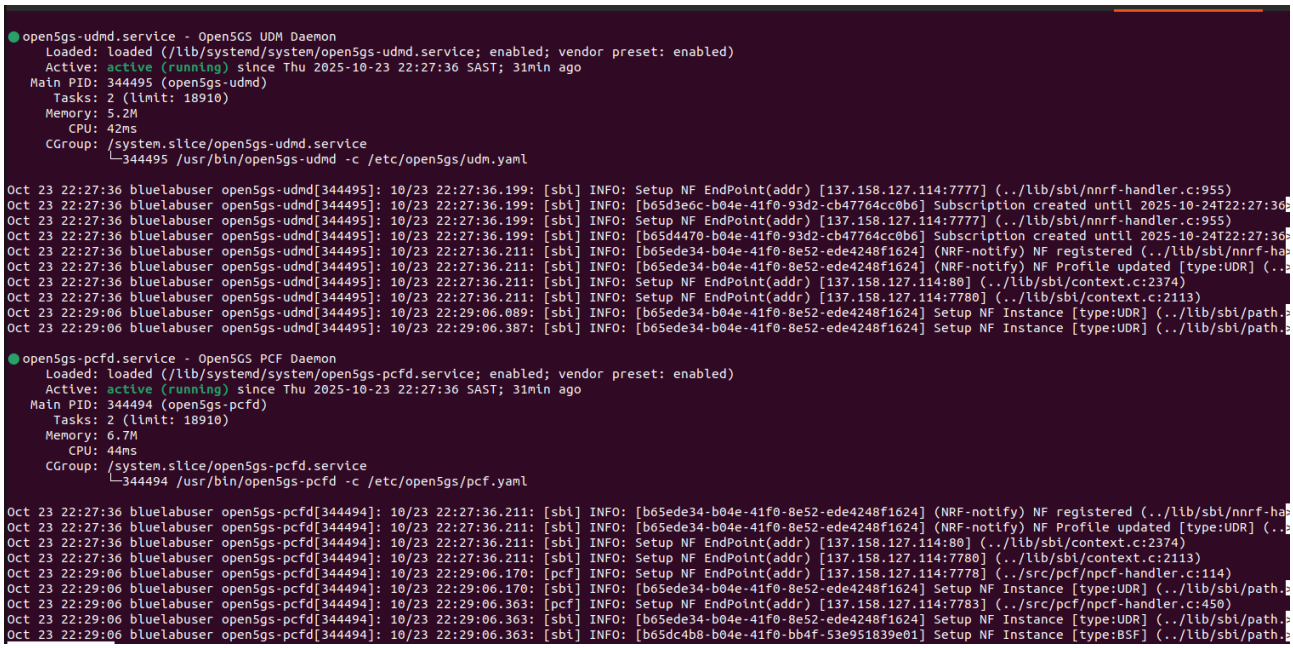
(b) Status of UPF and AUF core services



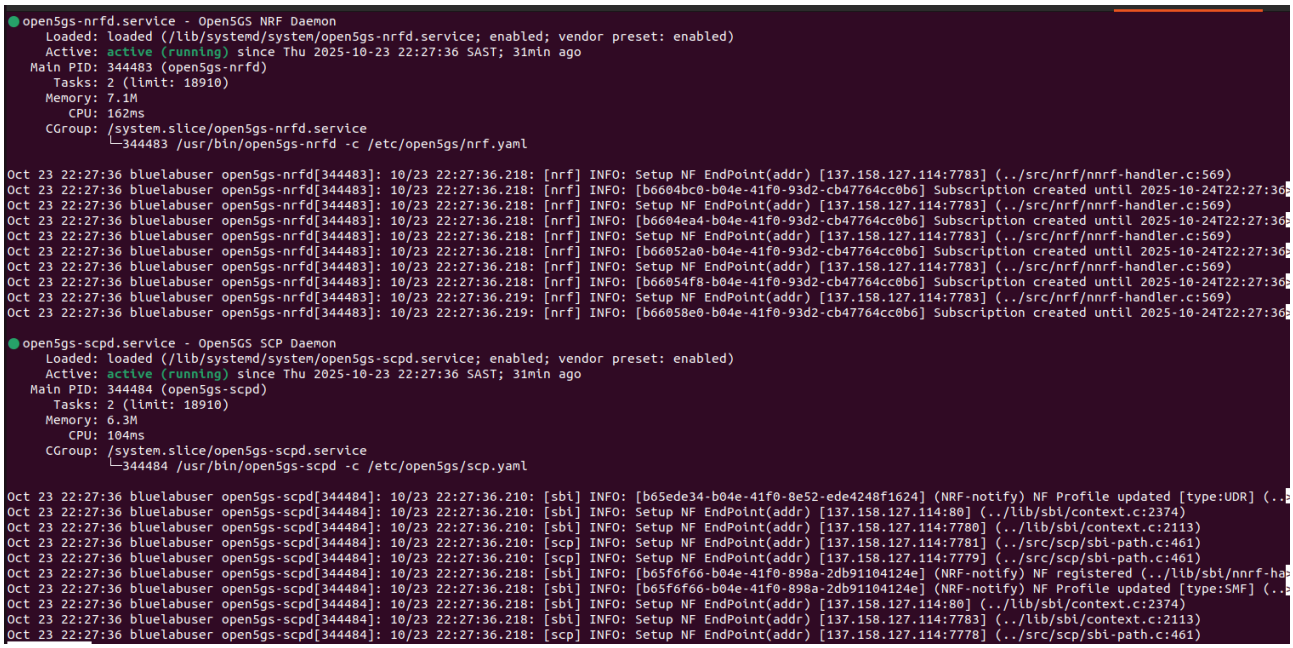
(a) Status of UDM and PCF core services



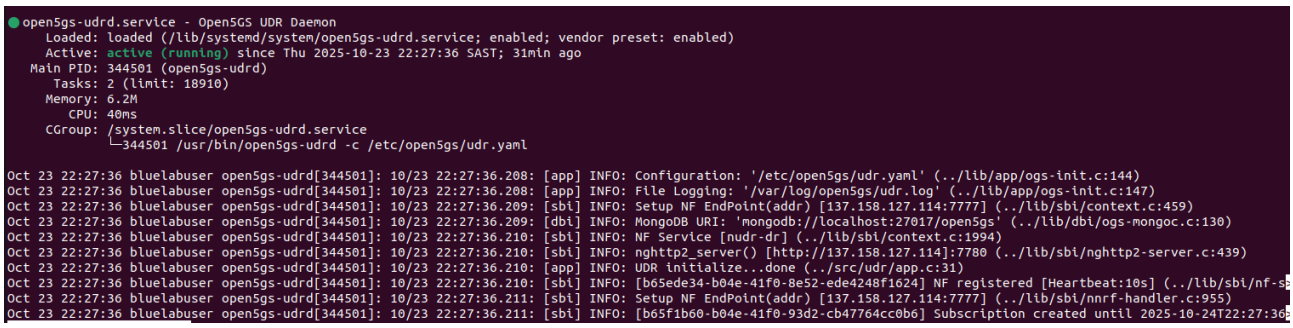
(b) Status of NRF and SCP core services



(a) Status of UDM and PCF core services



(b) Status of NRF and SCP core services

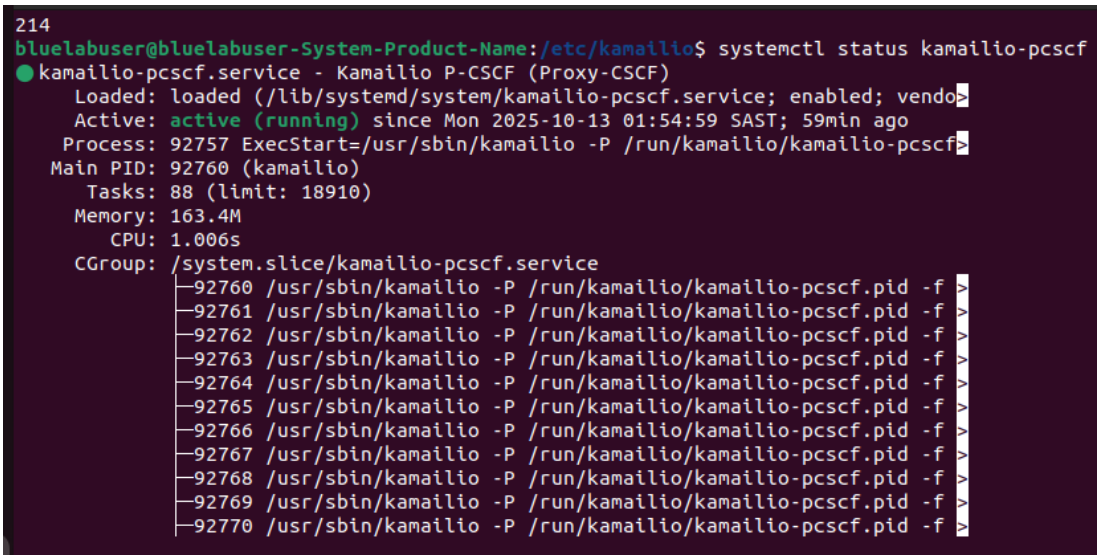


(c) status of UDR core service

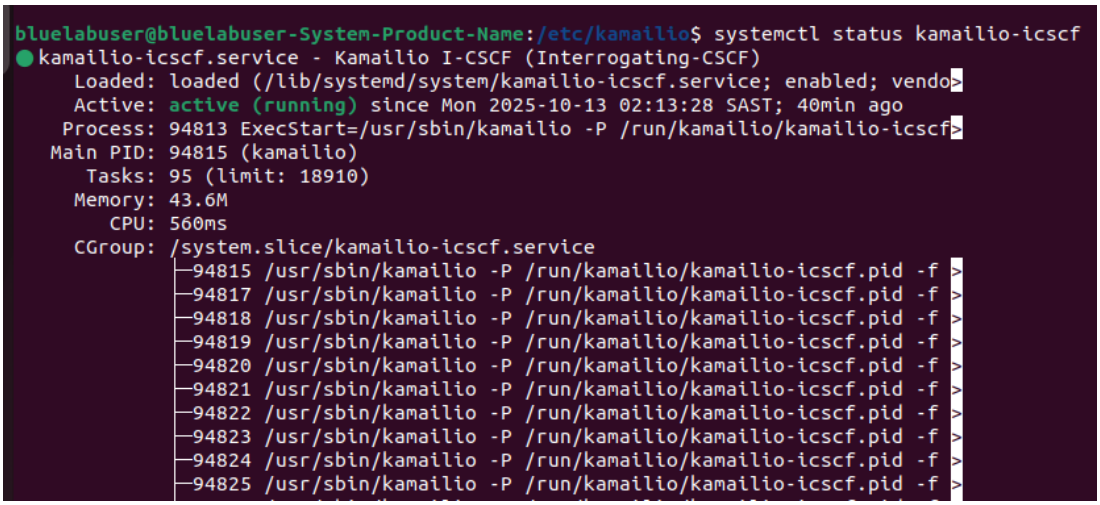
Figure 4.7: statuses of the core services

4.1.3 IMS Core (IP Multimedia Subsystem)

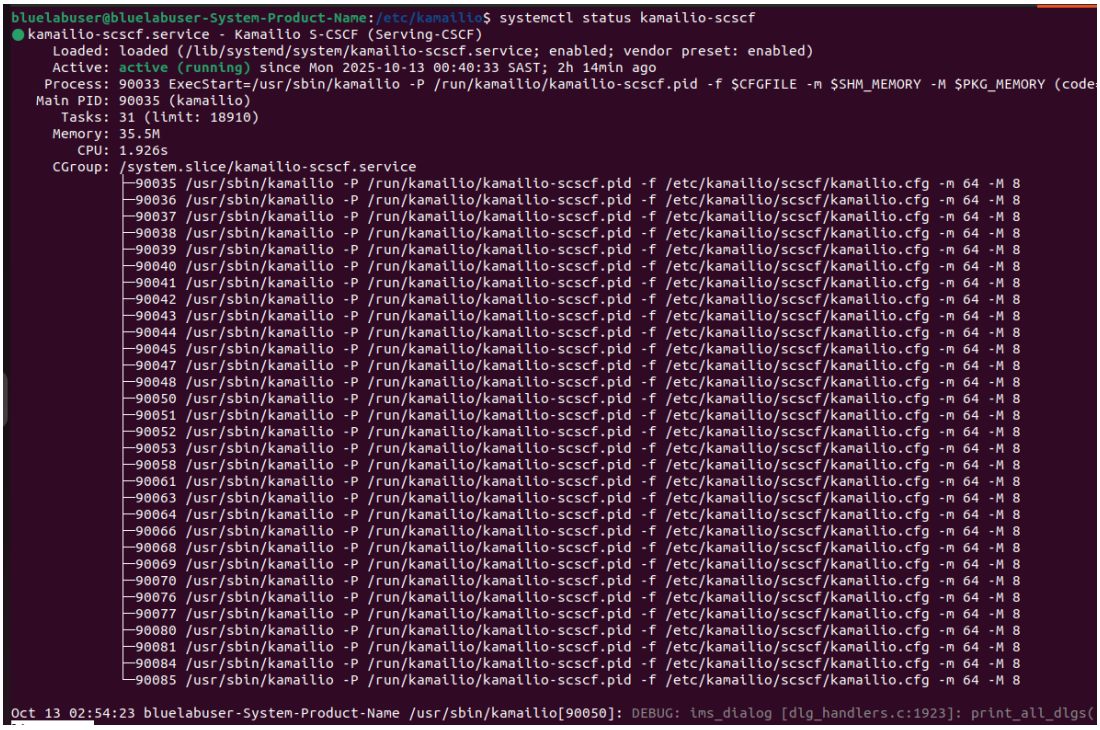
Voice call signaling and session management were implemented using Kamailio, an open-source SIP server acting as the Proxy-CSCF (P-CSCF) in the IMS core. The P-CSCF serves as the first point of contact for SIP registration requests, which are then forwarded to the Interrogating-CSCF (I-CSCF). The I-CSCF queries the HSS via the Cx interface to determine the appropriate Serving-CSCF (S-CSCF) for the subscriber. Once identified, the S-CSCF completes the registration process. The following figure 4.8 show the operational status of these three IMS control functions.



(a) status of the PCSCF



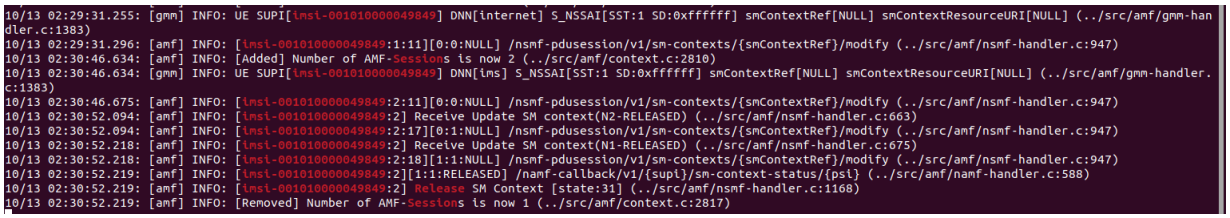
(b) status of the ICSCF



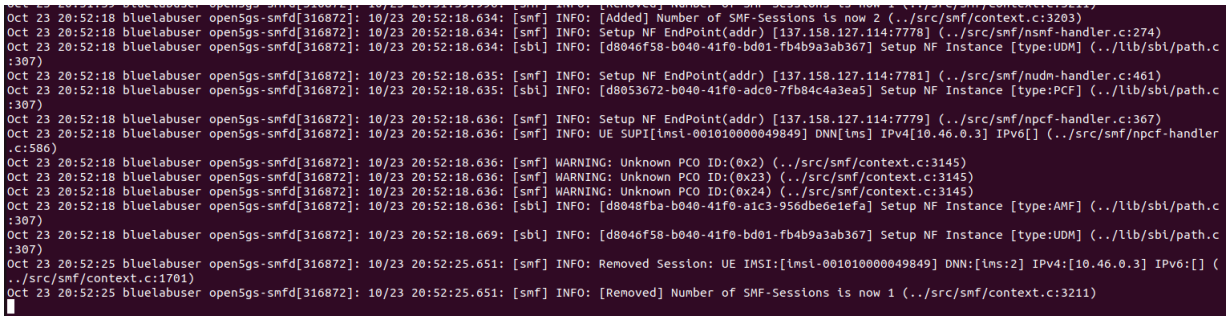
(c) status of the SCSCF

Figure 4.8: statuses of the IMS control functions

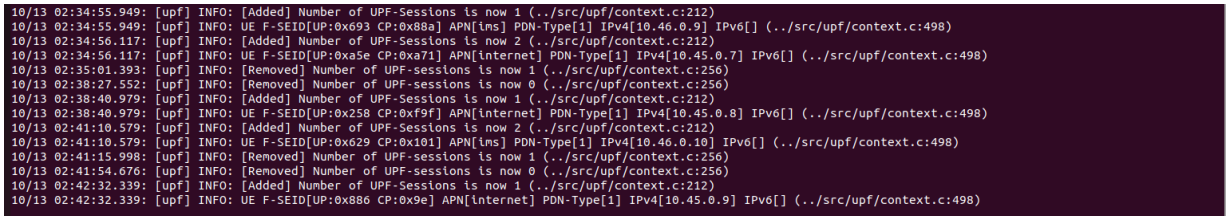
The figure 4.9 displays log entries generated by the 5G Core components during initialization and runtime. The logs include messages from key functions such as AMF, SMF, and UPF, showing their startup processes and signaling activities.



(a) AMF logs on the core network



(b) SMF logs on the core network



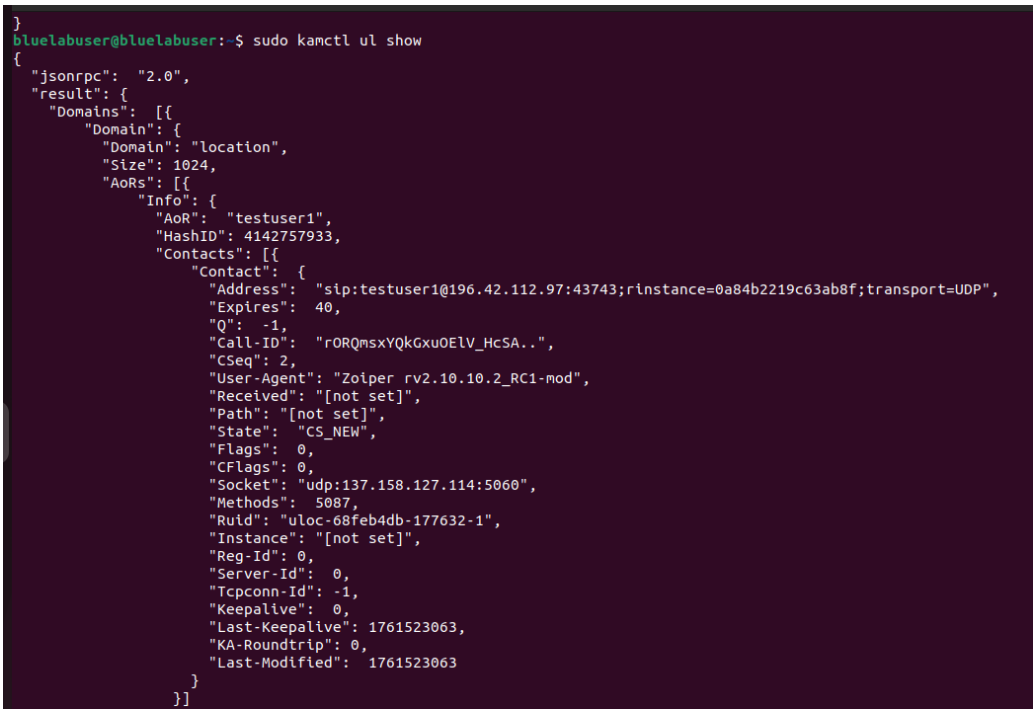
(c) UPF logs on the core network

Figure 4.9: Logs from core network functions during UE registration and PDU session establishment.

Implemented Configuration: To enable VoNR call testing, Kamailio was configured as a standalone SIP server. Two user equipment devices were configured with Zoiper SIP client applications:

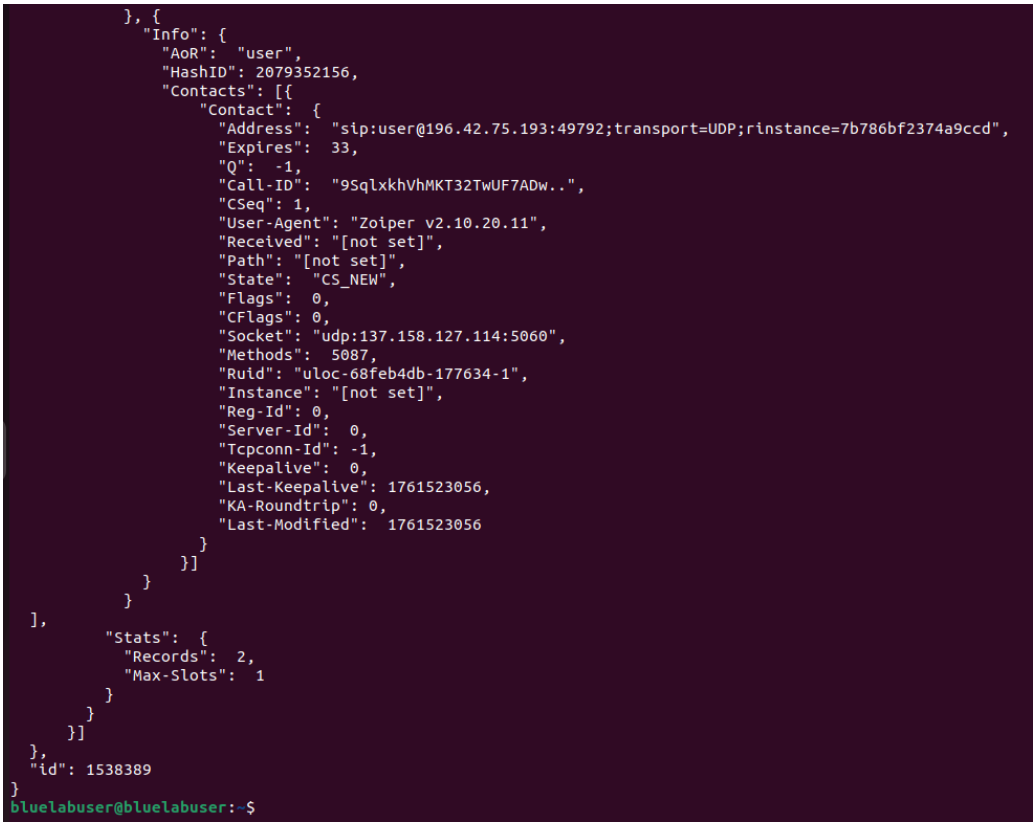
UE 1: Connected through the 5G gNB (radio access) UE 2: Connected via Wi-Fi

Both users were registered with subscriber credentials stored on the Kamailio server, which performed P-CSCF functionality for SIP registration and call signaling. Figure 4.10



```
}
bluelabuser@bluelabuser:~$ sudo kamctl ul show
{
  "jsonrpc":  "2.0",
  "result": {
    "Domains":  [{
        "Domain": {
          "Domain": "location",
          "Size": 1024,
          "AoRs": [{
              "Info": {
                "AoR":  "testuser1",
                "HashID": 4142757933,
                "Contacts": [{
                    "Contact":  {
                      "Address":  "sip:testuser1@196.42.112.97:43743;rinstance=0a84b2219c63ab8f;transport=UDP",
                      "Expires":  40,
                      "Q":  -1,
                      "Call-ID":  "rORQmsxYQkGxuOElV_HcSA..",
                      "CSeq": 2,
                      "User-Agent": "Zoiper rv2.10.10.2_RC1-mod",
                      "Received": "[not set]",
                      "Path": "[not set]",
                      "State":  "CS_NEW",
                      "Flags":  0,
                      "CFlags": 0,
                      "Socket": "udp:137.158.127.114:5060",
                      "Methods":  5087,
                      "Ruid": "uloc-68feb4db-177632-1",
                      "Instance": "[not set]",
                      "Reg-Id": 0,
                      "Server-Id":  0,
                      "Tcpconn-Id": -1,
                      "Keepalive":  0,
                      "Last-Keepalive": 1761523063,
                      "KA-Roundtrip": 0,
                      "Last-Modified":  1761523063
                    }
                  }]
```


(a) Record of first registered user in the Kamailio server



```
            }, {
              "Info": {
                "AoR":  "user",
                "HashID": 2079352156,
                "Contacts": [{
                    "Contact":  {
                      "Address":  "sip:user@196.42.75.193:49792;transport=UDP;rinstance=7b786bf2374a9ccd",
                      "Expires":  33,
                      "Q":  -1,
                      "Call-ID":  "9SqlxkhVhMKT32TwUF7ADw..",
                      "CSeq": 1,
                      "User-Agent": "Zoiper v2.10.20.11",
                      "Received": "[not set]",
                      "Path": "[not set]",
                      "State":  "CS_NEW",
                      "Flags":  0,
                      "CFlags": 0,
                      "Socket": "udp:137.158.127.114:5060",
                      "Methods":  5087,
                      "Ruid": "uloc-68feb4db-177634-1",
                      "Instance": "[not set]",
                      "Reg-Id": 0,
                      "Server-Id":  0,
                      "Tcpconn-Id": -1,
                      "Keepalive":  0,
                      "Last-Keepalive": 1761523056,
                      "KA-Roundtrip": 0,
                      "Last-Modified":  1761523056
                    }
                  }]
              }
            }
  ],
          "Stats":  {
            "Records":  2,
            "Max-Slots":  1
          }
        }
      }]
  },
  "id": 1538389
}
bluelabuser@bluelabuser:~$
```


(b) Record of the second registered user in the Kamailio server

Figure 4.10: Registered sip users

4.2 Home subscriber Server

PyHSS (hss2) was deployed as the Home Subscriber Server for the IMS platform, providing subscriber profile management, authentication, authorization, and location information to the serving Control Function of the IMS. The figure 4.11 shows the status of pyHSS, while figure4.12 displays two subscriber entries with MSISDN, IMSI, and associated S-CSCF bindings.

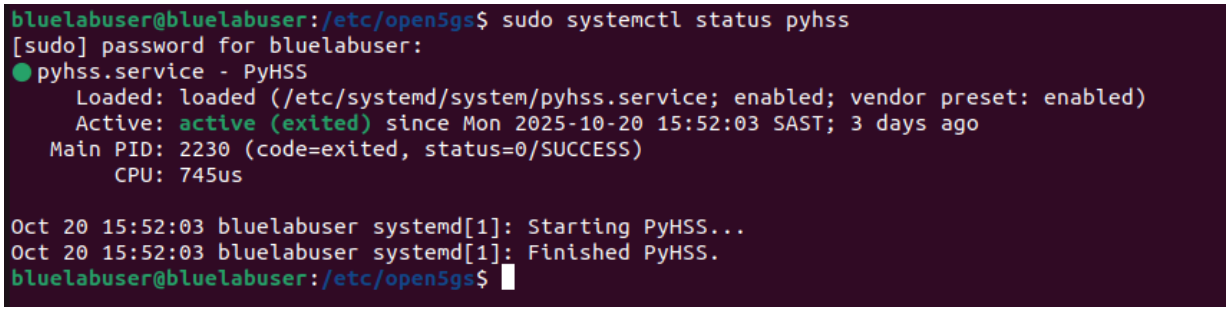


Figure 4.11: Status of PyHSS service

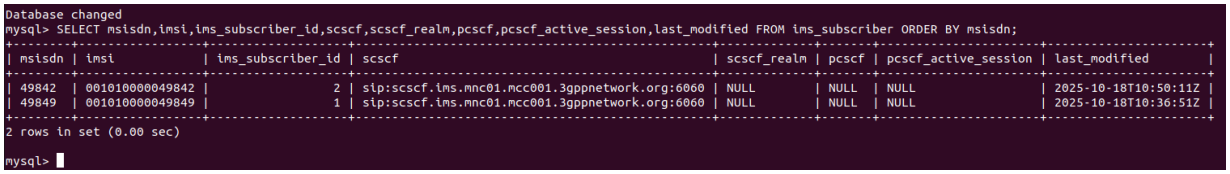


Figure 4.12: IMS Subscriber Records in PyHSS Database

4.3 Baseline Performance Measurements

Table 4.1: Baseline parameters and corresponding MOS

Parameter	MOS	sampling time
Latency(ms)	4.32	30
Jitter(ms)	4.05	30
Packet loss (%)	4.58	30

4.4 Impact of Individual Network Impairments

4.4.1 Latency Variation

```
Testing LATENCY:
------------------------------------------------------------------------
[11/50] Baseline... ..... ✓ 364KB
    Reference. MOS: 4.321
[12/50] Latency 50ms... ..... ✓ 338KB
    MOS: 1.000
[13/50] Latency 100ms... ..... ✓ 338KB
    MOS: 1.914
[14/50] Latency 150ms... ..... ✓ 338KB
    MOS: 1.000
[15/50] Latency 200ms... ..... ✓ 338KB
    MOS: 2.052
```

Figure 4.13: Latencies applied

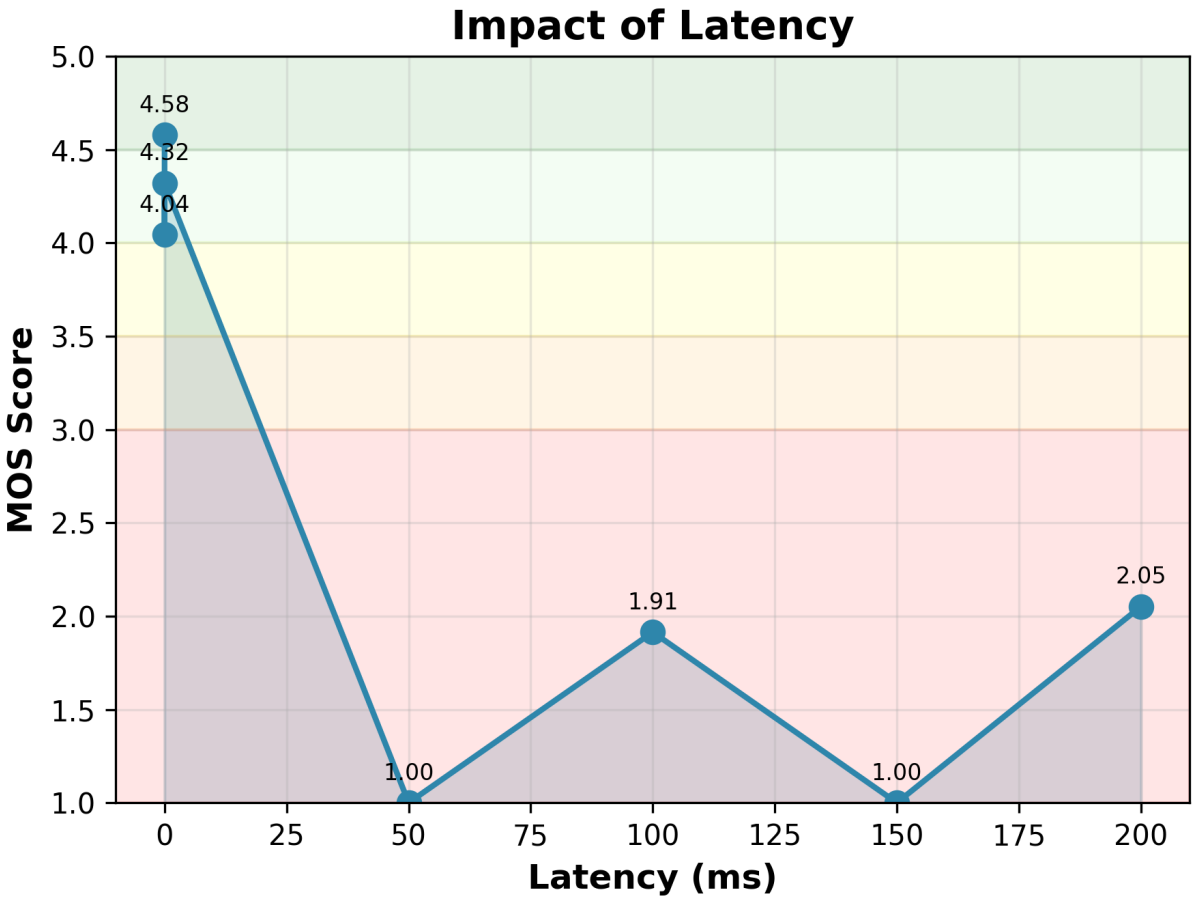


Figure 4.14: MOS vs latency

4.4.2 Jitter Variation

```
Testing JITTER:
------------------------------------------------------------------------
[6/50] Baseline... ..... ✓ 338KB
    Reference. MOS: 4.045
[7/50] Jitter 25ms... ..... ✓ 341KB
    MOS: 1.000
[8/50] Jitter 50ms... ..... ✓ 339KB
    MOS: 1.000
[9/50] Jitter 75ms... ..... ✓ 351KB
    MOS: 1.000
[10/50] Jitter 100ms... ..... ✓ 339KB
    MOS: 1.000
```

Figure 4.15: Jitter applied

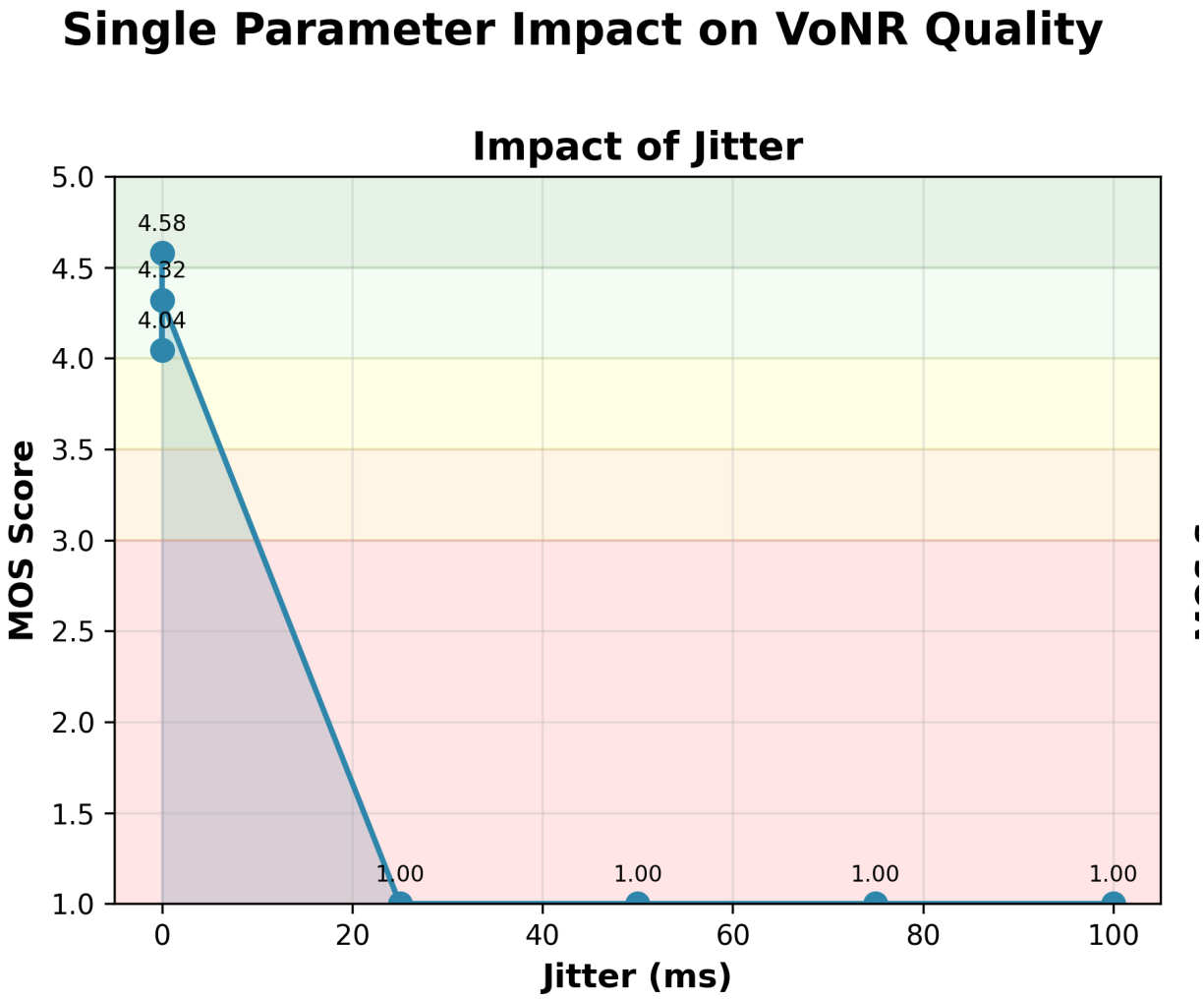


Figure 4.16: MOS vs jitter

4.4.3 Packet loss Variation

```
Testing LOSS:
----------------------------------------------------------------------
[1/50] Baseline... ..... ✓ 338KB
    Reference. MOS: 4.580
[2/50] Loss 5%... ..... ✓ 325KB
    MOS: 1.422
[3/50] Loss 10%... ..... ✓ 303KB
    MOS: 1.000
[4/50] Loss 15%... ..... ✓ 288KB
    MOS: 1.048
[5/50] Loss 20%... ..... ✓ 271KB
    MOS: 1.000
```

Figure 4.17: Packet Loss applied

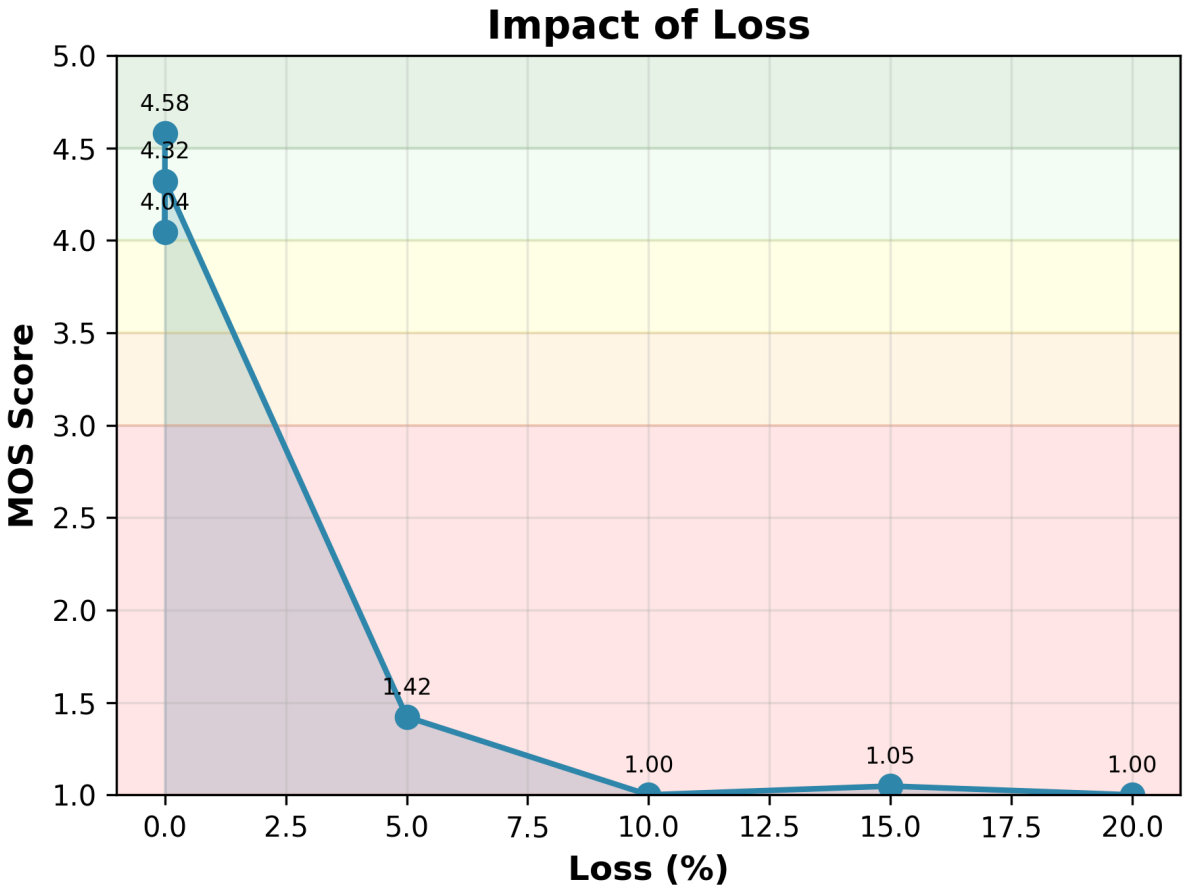


Figure 4.18: MOS vs Packet loss

4.5 Combined impairments Scenarios

```
================================================================================
PHASE 2: COMBINED PARAMETER TESTING
================================================================================

[16/50] Light combined... ..... ✓ 335KB
    MOS: 1.038
[17/50] Moderate combined... ..... ✓ 615KB
    MOS: 1.000
[18/50] Heavy combined... ..... ✓ 304KB
    MOS: 1.000
[19/50] Severe combined... ..... ✓ 290KB
    MOS: 1.289
[20/50] Loss + Latency... ..... ✓ 328KB
    MOS: 1.000
[21/50] Jitter + Latency... ..... ✓ 338KB
    MOS: 1.499
[22/50] Loss + Jitter... ..... ✓ 321KB
    MOS: 1.000
[23/50] High Loss + High Jitter... ..... ✓ 315KB
    MOS: 1.000
[24/50] Light Loss + High Latency... ..... ✓ 331KB
    MOS: 1.000
```

Figure 4.19: combined variation applied

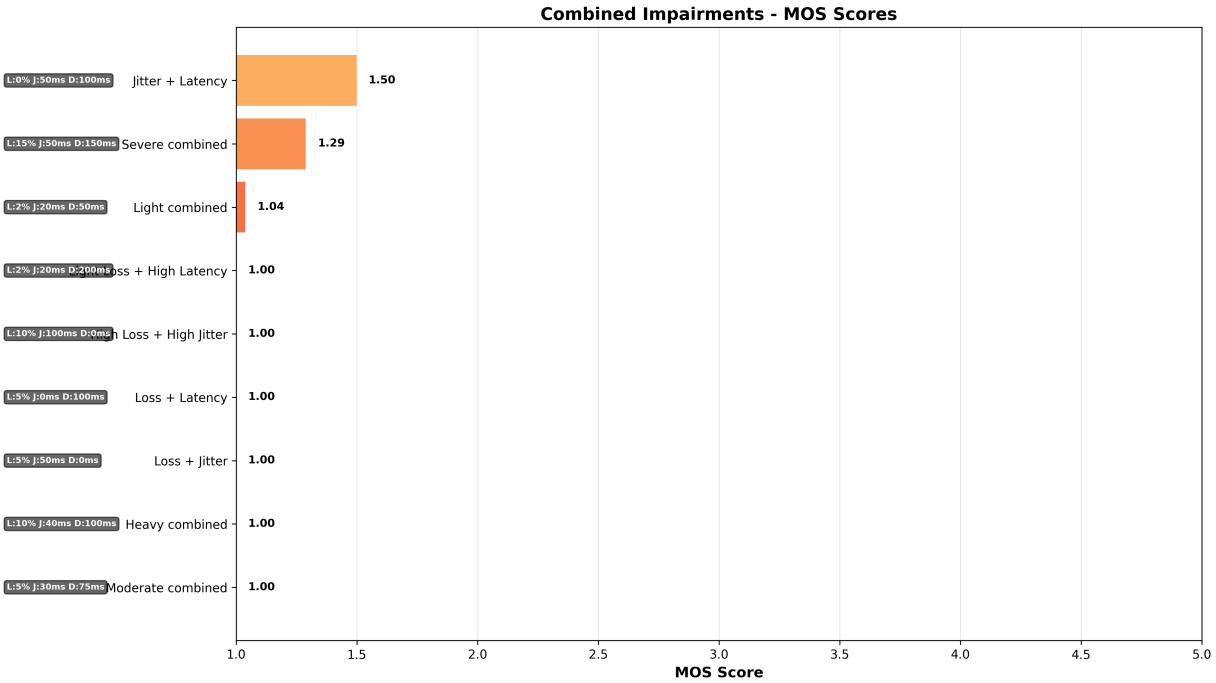
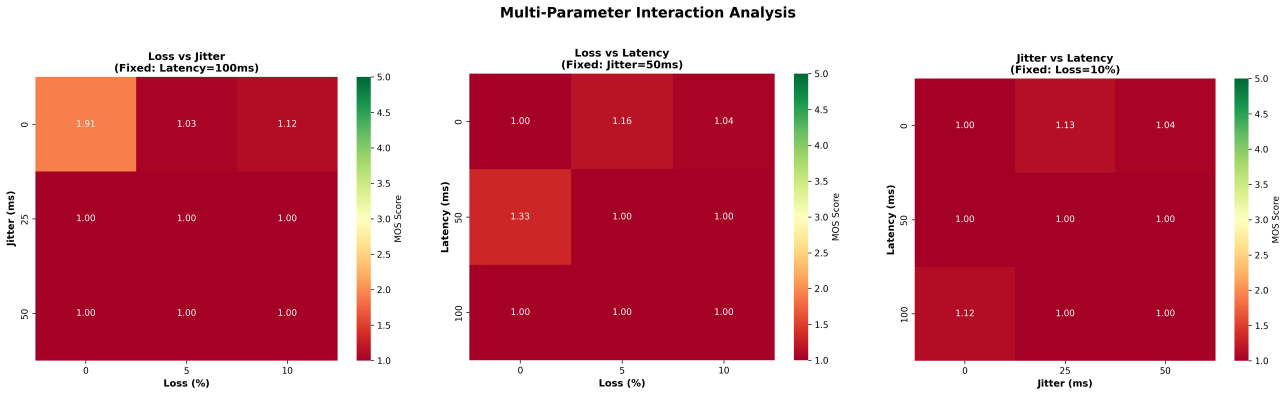


Figure 4.20: combined effect vs MOS

4.6 Full matrix effect

```
======================================================================
PHASE 3: FULL MATRIX TESTING
======================================================================

[25/50] L0% J25ms D50ms... ..... ✓ 339KB
    MOS: 1.000
[26/50] L0% J25ms D100ms... ..... ✓ 347KB
    MOS: 1.000
[27/50] L0% J50ms D50ms... ..... ✓ 337KB
    MOS: 1.328
[28/50] L0% J50ms D100ms... ..... ✓ 338KB
    MOS: 1.000
[29/50] L5% J0ms D50ms... ..... ✓ 324KB
    MOS: 1.000
[30/50] L5% J0ms D100ms... ..... ✓ 321KB
    MOS: 1.035
[31/50] L5% J25ms D0ms... ..... ✓ 321KB
    MOS: 1.000
[32/50] L5% J25ms D50ms... ..... ✓ 324KB
    MOS: 1.000
[33/50] L5% J25ms D100ms... ..... ✓ 322KB
    MOS: 1.000
[34/50] L5% J50ms D0ms... ..... ✓ 322KB
    MOS: 1.157
[35/50] L5% J50ms D50ms... ..... ✓ 322KB
    MOS: 1.000
[36/50] L5% J50ms D100ms... ..... ✓ 327KB
    MOS: 1.000
[37/50] L10% J0ms D50ms... ..... ✓ 305KB
    MOS: 1.000
[38/50] L10% J0ms D100ms... ..... ✓ 310KB
    MOS: 1.120
[39/50] L10% J25ms D0ms... ..... ✓ 304KB
    MOS: 1.126
[40/50] L10% J25ms D50ms... ..... ✓ 315KB
    MOS: 1.000
[41/50] L10% J25ms D100ms... ..... ✓ 306KB
    MOS: 1.000
[42/50] L10% J50ms D0ms... ..... ✓ 306KB
    MOS: 1.039
[43/50] L10% J50ms D50ms... ..... ✓ 307KB
    MOS: 1.000
[44/50] L10% J50ms D100ms... ..... ✓ 303KB
    MOS: 1.000
```

Figure 4.21: Variation two parameter with the third kept constant



(a) Varying packet loss and jitter with packet latency kept constant

(b) Varying latency and packet loss with jitter kept constant

(c) Varying jitter and latency with packet loss kept constant

Figure 4.22: Varying two parameter with third kept constant

Chapter 5

Discussion

5.1 Testbed Implementation and Validation

The successful deployment of a private 5G Standalone network testbed formed the foundation of this research. The base station implementation using srsRAN gNB software paired with the Ettus USRP B210 SDR demonstrated stable operation at 3.5 GHz (n78 band). The UHD FFT plot 4.3 confirmed successful gNB transmission with a clear carrier peak at -65 dB and a sampling rate of 30.72 MS/s, which is appropriate for 5G NR operation. This validated that the RF front-end was functioning correctly and providing adequate signal strength for the experimental setup. Addtionally, the AMF logs on the core site indicated the successful addition of the gNB as can be seen in figure4.2

The Open5GS core network provided all essential 5G SA functions, as shown in Figure 4.7. Key components such as AMF, SMF, and UPF initialized correctly and maintained runtime operation, establishing signaling and data plane connectivity for standard services without errors.

5.2 IMS Integration Challenges and PDU Session Stability

The three IMS core control functions responsible for signaling and user registration for VoNR were successfully deployed, as shown by their respective status outputs in figure 4.8 a to c.

A significant technical challenge emerged during the IMS implementation phase that directly impacted the VoNR testing approach. Analysis of the AMF, SMF, and UPF logs revealed a critical issue 4.9: while the user equipment successfully established both the default internet DNN (Data Network Name) and the IMS DNN, the IMS PDU session would be released after only a few seconds despite initial acceptance by the core network.

The root cause appeared to be the incomplete integration between the IMS core services (PCSCF, ICSCF, SCSCF running on Kamailio) and the 5G Core network. Although all IMS components were successfully instantiated and individually operational 4.8, proper communication channels between the IMS infrastructure and the 5G Core's control plane functions were not fully established. This disconnect meant that when the UE attempted to register with the IMS and establish a voice session, the lack of proper signaling between the IMS and core caused the UE to remain in an idle state for extended periods. The 5G Core, detecting no active IMS traffic or proper session management signaling, would then release the IMS PDU session to conserve resources, even though the default data bearer remained active.

This behavior is consistent with timeout mechanisms in 5G SA networks where inactive PDU sessions are automatically released if no proper keep-alive signaling occurs. The IMS registration and session

establishment procedures require continuous SIP signaling exchanges between the UE, PCSCF, and the core network's Policy Control Function (PCF). Without proper integration, these signaling flows could not complete, triggering session release.

Due to project time constraints, thorough troubleshooting of the IMS-to-Core interface—which would have required detailed analysis of SIP message flows, Diameter signaling (Cx/Dx interfaces), and potential routing misconfigurations—could not be completed within the available timeline.

5.3 Alternative Implementation: SIP-Based Voice Testing

To address the integration challenge while maintaining the core research objective, Kamailio was reconfigured as a standalone SIP server, operating independently of the 5G IMS architecture. Two user devices were registered using Zoiper SIP clients: one connected via the 5G gNB and another over Wi-Fi. An attempt was made to connect both UEs to the core network; however, issues arose during PDU session handling. The Google Pixel 7 initially attached and maintained a single PDU session but later disconnected completely, despite detecting the gNB. In contrast, the OnePlus 11 released only the IMS DNN while retaining internet connectivity. Both endpoints registered directly with the Kamailio SIP server using subscriber credentials managed in the local database 4.10

This configuration successfully enabled voice call establishment between the two endpoints, with voice packets transmitted as RTP streams over the data bearers. Critically, one call leg traversed the 5G radio access network and the 5G Core's user plane, allowing for voice quality assessment under controlled impairment scenarios.

It is important to acknowledge that this approach does not represent true VoNR service as defined by 3GPP standards, where voice calls must traverse the IMS core and utilize IMS-specific QoS mechanisms, dedicated voice bearers, and policy control. In the implemented configuration, voice traffic was treated as generic data traffic over the default PDU session rather than as a specialized IMS voice session with guaranteed QoS parameters.

However, for the purposes of this research, evaluating how network impairments latency, jitter, and packet loss affect voice quality over 5G infrastructure, this approach remained valid. The RTP voice streams still traversed the 5G radio interface, experienced the same network conditions applied at the UPF level via traffic control, and were subject to the same physical layer characteristics. The quality assessment methodology using ViSQOL and the systematic impairment testing remained unaffected by the SIP-vs-IMS distinction, as the actual voice codec, packetization, and transport mechanisms were identical. The primary difference was in call setup signaling and QoS handling, not in the voice stream characteristics being measured.

The baseline measurements under ideal conditions yielded MOS scores around 4.0-4.6 across all test scenarios 4.1, confirming that the system could deliver good quality voice service when network conditions were optimal. These scores provided a solid reference point for evaluating degradation under impaired conditions. The slight variations in baseline MOS (4.32 for latency, 4.05 for jitter, 4.58 for packet loss) reflect the inherent characteristics of the G.711 codec and natural variability in the radio environment, even under controlled conditions.

5.4 Individual Impairment Effects

The results showed severe degradation at much lower values, with MOS dropping to 1.0 at 50 ms and remaining below 2.1 at 200 ms. This contradicts the recommendation by ITU-T G.114 which recommends keeping one-way latency under 150 ms for acceptable call quality. Several factors explain this deviation from guidelines.

The methodology followed ITU-T P.862, using G.711 at 8 kHz to decode RTP streams, matching Zoiper's active codecs (μ-law and A-law). Although G.711 typically handles moderate latency well, it lacks packet loss concealment and is sensitive to jitter, which likely caused the sharp quality decline. GSM was also enabled as a fallback codec, offering lower quality and less resilience under network impairments. Additionally, Zoiper's jitter buffer may not have been optimally configured for the test conditions.

The supervisor initially recommended Linphone, which supports more robust codecs like Opus. However, given tight time constraints and the setup overhead required, Zoiper was chosen as a practical alternative to move forward with testing. These combined limitations explain why the results deviate significantly from ITU-T expectations.

5.5 Combined impairments Scenarios

Contrary to expectations, jitter and packet loss did not exhibit gradual degradation but rather sharp thresholds. As shown in Figures 1.14 and 1.16, MOS remained high (4.58–4.04) under minimal impairment but dropped abruptly to 1.00 at jitter values of 20 ms and beyond, and at packet loss rates of 10% or higher. This suggests that the system lacked adaptive jitter buffering or robust packet loss concealment, causing catastrophic quality failure once critical thresholds were reached. The results highlight the importance of codec resilience and buffering strategies in maintaining conversational quality under adverse conditions.

All combined impairment scenarios resulted in very poor MOS scores, mostly around 1.0, indicating unusable voice quality. This suggests that the system is extremely sensitive to multiple simultaneous impairments, and even light combinations push quality below acceptable thresholds.

5.6 Quality Thresholds and Practical Guidelines

Based on the impairment testing conducted, clear thresholds and practical insights emerged for voice service delivery over 5G infrastructure:

1. Latency: The system exhibited extreme sensitivity to delay, with MOS collapsing to 1.0 at 50 ms and remaining below 2.1 even at 200 ms4.14. This deviates from ITU-T G.114, which recommends keeping one-way latency below 150 ms for acceptable quality. The results highlight the need for aggressive latency control and optimized jitter buffering in the client application.

2. Jitter: Contrary to expectations, jitter tolerance was minimal. MOS dropped to 1.0 at 20 ms and remained unusable up to 100 ms 4.16, indicating that the RTP stack lacked adaptive jitter

buffering or that buffer configuration was insufficient. For practical deployments, jitter should be kept below 10 ms to maintain conversational quality.

3. Packet Loss: Packet loss proved catastrophic beyond 5%, with MOS falling to 1.42 at 5% and 1.0 at 10% and above4.18. This aligns with codec behavior for G.711, which lacks advanced packet loss concealment. Operators should aim for near-zero packet loss, ideally below 1%, to ensure service reliability.

5.7 Methodology Evaluation and Limitations

The automated testing framework built using Python script proved highly effective for systematic data collection and analysis. By integrating tshark for RTP extraction, tc netem for impairment simulation, and ViSQOL for objective quality assessment, the workflow ensured reproducibility and provided a foundation that can be easily adapted for future testing scenarios.

However, several methodological challenges were encountered. Ensuring complete isolation between successive test conditions proved difficult—the consistent MOS scores observed across certain impairment ranges suggest that audio samples may have contained overlapping content despite PCAP clearing procedures.

Chapter 6

Conclusion

This research successfully demonstrated the feasibility of implementing Voice over New Radio (VoNR) on a private 5G Standalone network using Software-Defined Radios (SDRs) and open-source software. The primary objectives, designing a VoNR testbed, integrating IMS components, and evaluating voice quality under varying network conditions, were achieved, albeit with practical constraints that influenced certain outcomes.

The study revealed that while the testbed approach provides valuable insights into real-world behaviors, achieving seamless IMS integration remains a significant challenge. The workaround of using SIP over the 5G data bearer mirrors early industry practices and highlights the flexibility of IP-based voice services, even if it departs from ideal architectural principles. Performance testing showed that latency is the most critical impairment, with MOS collapsing far earlier than ITU-T G.114 thresholds, likely due to codec limitations and lack of jitter buffer optimization in the chosen softphone. Jitter and packet loss exhibited abrupt degradation rather than gradual decline, and combined impairments consistently resulted in unusable quality. These findings underscore the importance of codec selection, robust buffering, and error concealment for VoNR deployments.

Despite these limitations, the automated testing framework proved effective and reproducible, leveraging Python, Jupyter, TShark, TC Netem, and ViSQOL for systematic analysis. The research also highlighted the need for broader codec support, such as EVS or Opus, and improved methodology to ensure isolation between test conditions. In summary, this work demonstrates both the promise and complexity of delivering high-quality voice services over 5G SA networks. It provides practical insights for operators and researchers, emphasizing low-latency design, holistic QoS management, and iterative testing as key enablers for successful VoNR deployment.

Chapter 7

Recommendation

It is recommended that future work expands codec testing beyond G.711 μ-law to include modern wideband codecs such as Opus and EVS, which offer improved resilience to network impairments. The use of Linphone, as advised by the supervisor, should be prioritized due to its broader codec support and advanced jitter buffer configuration options. This would address limitations encountered with Zoiper and enable more representative results.

Further methodological improvements should be implemented to ensure complete isolation between test conditions. Longer stabilization periods and automated sample verification are advised to prevent overlapping audio content. A significant challenge encountered during this research was the reliance on a real mobile phone for call initiation and termination, which prevented full automation of the process. Future iterations should consider using virtual endpoints or automated SIP clients to eliminate this dependency.

IMS integration should be revisited to enable native VoNR service delivery rather than relying on SIP over the data bearer. Complementing objective MOS measurements with subjective listening tests is also recommended to capture user-perceived quality, particularly under combined impairment conditions. Finally, scalability testing should be introduced to simulate multiple concurrent calls and realistic traffic patterns, providing a more comprehensive assessment of VoNR performance under operational load.

Bibliography

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

[2] S.-S. Manfred, "Circuit switching versus packet switching," *International Journal of Open Information Technologies*, vol. 3, no. 4, pp. 27–37, 2015.

[3] N. D. E. Jerger, L.-S. Peh, and M. H. Lipasti, "Circuit-switched coherence," in *Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008)*, 2008, pp. 193–202.

[4] M. Kassim, R. A. Rahman, M. A. A. Aziz, A. Idris, and M. I. Yusof, "Performance analysis of voip over 3g and 4g lte network," in *2017 International Conference on Electrical, Electronics and System Engineering (ICEESE)*, 2017, pp. 37–41.

[5] D. Zydek, N. Shlayan, E. Regentova, and H. Selvaraj, "Review of packet switching technologies for future noc," in *2008 19th International Conference on Systems Engineering*, 2008, pp. 306–311.

[6] K. Mubasier, F. Y. Li, J. A. S. Øgaard, and M.-C. Vochin, "Campus-based full-scale and portable open-source 5g sa networks: Prototyping and experiments," in *2023 26th International Symposium on Wireless Personal Multimedia Communications (WPMC)*. IEEE, 2023, pp. 1–6.

[7] L.-A. Phan, D. Pesch, U. Roedig, and C. J. Sreenan, "Building a 5g core network testbed: Open-source solutions, lessons learned, and research directions," in *2024 International Conference on Information Networking (ICOIN)*. IEEE, 2024, pp. 641–646.

[8] J. J. A. Arnez, W. M. Silva, R. K. G. Do Reis, L. A. Da Silva, and M. G. L. Damasceno, "Sdr 5g nsa mobile network and an ims core to provide voice over ip lte service," in *2022 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC)*. IEEE, 2022, pp. 071–076.

[9] M. Amini and C. Rosenberg, "A comparative analysis of open-source software in an e2e 5g standalone platform," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2024, pp. 1–6.

[10] R. Mohamed, S. Zemouri, and C. Verikoukis, "Performance evaluation and comparison between sa and nsa 5g networks in indoor environment," in *2021 IEEE international mediterranean conference on communications and networking (MeditCom)*. IEEE, 2021, pp. 112–116.

[11] J. J. A. Arnez, M. G. L. Damasceno, R. K. G. Dos Reis, and L. D. S. Pessoa, "Real time testbeds for 5g nsa and sa mobile architectures to provide volte and vonr services over ip multimedia core network subsystem," in *2022 18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2022, pp. 63–68.

[12] Z. Cheng, M. Ordean, F. D. Garcia, B. Cui, and D. Rys, "Watching your call: breaking volte privacy in lte/5g networks," *arXiv preprint arXiv:2301.02487*, 2023.

[13] H. Ji, S. Park, J. Yeo, Y. Kim, J. Lee, and B. Shim, "Ultra-reliable and low-latency communications in 5g downlink: Physical layer aspects," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 124–130, 2018.

[14] I. Rahman, S. M. Razavi, O. Liberg, C. Hoymann, H. Wiemann, C. Tidestav, P. Schliwa-Bertling, P. Persson, and D. Gerstenberger, "5g evolution toward 5g advanced: An overview of 3gpp releases 17 and 18," *Ericsson Technology Review*, vol. 2021, no. 14, pp. 2–12, 2021.

[15] H. Fehmi, M. F. Amr, A. Bahnasse, and M. Talea, "5g network: analysis and compare 5g nsa/5g sa," *Procedia Computer Science*, vol. 203, pp. 594–598, 2022.

[16] J. Yu, C. Hu, W. Xie, P. Lin, J. Hou, and X. Xu, "Research on coverage and capacity of 5g vonr," in *2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2022, pp. 1–5.

[17] J. Shi, S. Wang, M.-Y. Chen, G.-H. Tu, T. Xie, M.-H. Chen, Y. Hu, C.-Y. Li, and C. Peng, "Ims is not that secure on your 5g/4g phones," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 513–527.

[18] M. Dattatreya, "Ims in 5g," 2020.

[19] F. Zafar, "Ims service orchestration and interaction challenges," Master's thesis, NTNU, 2019.

[20] J. G. Beerends, C. Schmidmer, J. Berger, M. Obermann, R. Ullmann, J. Pomy, and M. Keyhl, "Perceptual objective listening quality assessment (polqa), the third generation itu-t standard for end-to-end speech quality measurement part i—temporal alignment," *journal of the audio engineering society*, vol. 61, no. 6, pp. 366–384, 2013.

[21] J. Pomy, "Polqa: The next-generation mobile voice quality testing standard," in *ZNIIS/ITU Workshop*, 2011.

[22] A. Hines, J. Skoglund, A. Kokaram, and N. Harte, "Robustness of speech quality metrics to background noise and network degradations: Comparing visqol, pesq and polqa," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3697–3701.

[23] M. Chinen, F. S. Lim, J. Skoglund, N. Gureev, F. O'Gorman, and A. Hines, "Visqol v3: An open source production ready objective speech and audio metric," in *2020 twelfth international conference on quality of multimedia experience (QoMEX)*. IEEE, 2020, pp. 1–6.

[24] E. Cipressi and M. L. Merani, "A comparative study on the quality of narrow-band and wide-band amr volte calls," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 1273–1278.

[25] A. Prasad, S. Bhatia, L. Duan, F. Zawaidehtt, C. Chen, J. Wu, M. Mittal, and S. Ramachandran, "Enhanced voice services based volte rate adaptation mechanism to improve quality of experience,"

in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–6.

[26] P. He, N. Zhang, and W. Liang, "Evaluation of the voice quality of vonr and volte," in *Proceedings of the 2024 6th International Symposium on Signal Processing Systems*, 2024, pp. 17–21.

[27] M. Chepkoech, N. Mombeshora, B. Malila, and J. Mwangama, "Evaluation of open-source mobile network software stacks: A guide to low-cost deployment of 5g testbeds," in *2023 18th Wireless On-Demand Network Systems and Services Conference (WONS)*. IEEE, 2023, pp. 56–63.

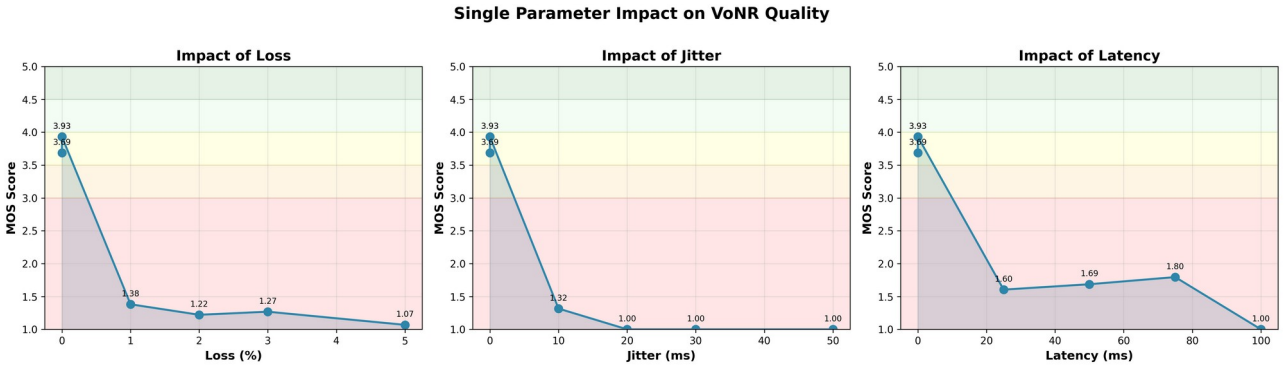
[28] E.-R. Modroiu, J. Mwangama, M. Corici, and T. Magedanz, "Experimental evaluation of open-source-based vonr and ott voice services in 5g sa network deployments," *Journal of Communications Software and Systems*, vol. 21, no. 2, pp. 166–177, 2025.

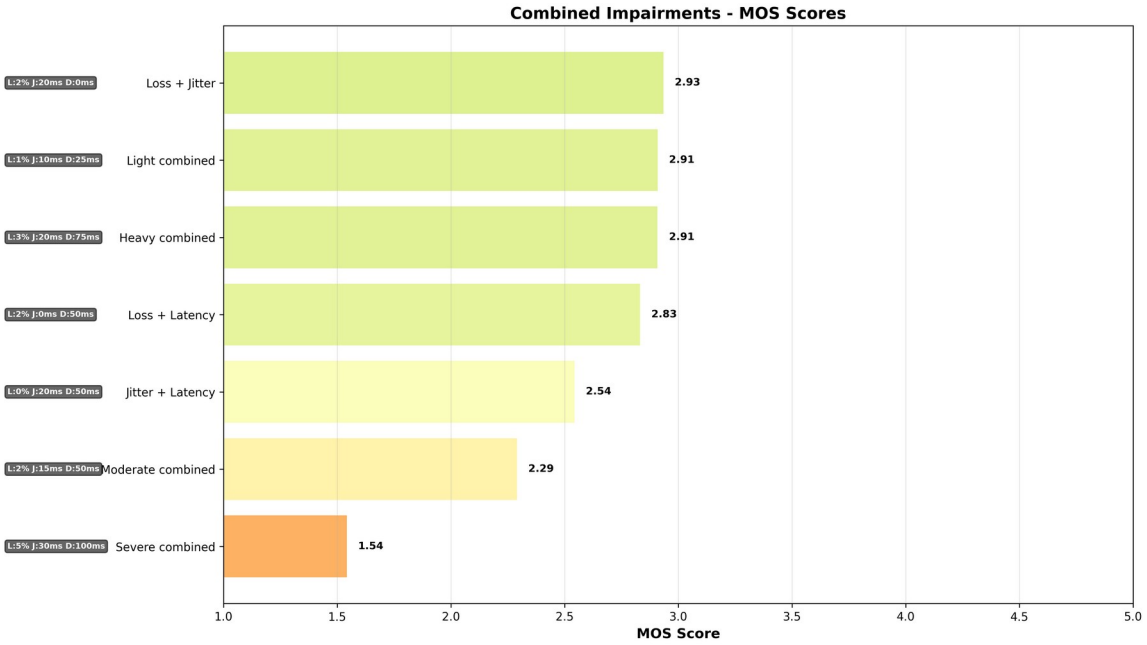
GA REPORT FORM

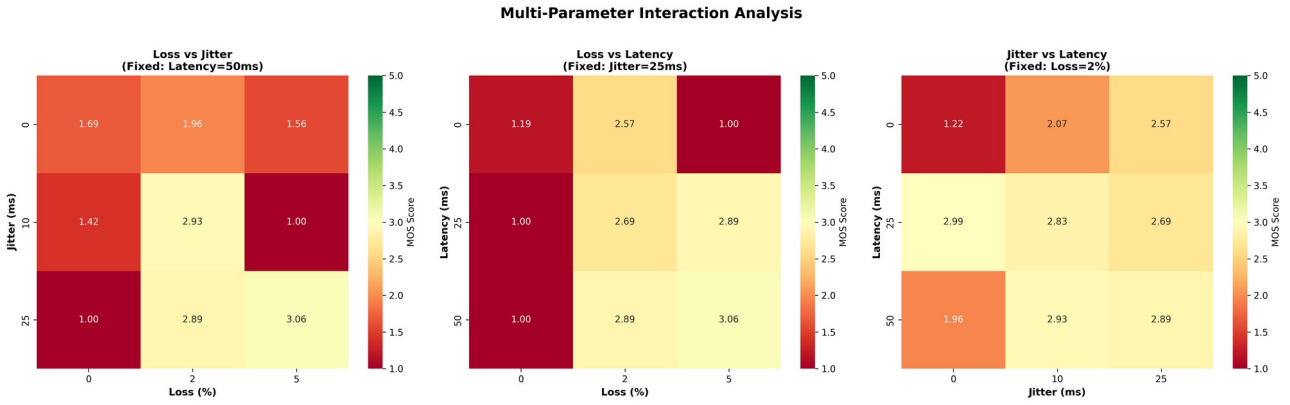
GA	Requirement	Justification and section in the report
1	Problem-solving	By applying theoretical knowledge from 5G SA networks and IMS, guided by my supervisor and relevant literature, I successfully built a functional testbed supporting voice services over 5G using Open5GS, srsRAN, Kamailio, and PyHSS. The B210 SDR was configured and verified, establishing stable NGAP connectivity between the gNB and core network. All essential core services (AMF, SMF, UPF, NRF, etc.) were successfully deployed and operational. While full IMS integration could not be achieved due to PDU session stability issues, I implemented Kamailio as a standalone SIP server, which closely represents early industry practices for IP voice delivery and enabled successful call establishment and quality assessment. Commercial 5G devices with provisioned USIM cards were successfully registered to the network. My supervisor's feedback was instrumental in navigating challenges and adapting the methodology systematically
4	Investigations, experiments and data analysis	I used UHD tools to confirm SDR hardware detection and successfully registered the gNB with the Open5GS core. The MongoDB subscriber database was verified and functioning correctly. Commercial 5G UE devices with provisioned USIM cards were successfully registered to the network. Wireshark was set up for packet capture to enable RTP stream analysis during voice calls. Controlled network impairments, latency, jitter, and packet loss, were systematically introduced using Linux Traffic Control (tc netem) at the UPF interface. ViSQOL was then used to analyze the extracted audio and quantify objective voice quality through MOS scores. This approach successfully connected measurable Quality of Service (QoS) impairments with user-perceived Quality of Experience (QoE), fulfilling a key objective of the research.
5	Use of engineering tools	I used several engineering tools for this project, guided by my supervisor's advice on their relevance. UHD was used for SDR verification, Open5GS for core network deployment, and srsRAN for the RAN. For IMS signaling, Kamailio was deployed as a standalone SIP server to enable voice call establishment. Wireshark was used for packet-level inspection and RTP stream capture. Mysql managed subscriber data for both the core network and IMS. ViSQOL was employed for objective voice

		quality scoring, calculating MOS values from extracted audio streams under various network impairment conditions.
6	Professional and technical communication (Long report)	I documented my work through structured progress reports, methodology drafts, and clear diagrams and flowcharts. Overleaf was used for report writing, which facilitated efficient reference management through its integrated bibliography tools. This process improved my ability to present results in a professional academic style suitable for communicating with peers and examiners.
8	Individual work	My supervisor's advice on key decision points was crucial to the project's success, though the work required significant independence. I took initiative in configuring and testing the B210 SDR, gNB registration, IMS deployment, and subscriber database provisioning. When challenges arose, such as IMS-to-Core integration issues and PDU session instability, I first investigated possible causes through log analysis and configuration review before consulting with my supervisor for guidance. This combination of self-driven troubleshooting and expert feedback was vital for adapting the methodology and achieving a functional voice testing framework
9	Independent learning ability	Setting up the project's testbed required learning new skills and tools like Kamailio IMS, Open5GS, srsRAN, and MongoDB, which were not part of my curriculum. While I had theoretical knowledge of 5G and databases, implementing them in an actual network environment presented significant challenges. I developed independence in research, relying on academic papers, open-source documentation, and my supervisor's guidance to find credible information and troubleshoot issues. This experience built my confidence in seeking knowledge independently while recognizing when to request expert guidance, a crucial balance for overcoming the project's technical difficulties and adapting the methodology when necessary

RESULTS OBTAINED WITH REDUCED IMPAIRMENTS







29	5.043494881	196.42.112.97	137.158.127.114	SIP	650 Request: REGISTER sip:137.158.127.114;transport=UDP (1 binding) |
32	5.043658978	137.158.127.114	196.42.112.97	SIP	551 Status: 200 OK (REGISTER) (1 binding) |
42	8.493361149	137.158.127.114	188.34.157.179	SIP	420 Request: OPTIONS sip:sbc-2.ng-voice.com |
43	8.493369894	137.158.127.114	49.13.207.109	SIP	420 Request: OPTIONS sip:sbc-1.ng-voice.com |
87	14.589914235	10.46.0.9	137.158.127.114	GTP <SIP/SDP>	1043 Request: INVITE sip:testuser1@sip.ims.mnc01.mcc001.3gppnetwork.org;tran…
88	14.590168047	137.158.127.114	10.46.0.9	GTP <SIP>	499 Status: 100 trying -- your call is important to us |
89	14.590954714	137.158.127.114	196.42.112.97	SIP/SDP	1249 Request: INVITE sip:testuser1@196.42.112.97:63737;rinstance=02e74b40dcf…
92	14.975741181	196.42.112.97	137.158.127.114	SIP	470 Status: 100 Trying |
99	15.993558948	137.158.127.114	188.34.157.179	SIP	420 Request: OPTIONS sip:sbc-2.ng-voice.com |
100	15.993650770	137.158.127.114	49.13.207.109	SIP	420 Request: OPTIONS sip:sbc-1.ng-voice.com |
103	16.101771923	196.42.112.97	137.158.127.114	SIP	737 Status: 180 Ringing |
104	16.101946746	137.158.127.114	10.46.0.9	GTP <SIP>	693 Status: 180 Ringing |
105	16.493360561	137.158.127.114	188.34.157.179	SIP	420 Request: OPTIONS sip:sbc-2.ng-voice.com |
106	16.493369610	137.158.127.114	49.13.207.109	SIP	420 Request: OPTIONS sip:sbc-1.ng-voice.com |
129	17.493354509	137.158.127.114	188.34.157.179	SIP	420 Request: OPTIONS sip:sbc-2.ng-voice.com |
130	17.493360806	137.158.127.114	49.13.207.109	SIP	420 Request: OPTIONS sip:sbc-1.ng-voice.com |
134	19.493356245	137.158.127.114	188.34.157.179	SIP	420 Request: OPTIONS sip:sbc-2.ng-voice.com |
135	19.493365481	137.158.127.114	49.13.207.109	SIP	420 Request: OPTIONS sip:sbc-1.ng-voice.com |
148	21.733415135	196.42.112.97	137.158.127.114	SIP/SDP	938 Status: 200 OK (INVITE) |
149	21.733964519	137.158.127.114	10.46.0.9	GTP <SIP/SDP>	977 Status: 200 OK (INVITE) |
152	21.919717873	10.46.0.9	137.158.127.114	GTP <SIP>	586 Request: ACK sip:testuser1@196.42.112.97:63737 |
153	21.919875654	137.158.127.114	196.42.112.97	SIP	614 Request: ACK sip:testuser1@196.42.112.97:63737 |

Code snippet
==========
```
#!/usr/bin/env python3

import subprocess
import time
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
from datetime import datetime
import numpy as np

plt.rcParams['font.family'] = 'DejaVu Sans'
plt.rcParams['font.size'] = 10
plt.rcParams['figure.dpi'] = 100
plt.rcParams['savefig.dpi'] = 300
plt.rcParams['savefig.bbox'] = 'tight'

# Configuration
OUTPUT_DIR = "/home/bluelabuser/vonr_multiparameter_test"
MODEL_DIR = "/home/bluelabuser/model"
TEST_DURATION = 30
CAPTURE_INTERFACE = "ogstun"
UE2_IP = "10.46.0.47"

# SUDO PASSWORD
SUDO_PASSWORD = ""

# Single impairment tests
SINGLE_TESTS = {
    "loss": [0, 5, 10, 15, 20],
    "jitter": [0, 25, 50, 75, 100],
    "latency": [0, 50, 100, 150, 200]
}

# Combined tests: pairs of [loss%, jitter_ms, latency_ms]
# Format: (loss, jitter, latency, description)
COMBINED_TESTS = [
    (2, 20, 50, "Light combined"),
    (5, 30, 75, "Moderate combined"),
    (10, 40, 100, "Heavy combined"),
    (15, 50, 150, "Severe combined"),
    (5, 0, 100, "Loss + Latency"),
    (0, 50, 100, "Jitter + Latency"),
    (5, 50, 0, "Loss + Jitter"),
    (10, 100, 0, "High Loss + High Jitter"),
    (2, 20, 200, "Light Loss + High Latency"),
]

# Full matrix: test all combinations
```

```
MATRIX_TESTS = {
    "loss": [0, 5, 10],
    "jitter": [0, 25, 50],
    "latency": [0, 50, 100]
}

# Create output directory
Path(OUTPUT_DIR).mkdir(parents=True, exist_ok=True)
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")

# Calculate totals
single_total = sum(len(v) for v in SINGLE_TESTS.values())
combined_total = len(COMBINED_TESTS)
matrix_total = len(MATRIX_TESTS['loss']) * len(MATRIX_TESTS['jitter']) *
len(MATRIX_TESTS['latency'])
# Subtract 1 from matrix because baseline (0,0,0) is already in single tests
matrix_total_unique = matrix_total - 1

total_tests = single_total + combined_total + matrix_total_unique
total_time = total_tests * TEST_DURATION / 60

print("="*70)
print("VoNR COMPREHENSIVE MULTI-PARAMETER QUALITY TEST")
print("="*70)
print()
print(f"Configuration:")
print(f"  UE2 (WiFi) IP: {UE2_IP}")
print(f"  Test Duration: {TEST_DURATION}s per sample")
print(f"  Capture Interface: {CAPTURE_INTERFACE}")
print(f"  Output: {OUTPUT_DIR}/")
print()
print(f"Test Plan:")
print(f"  PHASE 1 - Single Parameters: {single_total} tests")
print(f"     Packet Loss: {SINGLE_TESTS['loss']}")
print(f"     Jitter: {SINGLE_TESTS['jitter']}")
print(f"     Latency: {SINGLE_TESTS['latency']}")
print()
print(f"  PHASE 2 - Combined Parameters: {combined_total} tests")
for loss, jitter, latency, desc in COMBINED_TESTS[:5]:
    print(f"    • {desc}: L={loss}%, J={jitter}ms, D={latency}ms")
if len(COMBINED_TESTS) > 5:
    print(f"    ... and {len(COMBINED_TESTS)-5} more")
print()
print(f"  PHASE 3 - Full Matrix: {matrix_total_unique} tests")
print(f"    Loss: {MATRIX_TESTS['loss']}")
print(f"    Jitter: {MATRIX_TESTS['jitter']}")
print(f"    Latency: {MATRIX_TESTS['latency']}")
print()
print(f"Grand Total: {total_tests} tests, ~{total_time:.1f} minutes")
print()
print("Prerequisites:")
print("  ✓ Active call running")
```

```
print("  ✓ Reference audio looping")
print("  ✓ Call must stay up for entire duration!")
print()
input("Press ENTER to start comprehensive multi-parameter test...")


def run_sudo_command(cmd, timeout=60):
    """Run a sudo command with password if needed"""
    if SUDO_PASSWORD:
        full_cmd = f"echo '{SUDO_PASSWORD}' | sudo -S {cmd}"
    else:
        full_cmd = f"sudo {cmd}"
    return subprocess.run(full_cmd, shell=True, capture_output=True, text=True, timeout=timeout)


def capture_rtp_traffic(duration, output_pcap):
    """Capture RTP traffic destined to UE2 with progress indicator"""
    tcpdump_cmd = f"timeout {duration + 5} sudo tcpdump -i {CAPTURE_INTERFACE} -w
{output_pcap} 'udp and portrange 10000-60000 and dst host {UE2_IP}' 2>/dev/null &"
    subprocess.run(tcpdump_cmd, shell=True)

    for i in range(duration):
        time.sleep(1)
        if (i + 1) % 5 == 0:
            print(".", end="", flush=True)

    time.sleep(2)
    subprocess.run("sudo pkill -f 'tcpdump.*ogstun'", shell=True, stderr=subprocess.DEVNULL)
    time.sleep(1)

    if os.path.exists(output_pcap):
        size = os.path.getsize(output_pcap)
        if size > 10000:
            print(f" ✓ {size/1024:.0f}KB")
            return True
        else:
            print(f" ✗ Too small")
            return False
    else:
        print(" ✗ Failed")
        return False


def extract_audio_from_pcap(pcap_file, test_name):
    """Extract audio from PCAP file"""
    try:
        pcap_size = os.path.getsize(pcap_file)
        if pcap_size < 10000:
            return False

        port_cmd = f"tshark -r {pcap_file} -T fields -e udp.dstport 2>&1 | grep -v '^Running' | grep -v
'^$' | sort -u | grep -E '^[0-9]+' | head -5"
```

```
        result = subprocess.run(port_cmd, shell=True, capture_output=True, text=True)
        ports = [p for p in result.stdout.strip().split('\n') if p and p.isdigit()]

        if not ports:
            return False

        decode_opts = " ".join([f"-d udp.port=={port},rtp" for port in ports])

        ssrc_cmd = f"tshark -r {pcap_file} {decode_opts} -T fields -e rtp.ssrc 2>/dev/null | sort -u |
grep -v '^$' | head -1"
        result = subprocess.run(ssrc_cmd, shell=True, capture_output=True, text=True)
        ssrc = result.stdout.strip()

        if not ssrc:
            return False

        payload_cmd = f"tshark -r {pcap_file} {decode_opts} -Y 'rtp.ssrc=={ssrc}' -T fields -e
rtp.payload 2>/dev/null | xxd -r -p > /tmp/stream.raw"
        subprocess.run(payload_cmd, shell=True, timeout=60)

        if not os.path.exists('/tmp/stream.raw') or os.path.getsize('/tmp/stream.raw') == 0:
            subprocess.run("rm /tmp/stream.raw 2>/dev/null", shell=True)
            return False

        target_size = 30 * 8000
        raw_size = os.path.getsize('/tmp/stream.raw')
        if raw_size > target_size:
            subprocess.run(f"tail -c {target_size} /tmp/stream.raw > /tmp/stream_tail.raw", shell=True)
            subprocess.run("mv /tmp/stream_tail.raw /tmp/stream.raw", shell=True)

        output_file = f"{OUTPUT_DIR}/{test_name}.wav"
        ffmpeg_cmd = f"ffmpeg -f mulaw -ar 8000 -ac 1 -i /tmp/stream.raw {output_file} -y"
        subprocess.run(ffmpeg_cmd, shell=True, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)

        subprocess.run("rm /tmp/stream.raw 2>/dev/null", shell=True)

        return os.path.exists(output_file)

    except Exception as e:
        subprocess.run("rm /tmp/stream.raw 2>/dev/null", shell=True)
        return False


def calculate_mos(ref_file, deg_file):
    """Calculate MOS using ViSQOL"""
    try:
        tflite_model =
f"{MODEL_DIR}/lattice_tcditugenmeetpackhref_ls2_nl60_lr12_bs2048_learn.005_ep2400_train1
_7_raw.tflite"

        cmd = f"visqol --reference_file {ref_file} --degraded_file {deg_file}"
```

```
        cmd += f" --use_speech_mode --similarity_to_quality_model {tflite_model}"

        result = subprocess.run(cmd, shell=True, capture_output=True, text=True, timeout=120)

        for line in result.stdout.split('\n'):
            if "MOS-LQO:" in line:
                return float(line.split()[1])

        return None
    except Exception as e:
        return None


def apply_impairments(loss=0, jitter=0, latency=0):
    """Apply multiple network impairments simultaneously"""
    run_sudo_command("tc qdisc del dev ogstun root 2>/dev/null")
    time.sleep(1)

    if loss == 0 and jitter == 0 and latency == 0:
        return "baseline"

    # Build tc netem command with all impairments
    cmd_parts = ["tc qdisc add dev ogstun root netem"]

    if latency > 0:
        if jitter > 0:
            cmd_parts.append(f"delay {latency}ms {jitter}ms distribution normal")
        else:
            cmd_parts.append(f"delay {latency}ms")
    elif jitter > 0:
        # If jitter without base latency, add small base delay
        cmd_parts.append(f"delay 10ms {jitter}ms distribution normal")

    if loss > 0:
        cmd_parts.append(f"loss {loss}%")

    cmd = " ".join(cmd_parts)
    run_sudo_command(cmd)

    return f"L{loss}_J{jitter}_D{latency}"


def run_single_test(loss, jitter, latency, label, test_num, total, reference_file):
    """Run a single test with specified impairments"""

    print(f"[{test_num}/{total}] {label}...", end=" ", flush=True)

    # Apply impairments
    impairment_name = apply_impairments(loss, jitter, latency)

    # Capture
    pcap_file = f"/tmp/capture_{impairment_name}.pcap"
```

```
if not capture_rtp_traffic(TEST_DURATION, pcap_file):
    print(f"      Capture failed")
    return {
        "loss": loss, "jitter": jitter, "latency": latency,
        "label": label, "mos": None, "status": "capture_failed",
        "test_type": "unknown"
    }

# Extract audio
audio_file = f"test_{impairment_name}"
if not extract_audio_from_pcap(pcap_file, audio_file):
    print(f"     Audio extraction failed")
    os.remove(pcap_file)
    return {
        "loss": loss, "jitter": jitter, "latency": latency,
        "label": label, "mos": None, "status": "extraction_failed",
        "test_type": "unknown"
    }

os.remove(pcap_file)

# Calculate MOS
degraded_file = f"{OUTPUT_DIR}/{audio_file}.wav"

if loss == 0 and jitter == 0 and latency == 0:
    # This is the baseline
    mos = calculate_mos(degraded_file, degraded_file)
    mos_str = f"{mos:.3f}" if mos else "N/A"
    print(f"    Reference. MOS: {mos_str}")
    return {
        "loss": loss, "jitter": jitter, "latency": latency,
        "label": label, "mos": mos, "status": "success",
        "reference_file": degraded_file, "test_type": "baseline"
    }
else:
    if reference_file:
        mos = calculate_mos(reference_file, degraded_file)
        mos_str = f"{mos:.3f}" if mos else "N/A"
        print(f"    MOS: {mos_str}")
        return {
            "loss": loss, "jitter": jitter, "latency": latency,
            "label": label, "mos": mos,
            "status": "success" if mos else "mos_failed",
            "test_type": "unknown"
        }
    else:
        print(f"     No reference")
        return {
            "loss": loss, "jitter": jitter, "latency": latency,
            "label": label, "mos": None, "status": "no_reference",
            "test_type": "unknown"
        }
```

```
#
================================================================================
======
# MAIN EXECUTION
#
================================================================================
======

all_results = []
start_time = time.time()
reference_file = None
test_counter = 1

# PHASE 1: Single parameter tests
print(f"\n{'='*70}")
print(f"PHASE 1: SINGLE PARAMETER TESTING")
print(f"{'='*70}\n")

for imp_type, values in SINGLE_TESTS.items():
    print(f"\nTesting {imp_type.upper()}:")
    print("-" * 70)

    for value in values:
        if imp_type == "loss":
            loss, jitter, latency = value, 0, 0
        elif imp_type == "jitter":
            loss, jitter, latency = 0, value, 0
        else:
            loss, jitter, latency = 0, 0, value

        unit = "%" if imp_type == "loss" else "ms"
        label = "Baseline" if value == 0 else f"{imp_type.title()} {value}{unit}"

        result = run_single_test(loss, jitter, latency, label,
                                 test_counter, total_tests, reference_file)

        result['test_type'] = 'baseline' if value == 0 else 'single'

        if value == 0 and 'reference_file' in result:
            reference_file = result['reference_file']

        all_results.append(result)
        test_counter += 1

print(f"\n√ Phase 1 complete. Taking 5s break...\n")
time.sleep(5)

# PHASE 2: Combined parameter tests
print(f"\n{'='*70}")
print(f"PHASE 2: COMBINED PARAMETER TESTING")
```

```
print(f"{'='*70}\n")

for loss, jitter, latency, desc in COMBINED_TESTS:
    result = run_single_test(loss, jitter, latency, desc,
                             test_counter, total_tests, reference_file)
    result['test_type'] = 'combined'
    all_results.append(result)
    test_counter += 1

print(f"\n√ Phase 2 complete. Taking 5s break...\n")
time.sleep(5)

# PHASE 3: Full matrix tests
print(f"\n{'='*70}")
print(f"PHASE 3: FULL MATRIX TESTING")
print(f"{'='*70}\n")

for loss in MATRIX_TESTS['loss']:
    for jitter in MATRIX_TESTS['jitter']:
        for latency in MATRIX_TESTS['latency']:
            # Skip baseline as it was already done in Phase 1
            if loss == 0 and jitter == 0 and latency == 0:
                continue

            # Skip if already tested in single parameter phase
            already_tested = False
            if (loss > 0 and jitter == 0 and latency == 0) or \
               (loss == 0 and jitter > 0 and latency == 0) or \
               (loss == 0 and jitter == 0 and latency > 0):
                if (loss in SINGLE_TESTS.get('loss', []) and jitter == 0 and latency == 0) or \
                   (jitter in SINGLE_TESTS.get('jitter', []) and loss == 0 and latency == 0) or \
                   (latency in SINGLE_TESTS.get('latency', []) and loss == 0 and jitter == 0):
                    already_tested = True

            if already_tested:
                continue

            label = f"L{loss}% J{jitter}ms D{latency}ms"

            result = run_single_test(loss, jitter, latency, label,
                                     test_counter, total_tests, reference_file)
            result['test_type'] = 'matrix'
            all_results.append(result)
            test_counter += 1

# Remove impairments
run_sudo_command("tc qdisc del dev ogstun root 2>/dev/null")

elapsed = time.time() - start_time
```

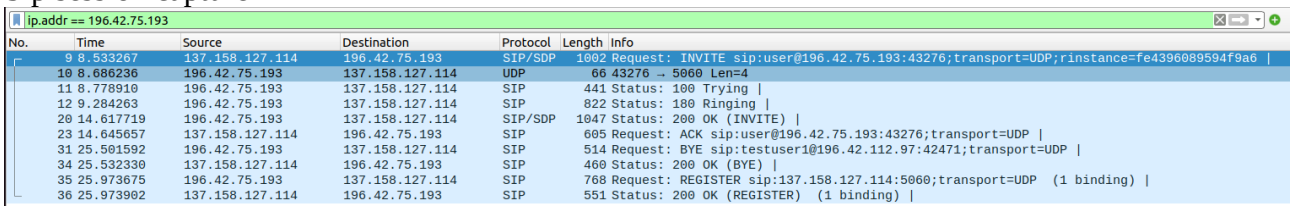
```
#
================================================================================
======
# SAVE RESULTS
#
================================================================================
======

print("\n" + "="*70)
print("SAVING RESULTS")
print("="*70)

df = pd.DataFrame(all_results)
csv_path = f"{OUTPUT_DIR}/results_{timestamp}.csv"
df.to_csv(csv_path, index=False)
print(f"√ CSV saved: {csv_path}")

#
================================================================================
======
# GENERATE GRAPHS
#
================================================================================
======

print("\nGenerating graphs...")

success_df = df[df['status'] == 'success']

# GRAPH 1: Single parameter impacts (3-panel)
print("  Creating Graph 1: Single parameter impacts...")
fig1, axes = plt.subplots(1, 3, figsize=(18, 5))

for idx, imp_type in enumerate(["loss", "jitter", "latency"]):
    ax = axes[idx]
    data = success_df[success_df['test_type'].isin(['baseline', 'single'])].copy()

    # Filter for this parameter only
    other_params = [p for p in ['loss', 'jitter', 'latency'] if p != imp_type]
    data = data[(data[other_params[0]] == 0) & (data[other_params[1]] == 0)]
    data = data.sort_values(imp_type)

    if len(data) > 0:
        x = data[imp_type]
        y = data['mos']
        ax.plot(x, y, 'o-', linewidth=2, markersize=8, color='#2E86AB')
        ax.fill_between(x, y, 1, alpha=0.2, color='#2E86AB')

        ax.set_xlabel(f"{imp_type.title()} ({'%' if imp_type == 'loss' else 'ms'})",
                      fontsize=12, fontweight='bold')
        ax.set_ylabel('MOS Score', fontsize=12, fontweight='bold')
        ax.set_title(f'Impact of {imp_type.title()}', fontsize=14, fontweight='bold')
```

```
        ax.set_ylim(1, 5)
        ax.grid(True, alpha=0.3)

        # Quality zones
        ax.axhspan(4.5, 5, alpha=0.1, color='green')
        ax.axhspan(4.0, 4.5, alpha=0.1, color='lightgreen')
        ax.axhspan(3.5, 4.0, alpha=0.1, color='yellow')
        ax.axhspan(3.0, 3.5, alpha=0.1, color='orange')
        ax.axhspan(1, 3.0, alpha=0.1, color='red')

        # Value labels
        for xi, yi in zip(x, y):
            if yi and not pd.isna(yi):
                ax.annotate(f'{yi:.2f}', (xi, yi), textcoords="offset points",
                            xytext=(0,8), ha='center', fontsize=8)

plt.suptitle(f'Single Parameter Impact on VoNR Quality', fontsize=16, fontweight='bold', y=1.02)
plt.tight_layout()
graph1_path = f"{OUTPUT_DIR}/graph1_single_params_{timestamp}.png"
plt.savefig(graph1_path, dpi=300, bbox_inches='tight')
print(f"    ✓ Saved: {graph1_path}")
plt.close()

# GRAPH 2: Combined parameter tests (horizontal bar chart)
print("  Creating Graph 2: Combined parameter tests...")
fig2, ax = plt.subplots(figsize=(14, 8))

combined_data = success_df[success_df['test_type'] == 'combined'].copy()
if len(combined_data) > 0:
    combined_data = combined_data.sort_values('mos', ascending=True)

    colors = plt.cm.RdYlGn(combined_data['mos'] / 5.0)

    bars = ax.barh(range(len(combined_data)), combined_data['mos'], color=colors)
    ax.set_yticks(range(len(combined_data)))
    ax.set_yticklabels(combined_data['label'], fontsize=10)
    ax.set_xlabel('MOS Score', fontsize=12, fontweight='bold')
    ax.set_title('Combined Impairments - MOS Scores', fontsize=14, fontweight='bold')
    ax.set_xlim(1, 5)
    ax.grid(True, alpha=0.3, axis='x')

    # Add value labels
    for i, (idx, row) in enumerate(combined_data.iterrows()):
        ax.text(row['mos'] + 0.05, i, f"{row['mos']:.2f}",
                va='center', fontsize=9, fontweight='bold')

    # Add parameter labels
    for i, (idx, row) in enumerate(combined_data.iterrows()):
        param_text = f"L:{row['loss']}% J:{row['jitter']}ms D:{row['latency']}ms"
        ax.text(0.05, i, param_text, va='center', fontsize=7,
                color='white', fontweight='bold',
                bbox=dict(boxstyle='round', facecolor='black', alpha=0.6))
```

```
plt.tight_layout()
graph2_path = f"{OUTPUT_DIR}/graph2_combined_{timestamp}.png"
plt.savefig(graph2_path, dpi=300, bbox_inches='tight')
print(f"    ✓ Saved: {graph2_path}")
plt.close()

# GRAPH 3: Full matrix heatmaps (3-panel showing interactions)
print("  Creating Graph 3: Full matrix heatmaps...")
fig3, axes = plt.subplots(1, 3, figsize=(20, 6))

matrix_data = success_df[success_df['test_type'].isin(['baseline', 'single', 'matrix'])].copy()

pairs = [
    ('loss', 'jitter', 'latency', 0, '%', 'ms'),
    ('loss', 'latency', 'jitter', 1, '%', 'ms'),
    ('jitter', 'latency', 'loss', 2, 'ms', 'ms')
]

for x_param, y_param, fixed_param, idx, x_unit, y_unit in pairs:
    ax = axes[idx]

    # Use middle value of fixed parameter
    fixed_vals = sorted(matrix_data[fixed_param].unique())
    if len(fixed_vals) > 0:
        fixed_val = fixed_vals[len(fixed_vals)//2]

        subset = matrix_data[matrix_data[fixed_param] == fixed_val]

        if len(subset) > 1:
            pivot = subset.pivot_table(values='mos', index=y_param, columns=x_param,
aggfunc='mean')

            sns.heatmap(pivot, annot=True, fmt='.2f', cmap='RdYlGn',
                        vmin=1, vmax=5, ax=ax, cbar_kws={'label': 'MOS Score'})

            ax.set_xlabel(f"{x_param.title()} ({x_unit})", fontsize=11, fontweight='bold')
            ax.set_ylabel(f"{y_param.title()} ({y_unit})", fontsize=11, fontweight='bold')

            fixed_unit = '%' if fixed_param == 'loss' else 'ms'
            ax.set_title(f'{x_param.title()} vs {y_param.title()}\n(Fixed:
{fixed_param.title()}={fixed_val}{fixed_unit})',
                         fontsize=12, fontweight='bold')

plt.suptitle(f'Multi-Parameter Interaction Analysis', fontsize=16, fontweight='bold', y=1.02)
plt.tight_layout()
graph3_path = f"{OUTPUT_DIR}/graph3_matrix_{timestamp}.png"
plt.savefig(graph3_path, dpi=300, bbox_inches='tight')
print(f"    ✓ Saved: {graph3_path}")
plt.close()

print("✓ All graphs generated")
```

```
#
================================================================================
======
# SUMMARY
#
================================================================================
======

print("\n" + "="*70)
print("TEST COMPLETE!")
print("="*70)
print(f"\nTotal time: {elapsed/60:.1f} minutes")
print(f"Tests completed: {len(success_df)}/{len(df)}")
print(f"\nOutput directory: {OUTPUT_DIR}/")
print(f"  - results_{timestamp}.csv")
print(f"  - graph1_single_params_{timestamp}.png")
print(f"  - graph2_combined_{timestamp}.png")
print(f"  - graph3_matrix_{timestamp}.png")
print(f"  - {len(df)} WAV files")

# Show statistics by phase
print("\n" + "-"*70)
print("RESULTS BY PHASE:")
print("-" * 70)

for phase_type, phase_name in [('single', 'Single Parameters'),
                               ('combined', 'Combined Tests'),
                               ('matrix', 'Matrix Tests')]:
    phase_data = success_df[success_df['test_type'] == phase_type]
    if len(phase_data) > 0:
        avg_mos = phase_data['mos'].mean()
        min_mos = phase_data['mos'].min()
        max_mos = phase_data['mos'].max()
        print(f"{phase_name:20s}: {len(phase_data):2d} tests | Avg MOS: {avg_mos:.3f} | Range:
{min_mos:.3f} - {max_mos:.3f}")

# Show top/bottom results overall
print("\n" + "-"*70)
print("BEST QUALITY (Top 5):")
for idx, row in success_df.nlargest(5, 'mos').iterrows():
    print(f"  {row['label']:35s} MOS: {row['mos']:.3f} | L:{row['loss']:2.0f}% J:{row['jitter']:3.0f}ms
D:{row['latency']:3.0f}ms")

print("\nWORST QUALITY (Bottom 5):")
for idx, row in success_df.nsmallest(5, 'mos').iterrows():
    print(f"  {row['label']:35s} MOS: {row['mos']:.3f} | L:{row['loss']:2.0f}% J:{row['jitter']:3.0f}ms
D:{row['latency']:3.0f}ms")

print("="*70)
```

Sip session capture



ip.addr == 196.42.75.193

No.	Time	Source	Destination	Protocol	Length	Info
9	8.533267	137.158.127.114	196.42.75.193	SIP/SDP	1002	Request: INVITE sip:user@196.42.75.193:43276;transport=UDP;rinstance=fe4396089594f9a6 |
10	8.686236	196.42.75.193	137.158.127.114	UDP	66	43276 → 5060 Len=4
11	8.778910	196.42.75.193	137.158.127.114	SIP	441	Status: 100 Trying |
12	9.284263	196.42.75.193	137.158.127.114	SIP	822	Status: 180 Ringing |
20	14.617719	196.42.75.193	137.158.127.114	SIP/SDP	1047	Status: 200 OK (INVITE) |
23	14.645657	137.158.127.114	196.42.75.193	SIP	605	Request: ACK sip:user@196.42.75.193:43276;transport=UDP |
31	25.501592	196.42.75.193	137.158.127.114	SIP	514	Request: BYE sip:testuser1@196.42.112.97:42471;transport=UDP |
34	25.532330	137.158.127.114	196.42.75.193	SIP	460	Status: 200 OK (BYE) |
35	25.973675	196.42.75.193	137.158.127.114	SIP	768	Request: REGISTER sip:137.158.127.114:5060;transport=UDP (1 binding) |
36	25.973902	137.158.127.114	196.42.75.193	SIP	551	Status: 200 OK (REGISTER) (1 binding) |