



Aufbau eines 5G SA Netzwerks

Mobile Netze

Prof. Dr. Alf Zugenmaier

September 2024

Bennet Kiessling, Sarah König, Chiara Piccolroaz, Johann Rittenschober, Marina Trinz

Inhaltsverzeichnis

1 Einleitung (Bennet Kießling)	3
2 Theorie	3
2.1 5G allgemein (Marina Trinz)	4
2.2 Erklärung der Architektur (Johann Rittenschober)	5
3 Projektaufbau	6
3.1 Open5Gs und srsRAN Aufbau (Sarah König)	6
3.1.1 Installation und Initiales Setup Open5Gs (Sarah König)	7
3.1.2 Installation und Initiales Setup srsRAN (Sarah König)	7
3.1.3 Konfigurationsmodifikationen und Versuche (Sarah König)	8
3.2 Verteiltes Netzwerk mit Open5Gs und srsRAN (Johann Rittenschober)	10
3.2.1 Verteilung auf zwei verschiedene PCs (Johann Rittenschober)	11
3.2.2 Verteilung auf drei verschiedene PCs (Johann Rittenschober)	11
3.3 User Equipments	12
3.3.1 Theorie (Marina Trinz)	12
3.3.2 Open cells SIM-Karte (Marina Trinz)	13
3.3.3 Quectel UE (Chiara Piccolroaz)	15
3.3.4 srsUE - Installation (Johann Rittenschober)	17
3.3.5 srsUE - Konfiguration (Sarah König)	18
3.3.6 Handy (Chiara Piccolroaz)	19
4 Analyse der durchgeführten Versuche	21
4.1 Spektrum-Analyse (Johann Rittenschober)	21
4.2 Wireshark (Chiara Piccolroaz)	23
4.3 File-Sharing Service (Sarah König)	24
4.4 NetCat (Chiara Piccolroaz)	25
4.5 Netzwerktest durch Simulation von Netzwerkverkehr (Bennet Kießling)	26
4.5.1 Getestete Parameter (Bennet Kießling)	26
4.5.2 Getestete Komponenten (Bennet Kießling)	27
4.5.3 iPerf3 (Bennet Kießling)	27
4.5.4 Ergebnisse (Bennet Kießling und Johann Rittenschober)	27
5 Erfolglose Tests	30
5.1 Handover	30
5.1.1 Theorie (Marina Trinz)	30

5.1.2	Inter Handover (Chiara Piccolroaz und Sarah König)	31
5.1.3	Intra Handover (Marina Trinz)	32
5.2	SMS (Chiara Piccolroaz)	33
6	Diskussion (Alle)	33
7	Fazit (Alle)	35
A	srsUE Installation Guide (Johann Rittenschober)	39
B	Spectrum Analyser using GNU Radio (Johann Rittenschober)	39

1 Einleitung (Bennet Kießling)

Die Einführung von 5G-Technologie markiert einen entscheidenden Meilenstein in der Evolution mobiler Netzwerke, die weit über die Kapazitäten von LTE hinausgeht. Mit einer deutlich gesteigerten Datenrate, extrem niedriger Latenz und der Möglichkeit, Millionen von Geräten zu vernetzen, bietet 5G ein enormes Potenzial für zahlreiche Anwendungen, von autonomen Fahrzeugen bis hin zum Internet der Dinge (IoT) [15]. Der Aufbau eines 5G Standalone (SA) Netzwerks ist jedoch eine komplexe Herausforderung, die nicht nur spezialisierte Softwarelösungen, sondern auch maßgeschneiderte Hardwarekonfigurationen erfordert. In dieser Arbeit untersuchen wir den Aufbau eines solchen 5G SA Netzwerks mithilfe von Open5GS und SRS RAN, sowie die Integration von User Equipments (UEs) für Testzwecke.

Ziel dieser Arbeit ist es, ein funktionierendes 5G Standalone Netzwerk zu errichten und verschiedene UEs, darunter kommerzielle Geräte und spezialisierte Test-Hardware wie das Quectel-UE oder das SRSUE, zu integrieren. Wir stellen uns die Frage, welche Herausforderungen beim Aufbau eines solchen Netzwerks auftreten und wie gut die unterschiedlichen UEs mit der 5G-Infrastruktur interagieren. Besondere Schwerpunkte liegen auf der Konfiguration der Core Netzwerkkomponenten und der Analyse der Verbindungsqualität zwischen den UEs und dem Netz. Dabei soll auch untersucht werden, welche Hindernisse, wie z.B. Software und Hardware Inkompatibilitäten, im Verlauf des Projekts auftreten und wie diese gelöst werden konnten.

Im Fokus dieser Arbeit steht die praktische Umsetzung eines 5G SA Netzwerks unter Verwendung von Open5GS als Core und SRS RAN für das Radio Access Network. Dabei greifen wir auf eine Kombination von Open-Source-Software, kommerziellen UEs und spezialisierter Testhardware zurück, um eine möglichst realitätsnahe Umgebung zu schaffen. Der Einsatz von Tools wie GNU Radio und Wireshark ermöglicht eine detaillierte Analyse der Signalverarbeitung und des Verbindungsablaufs.

Unser Vorgehen gliedert sich in mehrere Schritte: Zunächst wird die theoretische Grundlage zu 5G und den verwendeten Softwarelösungen Open5GS und SRS RAN dargelegt. Daraufhin folgt der detaillierte Aufbau des Netzwerks, einschließlich der Konfiguration der verschiedenen pcapUEs und der Durchführung von Tests zur Verbindungsqualität und Netzwerkstabilität. Die gesammelten Daten werden anschließend analysiert und diskutiert, wobei sowohl erfolgreiche als auch gescheiterte Versuche betrachtet werden.

2 Theorie

Für die Realisierung eines 5G-Netzes sind fundierte Kenntnisse der Grundlagen der mobilen Kommunikationssysteme notwendig. Besonders wichtig ist es, die Unterschiede zwischen 5G-Netzen und früheren Versionen wie Long Term Evolution (LTE) zu verstehen. Die theoretischen Grund-

lagen, die für die Umsetzung des Projekts wichtig waren, werden in diesem Kapitel erläutert.

2.1 5G allgemein (Marina Trinz)

5G, die fünfte Generation des Mobilfunkstandards, revolutioniert die drahtlose Kommunikation durch extrem hohe Datenraten, minimale Latenzzeiten und die Fähigkeit, eine Vielzahl von Geräten gleichzeitig zu vernetzen [1]. Dies wird durch neue Netzkomponenten und Funktionen von 5G ermöglicht. Diese Komponenten und die allgemeine Architektur von 5G Kernnetz (vgl. Abbildung 1) werden in diesem Abschnitt erläutert. Da im Rahmen dieses Projekts nur die Standalone (SA) Version von 5G umgesetzt wird, wird die Non-Standalone Architektur nicht näher behandelt.

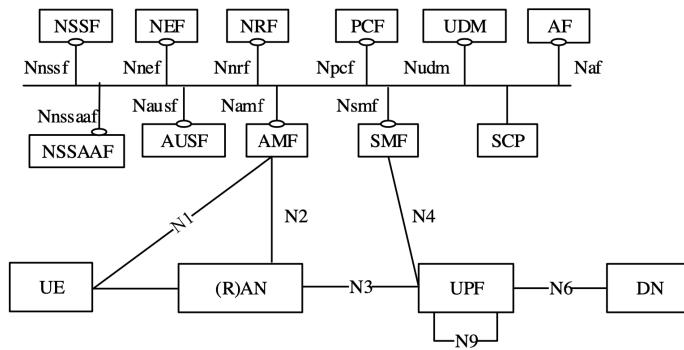


Abbildung 1: 5G Standalone Architektur - Service Based Representation (Quelle [3, p. 31]).

Zu den wichtigsten Komponenten des 5G-Kernnetzes, die in diesem Projekt verwendet wurden, gehören:

AMF	Access and Mobility Management Function
AUSF	Authentication Server Funktion
SMF	Session Management Function
UPF	User Plane Function
PCF	Policy Control Function
UDM	Unified Data Management
UDR	Unified Data Repository

Tabelle 1: 7 wichtigste Komponenten des 5G-Kernnetzes.

Die AMF ist zentral für die Verwaltung von Verbindungen im 5G-Netzwerk [43, p. 177]. Sie nutzt NAS-Signalsierung, um Verbindungen mit Benutzergeräten (UE) herzustellen und verwaltet alle Verbindungsanfragen. Außerdem leitet sie Nachrichten zum Sitzungsmanagement an die SMF weiter.

Die AUSF ist eine wesentliche Authentifizierungskomponente in 5G-Netzen. Sie ergänzt die Funktionen der AMF, indem sie die Identifikationen der Netzteilnehmer verwaltet und die Authentifizierung der Benutzergeräte durchführt [2]. Zudem verwaltet die AUSF den Root Session

Key, Key Access and User Security Function (KAUSF), der für die Sicherheit der Benutzerdaten entscheidend ist.

Die SMF ist für den Aufbau von Protocol Data Unit (PDU) Sessions zur Kommunikation im Netz verantwortlich [43, p. 177, 184]. Sie kommuniziert nicht direkt mit dem UE, sondern erhält Anfragen zur Sitzungsherstellung über die AMF. Diese Anfragen werden vom UE an die AMF gesendet und dann an die entsprechende SMF weitergeleitet. Die SMF ermöglicht auch die Nutzung mehrerer Instanzen für Network Slicing, um verschiedene Netzwerkdienste effizient zu verwalten.

Die UPF ermöglicht den Aufbau von User Plane Tunneln für jede Kommunikation von einem UE und sorgt für die Weiterleitung von Nutzdatenpaketen [43, p. 177, 184].

Die PCF verwaltet die Nutzung von Netzwerkressourcen, setzt Prioritäten der Kommunikationskanäle bei Überlast und überwacht die Quality of Service (QoS) für Datenströme [43, p. 178].

Das UDM ist eine zentrale Authentifizierungskomponente in 5G-Netzen, die für die Verwaltung von Teilnehmerdaten verantwortlich ist [43, p. 178]. Es bearbeitet Anfragen von der SMF und sorgt so für eine effiziente Datenverwaltung.

Die UDR realisiert die Speicherung von Daten wie Netzteilnehmerdaten und Netznutzungsdaten [43, p. 178]. Dafür kommuniziert sie mit dem UDM und der PCF (vgl. Abbildung 2).

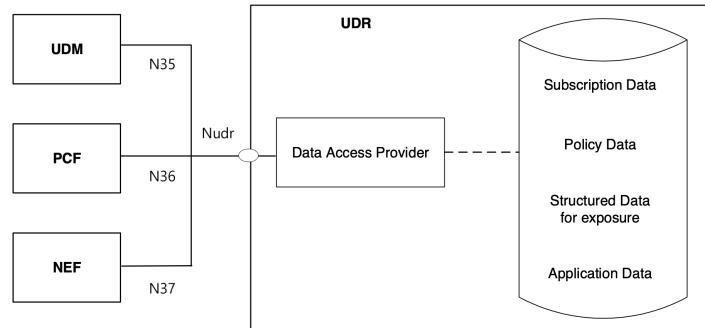


Abbildung 2: Datenspeicherarchitektur (Quelle [3, p. 38]).

2.2 Erklärung der Architektur (Johann Rittenschober)

Die Architektur eines 5G SA Netzes, mit den einzelnen bereits in Abbildung 1 beschriebenen modularen Komponenten, lässt sich gruppieren in das Kernnetz (Core) und das Radio Access Network (RAN). Letzteres besteht aus sog. Nodes, also Funkzellen die am Kernnetz angeschlossen sind und eine Drahtlos-Verbindung zu Endgeräten (UE) herstellt. Im 5G heißen diese gNodeB (gNB), veranschaulicht in Abbildung 3a.

Die skizzierten ng-eNB Nodes stellen im 5G Netz ein LTE Interface bereit, sind in diesem Projekt jedoch nicht von Bedeutung. Eine gNB kann zusätzlich in ein Central Unit (CU) und

mehrere Distributed Units (DU) unterteilt werden, die gNB bleibt dabei aber eine logische Einheit (vgl. Abbildung 3b).

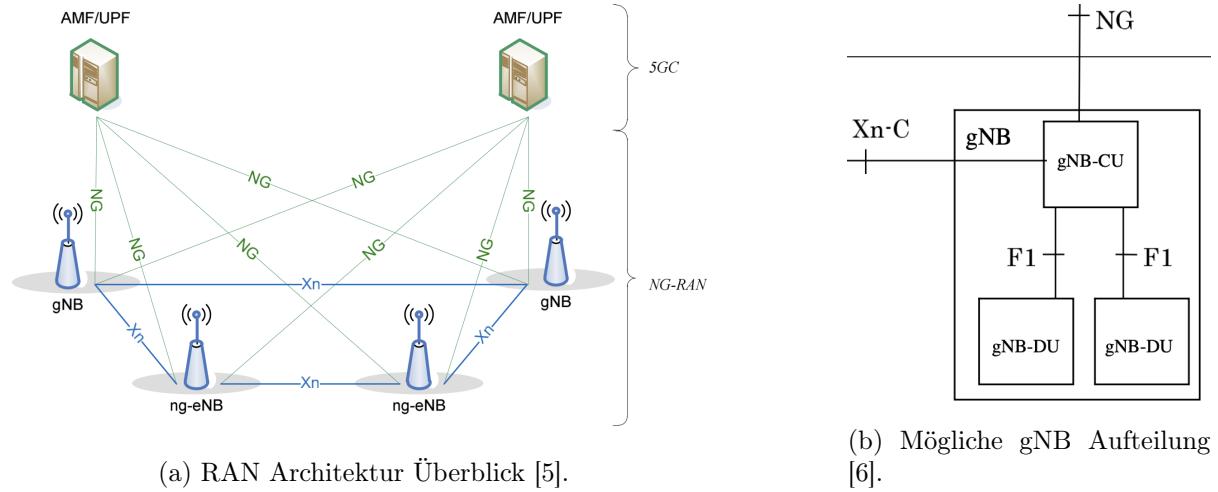


Abbildung 3: 5G RAN Architektur.

Die beschriebene Architektur wird in diesem Projekt konkret umgesetzt wie in Abbildung 4 veranschaulicht. Im folgenden Kapitel wird dieser Projektaufbau detailliert ausgeführt.

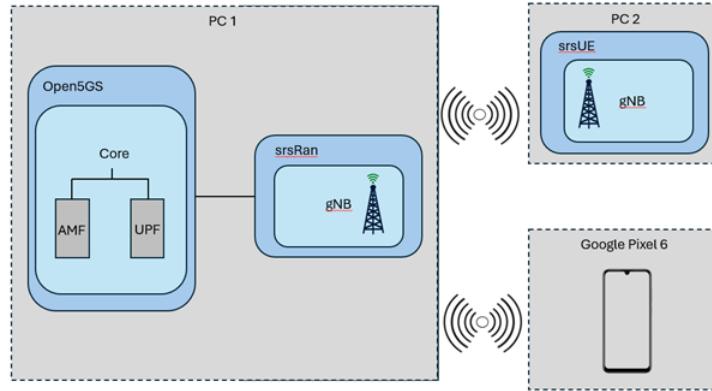


Abbildung 4: Aufbau Testbed (Bennet Kießling).

3 Projektaufbau

In diesem Kapitel werden alle Installationen beschrieben, die für die Realisierung eines 5G-Netzes benötigt wurden.

3.1 Open5Gs und srsRAN Aufbau (Sarah König)

Wie in Abschnitt 2.1 beschrieben, besteht unser 5G Netzwerk aus einem Core-Netzwerk mit Open5Gs und einem RAN-Netzwerk mit srsRAN. Die Installation und Konfiguration dieser werden in den folgenden Abschnitten erklärt.

3.1.1 Installation und Initiales Setup Open5Gs (Sarah König)

Für das Aufsetzen des 5G-Core-Netzwerks wurde den Standalone-Teilen der offiziellen Anleitung gefolgt [18]. Zuerst wurde MongoDB als Datenbank für Subscriber-Informationen und IP-Adressen installiert und gestartet. Anschließend konnte über apt die neueste Open5Gs-Version installiert werden. Während des Installationsprozesses wurde der DNS-Eintrag auf Localhost gesetzt, sodass alle folgenden DNS-Anfragen dorthin weitergeleitet und folglich nicht mehr aufgelöst wurden. Ein Neustart hat das Problem behoben. Open5Gs stellt außerdem eine WebUI `http://localhost:9999` zur Verfügung, welche die einfache Erstellung und Verwaltung von Subscriber-Daten (IMSI, Sicherheitsdaten und APN) ermöglicht. Um diese zu nutzen, wurde zuerst Node.js installiert und anschließend die WebUI mit einem einfachen curl-Befehl.

Für den Standalone-Modus musste die PLMN, bestehend aus MCC und MNC, in der Network Repository Funktion (NRF) und AMF-Konfiguration angepasst werden sowie die TAC Information der AMF. Wir haben die internationale Test-PLMN `001/01` verwendet. Außerdem haben wir zuerst fälschlicherweise die NGAP-Bind Adresse in der AMF-Konfiguration und die GTPU-Bind-Adresse des UPF geändert. Die beiden Adressen wurden später wieder auf die lokalen Adressen zurückgesetzt, da die Core-Komponenten in diesem Setup auf dem gleichen Rechner laufen und sonst nicht gefunden wurden. Nach einem Neustart der modifizierten Komponenten wurden außerdem die Timestamps im Logging richtig konfiguriert. Zuletzt wurde noch die WAN-Verbindung für das Netzwerk über IP-Forwarding und entsprechenden Regeln für die IP-Tables und Firewall konfiguriert, damit verbundene UEs auch Internet-Funktionen in dem 5G-Netzwerk verwenden können.

3.1.2 Installation und Initiales Setup srsRAN (Sarah König)

Für die Installation von srsRAN wurde das srsRAN Projekt basierend auf der offiziellen Anleitung [30] gebaut, da wir uns die Möglichkeit offen lassen wollten, den Source-Code im Laufe der Veranstaltung zu modifizieren. Zuerst wurden dafür die notwendigen Abhängigkeiten und Ccache für schnelleres Kompilieren installiert. UHD war bereits in einer aktuellen Version vorhanden. Bei den RF-Drivers wurde sich gegen die Installation von ZeroMQ entschieden, da diese für uns für die Erstellung eines funktionsfähigen 5G-Netzwerkes nicht notwendig war.

Nach erfolgreicher Vorbereitung des Builds wurde das GitHub-Repository ge-cloned und die enthaltene Code-Basis mit `make` gebaut. Zuletzt wurde der gnb-Befehl über `make install` global verfügbar gemacht.

Für die gNB wird ein USRP B210 mit Antennen für einen Bereich von 791-849 MHz und 1920-2170 MHz als Basisstation und Verbindung zwischen dem Core-Netzwerk und den UEs verwendet. An dieses wurde eine externe Referenzquelle für die Clock und Synchronisation angeschlossen. Für die korrekte Integration des Geräts in das srsRAN-Netzwerk wurde eine neue Konfigurationsdatei

basierend auf zwei Beispiel-Konfigurationen (für ein B210 mit FDD und ein Hardware-UE) aus dem srsRAN-Projekt erstellt und entsprechend modifiziert: Da Band 3 nicht optimal für unsere Antennen war, wurde zu Band 1 gewechselt und die ARFCN entsprechend für geeignete Sendefrequenzen angepasst. Die Bandbreite von 20MHz wurde beibehalten, da das B210 diese unterstützt. Außerdem wurden die IP-Daten für das AMF und die Bind-Adresse des gNBs auf lokale Adressen gesetzt. Der TAC musste auf 1 gesetzt werden, damit das srsRAN und die gNB fehlerfrei starten konnten. Die Modifikationen wurden auf Basis erhaltener Fehlermeldungen vorgenommen, wie zum Beispiel, dass die CU keine Verbindung mit der CP aufbauen konnte.

Basierend auf den finalen Log-Ausgaben der gNB und Core-Komponenten wurden damit das RAN-Netzwerk und die gNB erfolgreich gestartet und die gNB erfolgreich zu dem AMF und somit dem 5G-Core-Netzwerk verbunden.

3.1.3 Konfigurationsmodifikationen und Versuche (Sarah König)

Da das aufgesetzte Netzwerk allerdings weder von dem Quectel-UE, dem srsUE noch verschiedenen 5G fähigen Smartphones gefunden werden konnten, wurden im Anschluss verschiedene Lösungsansätze basierend auf Internetrecherche, Log-Ausgaben und Wireshark-Analysen getestet. Einzelne davon sind im Folgenden genauer beschrieben.

Eine Vermutung war, dass die ausgewählten Antennen, Frequenzen, Bänder und Bandbreiten nicht optimal zusammenpassen oder eine zu niedrige Leistung haben und dadurch die gNB die Daten nicht korrekt senden kann. Durch den Wechsel auf andere Antennen und Konfigurationen haben wir ein stabileres Signal erwartet, dass von einem unserer UEs empfangen werden kann. Zuerst wurden also Antenne im Bereich 1710-1990 MHz [7], 3500 -3700 MHz und explizite 5G-Antennen [11] auf den dazu passenden Frequenzbereichen und Bändern getestet. Die Zuordnung, welche Bänder für welchen Frequenzen am besten sind, wurde einer Tabelle entnommen [45]. Basierend darauf wurden mit einem ARFCN-Frequency Rechner die ARFCN Werte für die gNB berechnet [42]. Da das Netzwerk trotzdem nicht gefunden wurde, wurden anschließend allgemein mit allen Antennen mehrere Bänder (1, 3, 7, 78) mit entsprechenden Frequenzen leider ebenfalls erfolglos getestet.

Analysen mit einem Signalanalyzer haben gezeigt, dass die gNB überhaupt Daten sendet. Nach Untersuchen der Signalstärke mit verschiedenen Antennen und Frequenzen wurde die initiale Konfiguration beibehalten, da das Signal damit recht stark war. Als nächste Ursache wurden einzelne Konfigurationsparameter entsprechend der Configuration Reference betrachtet [29]. Zuerst wurde die PLMN-Einstellung getestet. Um auszuschließen, dass das Netzwerk nicht von den offiziellen Handys gefunden wird, da es ein Test-Netzwerk ist, haben wir die MCC und MNC für das deutsche Telefonica / O2-Netzwerk 262/03 ausprobiert. Außerdem wurde noch 001/02 getestet, um Probleme mit anderen Gruppen mit gleicher PLMN-Konfiguration auszuschließen. Beide Änderungen haben keine Verbesserungen gebracht. Die PLMN Daten mussten jeweils in

dem Core-Netzwerk für die AMF und NRF, in der gNB Konfiguration und in den Sim-Karten gesetzt und die Open5Gs Deamons und gNB neu gestartet werden.

Bei 5G-Netzen sind die Clock-Rate und Synchronisation für eine stabile Verbindung sehr wichtig. Deswegen wurden sowohl clock als auch sync in der gNB Konfiguration auf external gesetzt und mit verschiedenen Sampling Rates getestet. Auch wurde auf ein USRP B210 Gerät gewechselt, welches kein GPSDO hat, um sicherzustellen, dass die externe Clock verwendet wird. Das srsUE wurde extern über GPSDO synchronisiert. Eine Verbindung ist trotzdem nicht zustande gekommen, aber Synchronisationsprobleme wurden als Ursache ausgeschlossen.

Basierend auf verschiedenen Beispielen, Dokumentationen und Tutorials wurden verschiedene Konfigurationen ausprobiert. Letztendlich konnte mit dem Wechsel auf Band 3, einer Frequenz von 1842,5 MHz und Hinzufügen verschiedener Konfigurationen wie pdccch und prach ein RRC-Connected mit dem srsUE erreicht werden. Eine erfolgreiche Verbindung ist trotzdem nicht zustande gekommen, da im Anschluss keine PDU Session erzeugt wurde.

Die Analyse der Log-Dateien des gNBs und UEs sowie der Wireshark-Aufzeichnungen hat ergeben, dass das Senden der RRC-Request von dem UE erfolgreich war. Diese ist bei dem gNB angekommen, woraufhin versucht wurde eine RRC-Response zu Senden. Da die gNB darauf kein Acknowledge erhalten hat, wurde der Verbindungsversuch nach einem Timeout als fehlgeschlagen gewertet. Das UE erhält laut Logs die RRC-Response und wertet die Verbindung damit als erfolgreich (RRC-Connected). Das Acknowledge kann aber anscheinend nicht an die gNB gesendet werden. Zusätzlich kam noch die Fehlermeldung „UE has no NGAP-context“, Timeout Nachrichten in den Log-/Wireshark Dateien vor und einzeln sind auch Overflows oder Underflows aufgetreten. Diese Verbindungsprobleme könnten durch Fehler in der Signalisierung zwischen UE und gNB verursacht werden. In Folge wurden die Konfigurationsparameter auf Übereinstimmung geprüft und verschiedene Einstellungen erfolglos getestet. Unter anderem wurden verschiedene Bandbreiten und entsprechende Corset-Indices getestet, um so ein besseres und stabileres Signal zu erhalten. Auf verschiedenen Bändern haben wir die Bandbreite von 20 auf 15 und 10 mit entsprechenden Corset-Indices (8 und 6) gesetzt. Es ist allerdings mit keiner anderen Kombination ein RRC-Connected zustande gekommen. Ebenso nicht mit den für 5G weiter verbreiteten Bändern 1 oder 78.

Da die Frequenzen von Band 3 nicht in dem für die Antenne vorgesehenen Frequenzbereich liegen, haben wir die Geräte anschließend über ein Kabel und 30 dB Dämpfung verbunden, um Funk-Probleme auszuschließen. Dafür wurde der Transmitting-Output des gNBs mit dem Receiving-Input des UEs verbunden. Analog wurden die anderen beiden Ports verbunden. Da so keine Verbesserung erzielt wurde, konnten Antennen und Funkübertragung ebenfalls als Ursache ausgeschlossen werden.

Basierend auf einem Beispiel mit einem identischen Setup [10, 9] wurde diese Konfiguration mit unserer verglichen und einzelne Änderungen übernommen. Außerdem wurde der ge-

samtel Netzwerkverkehr des Rechners mit `sudo tcpdump -i any -w 5g.pcap` aufgezeichnet, die 5G-Kommunikationspakete extrahiert und analysiert. Eine weitere Verbesserung hat die Änderung nicht gebracht.

Nachdem unser Netzwerk dann allerdings zufälligerweise von dem Handy einer anderen Gruppe gefunden wurde und durch einen Wechsel aller PLMNs, MCCs/MNCs und SIM-Karten Einstellungen auf `001/01` auch verbunden werden konnte, stand fest, dass die Konfiguration passt und die Verbindungsprobleme nicht an der gNB liegen. Das Internet unseres Netzwerks konnte von dem Smartphone ebenfalls genutzt werden. Basierend darauf wurden im Anschluss die Verbindungsprobleme bei den UEs weiter untersucht.

Das Netzwerk war später, als erfolgreiche Verbindungen mit UEs möglich waren, dennoch relativ instabil und Geräte wurden trotz erfolgreicher Verbindung nach einigen Sekunden oder Minuten getrennt. Teilweise wurden auch mehrere Versuche für eine erfolgreiche Verbindung benötigt. Grund für die Trennung des UEs war ein „RRC-Container not ACKed within a time window of 120 msec“, siehe Abbildung 5. Um diesen Timeout zu verhindern, wurde in der gNB Konfiguration der `rrc_procedure_timeout_ms` erhöht. Die Einstellung hatte aber leider keinen Einfluss auf die 120 ms des Acknowledgement-Fensters, sondern hat nur den gesamt Timeout für den RRC Message-Exchange erhöht. Um trotzdem dem schnellen und häufigen Verbindungsabbruch entgegen zu wirken, wurde in der gNB-Konfiguration der `inactivity_timer` des UEs erhöht und der `force_reestablishmentFallback`-Wert auf true gesetzt. Der inactivity-Timer hat die Verbindung etwas verbessert, ein Wiederherstellen der Session hat jedoch nicht stattgefunden.

```
[PHY] [I] [ 120.0] PUCCH: rnti=0x4601 format=1 prb1=105 prb2=na symb=[0, 14) cs=0 occ=1 sr=no t=36.6us
[PHY] [I] [ 120.6] PUCCH: rnti=0x4601 format=2 prb=[2, 3) prb2=na symb=[4, 6) csi1=2222 t=31.7us
[DU-F1] [I] ue=0 du_ue=0 cu_ue=0 proc="UE Context Release": RRC container not ACKed within a time window of 120msec. Proceeding with the UE context release...
[DU-MNG] [I] ue=0 proc="UE Delete": Procedure started...
[DU-MNG] [I] ue=0: SRA and DRB traffic stopped
[DU-F1] [I] ue=0 c-rnti=0x4601 du_ue=0 cu_ue=0: F1 UE context removed.
[PHY] [I] [ 120.7] PUCCH: rnti=0x4601 format=1 prb1=0 prb2=na symb=[0, 14) cs=0 occ=0 ack=2 t=35.9us
[PHY] [I] [ 122.0] PUCCH: rnti=0x4601 format=1 prb1=105 prb2=na symb=[0, 14) cs=0 occ=1 sr=no t=37.1us
[PHY] [I] [ 122.6] PUCCH: rnti=0x4601 format=2 prb=[2, 3) prb2=na symb=[4, 6) csi1=2222 t=36.3us
[MAC] [I] [ 124.4] ue=0 crnti=0x4601 proc="UE Delete Request": finished successfully
[DU-MNG] [I] ue=0 proc="UE Delete": Procedure finished successfully.
[DU-F1] [I] ue=0 proc="UE Delete": Procedure finished successfully.
[DU-F1] [I] Tx PDU du_ue=0 cu_ue=0 du_ue=0: UEContextReleaseComplete
[CU-CP-F1] [I] Rx PDU du_ue=0 cu_ue=0 du_ue=0: UEContextReleaseComplete
[CU-CP] [I] ue=0: "UE Removal Routine" finalized
[NGAP] [I] ue=0 ran_ue=0 amf_ue=132: Sending UeContextReleaseComplete
[NGAP] [I] Tx PDU ran_ue=0 amf_ue=132: UEContextReleaseComplete
```

Abbildung 5: Ausschnitt der AMF-Logs bei Verbindungsabbruch aufgrund von RRC-Timeout.

Eine signifikante Verbesserung der Stabilität konnte durch das Wechseln auf eine andere Frequenz erzielt werden, nachdem es zu Interferenzen mit dem 4G-Netzwerk einer anderen Gruppe gekommen ist, die auf derselben Frequenz gesendet haben. Das Ausschalten von Wireshark-Aufzeichnungen und einzelnen Log-Leveln (wie Physical oder Metrics) hat die Verbindung ebenfalls etwas stabiler gemacht.

3.2 Verteiltes Netzwerk mit Open5Gs und srsRAN (Johann Rittenschober)

Neben dem ursprünglichen Aufbau mit Core Netzwerk und srsRAN auf dem selben Rechner, wie zuvor in Abschnitt 3.1 beschrieben und in Abbildung 4 veranschaulicht, wurde ebenfalls ein verteiltes Netzwerk aufgebaut und getestet. Dieses verteilte Netzwerk wurde in verschiedenen

Varianten realisiert, die in den folgenden Abschnitten beschrieben werden.

3.2.1 Verteilung auf zwei verschiedene PCs (Johann Rittenschober)

Bei der Verteilung auf zwei PCs ist das Core Netzwerk wie zuvor auf dem ursprünglichen Rechner verblieben, das srsRAN bzw. gNB wurde auf einem zweiten Labor-Rechner installiert. Dieser Aufbau ist in Abbildung 6 entsprechend veranschaulicht.

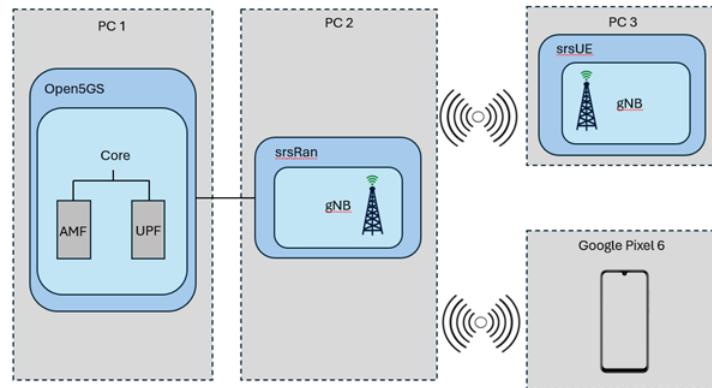


Abbildung 6: Aufbau mit 2 PCs (Bennet Kießling).

Für diese Verteilung musste ebenfalls die Konfiguration angepasst werden, die bisher mit den `localhost` Adressen aus der Default-Konfiguration funktionierte. So muss auf Seite des Core Netzwerks die IP Adresse des NGAP Servers (konfiguriert in `amf.yaml`) und die des GTPU Servers (konfiguriert in `upf.yaml`) auf die Public IP Adresse des PCs gesetzt werden, dass eine Verbindung mit einem externen Rechner aufgebaut werden kann. Auf Seite der gNB muss ebenfalls die eben erwähnte IP Adresse als AMF Adresse gesetzt werden. Die Bind-Adresse ist hierbei die Public IP des srsRAN PCs selbst, da sonst keine (bidirektionale) Verbindung hergestellt werden kann.

Nach den Konfigurationsänderungen und entsprechenden Neustarts der Services konnte die nun externe gNB wie zuvor für die Verbindung verwendet werden.

Da die beiden Rechner per LAN mit dem Netzwerk und somit auch miteinander verbunden sind, war die Annahme, dass die Auslagerung der gNB problemlos möglich sein müsste. Um diese zu prüfen und die Auslagerung als mögliche Fehlerquelle auszuschließen, wurde mit `iperf3` der Durchsatz getestet. Dieser hat mit einer stabilen Verbindung und ca. 950 Mbit/s die Gigabit-Ethernet Verbindung bestätigt.

3.2.2 Verteilung auf drei verschiedene PCs (Johann Rittenschober)

Eine weitere Variante des Netz-Aufbaus war das Core Netzwerk mit zwei verschiedenen gNBs zu betreiben. Dieser Ansatz wurde hauptsächlich für spätere Versuche mit Handover, wie später in Abschnitt 5.1 beschrieben, verfolgt. Zunächst war der Plan eine der gNBs wie zu Beginn auf dem selben PC wie das Core Netzwerk, die zweite (wie im vorherigen Abschnitt 3.2.1 beschrieben)

auf einem externen Rechner zu betreiben. Dabei war jedoch die Konfiguration der IP Adressen ein Problem, da für die externe Node das Public Interface und für die lokale Node `localhost` benötigt wird. Während die eigene Public IP als AMF Adresse vermutlich funktioniert hätte, konnte diese nicht ebenfalls als Bind-Adresse verwendet werden, da der UDP Port bereits vom Core in Verwendung war. Bei einem Bind stattdessen auf `localhost` war kein Verbindungsauftbau möglich.

Daher wurde die zweite gNB auf einen dritten Labor-Rechner ausgelagert, wie in Abbildung 7 veranschaulicht. Dabei gilt es zusätzlich zu beachten, dass dafür – im Gegensatz zu einem UE (siehe 3.3) – kein eigener Laptop verwendet werden kann, da die Labor-Rechner nicht über das VLAN-13 (verbunden über eduVPN) erreichbar sind.

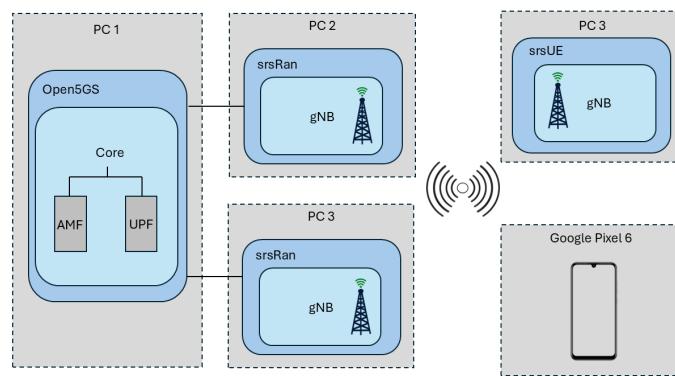


Abbildung 7: Aufbau mit 3 PCs (Bennet Kießling).

3.3 User Equipments

Nach dem Einrichten eines 5G-Netzes mit open5GS und srsRAN wurden mehrere User Equipment (UE) Alternativen für das Testen der Verbindung zum Netz verwendet. Diese werden in diesem Kapitel näher beschrieben.

3.3.1 Theorie (Marina Trinz)

Jeder Teilnehmer von mobilen Netzen muss über eine eindeutige Identifikationsnummer verfügen. Diese Nummer wird als International Mobile Subscriber Identity (IMSI) in 4G-Netzen oder als Subscription Permanent Identifier (SUPI) in 5G-Netzen bezeichnet [43, p. 179]. Die IMSI/SUPI-Nummer wird einem UE über eine SIM-Karte eines Mobilfunknetzbetreibers zugewiesen, auf der die Nummer auch gespeichert ist. Die IMSI/SUPI-Nummer wird aus drei weiteren Nummern zusammengestellt, darunter sind [43, p. 14] (vgl. Abbildung 8):

- Mobile Country Code (MCC) - eine Nummer, die angibt, aus welchem Land ein Netzteilnehmer kommt. Jedem Land wurde eine eindeutige dreistellige Nummer zugewiesen. Zum Beispiel verwenden die Mobilfunknetzbetreiber aus Deutschland die Nummer 262;

- Mobile Network Code (MNC) - mit dieser Nummer werden unterschiedliche Mobilfunknetzbetreiber aus einem Land voneinander unterschieden. Zum Beispiel haben die Mobilfunknetzbetreiber Vodafone und Telefonica aus Deutschland die MNC-Nummern 02 beziehungsweise 03;
- Mobile Subscriber Identification Number (MSIN) - eine national eindeutige Nummer eines Netzteilnehmers.

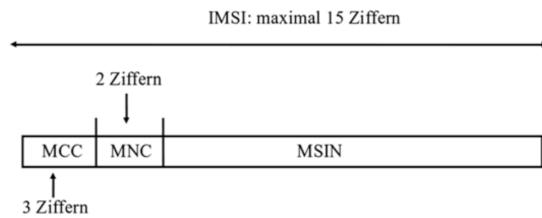


Abbildung 8: Zusammensetzung einer IMSI/SUPI-Nummer (Quelle [43, p. 15]).

Somit ist jede IMSI/SUPI-Nummer durch die Kombination dieser drei Nummern international eindeutig. Diese Eigenschaft ermöglicht auch das Verbinden eines UE im Roaming zu ausländischen Netzen.

Die IMSI/SUPI-Nummer wird für die Verwaltung von Netzteilnehmereinstellungen wie Tarifinformationen, maximale Datenrate und Zugang zu bestimmten Diensten durch die Mobilfunknetzbetreiber benötigt. Für die Herstellung einer Verbindung zum Heimatnetz wird die IMSI/SUPI-Nummer vom UE an das Netz verschickt. Damit ein UE sich mit einem Netz verbinden kann, muss die IMSI/SUPI-Nummer dieses UE in der Datenbank des Netzes hinterlegt werden. Für die Verwaltung von Netzteilnehmerdaten, die im Unified Data Repository (UDR) gespeichert sind, ist in 5G-Netzen das Unified Data Management (UDM) verantwortlich [43, p. 178].

Weitere Werte, die auf der SIM-Karte gespeichert sind, sind Operator Code / Derived Operator Code (OP/OPc) und Authentication Key (K). Sie werden für die Authentifizierung eines Netzteilnehmers und die Sicherheit der Kommunikation zwischen UE und dem Netz eingesetzt [44].

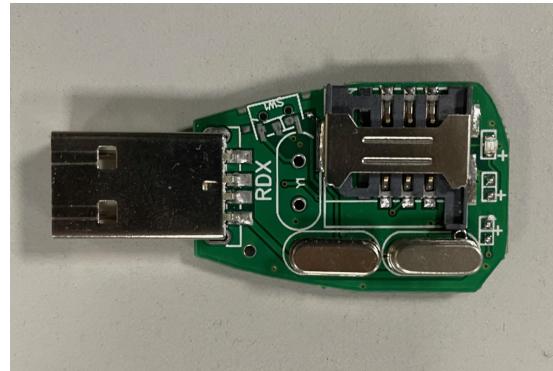
3.3.2 Open cells SIM-Karte (Marina Trinz)

Um die Verbindung zum eingerichteten Netz mit einem Smartphone herstellen zu können, wurde eine SIM-Karte von Open cells konfiguriert (vgl. Abbildung 9a). Für die Konfiguration wurde ebenfalls ein Kartenlesegerät von Open cells eingesetzt (vgl. Abbildung 9b).

Für die Konfiguration der SIM-Karte wurde der Anleitung von Open cells [23] gefolgt. Nach dem Herunterladen wurde das Plugin UICC v3.3 mit dem Befehl `make` kompiliert. Daraufhin



(a) Eine SIM-Karte von Open cells.



(b) Ein Kartenlesegerät von Open cells.

Abbildung 9: Von Open cells verwendete Hardware.

konnten die auf der SIM-Karte gespeicherten Daten mit dem Befehl `sudo ./program_uicc` ausgelesen werden. Die Ausgabe in der Kommandozeile kann Abbildung 10 entnommen werden.

```
matthias@matthias-Precision-5820-Tower-X-Series:~/Downloads/uicc-v3.3$ sudo ./program_uicc
Existing values in USIM
ICCID: 89860061100000000645
WARNING: iccid luhn encoding of last digit not done
USIM IMSI: 001020100001645
PLMN selector: : 0x00f1207c
Operator Control PLMN selector: : 0x00f1207c
Home PLMN selector: : 0x00f1207c
USIM MSISDN: 00000645
USIM Service Provider Name: internet

No ADM code of 8 figures, can't program the UICC
```

Abbildung 10: Auslesen der Daten einer Open cells SIM-Karte.

Da während des Projekts mehrere Experimente mit unterschiedlichen MCC und MNC durchgeführt wurden, musste die SIM-Karte entsprechend immer wieder neu konfiguriert werden, damit die IMSI/SUPI-Nummer den neuen Einstellungen des Kernnetzes entspricht. Das Über schreiben der SIM-Karte mit neuen Werten im letzten Experiment erfolgte mit dem Befehl

```
sudo ./program_uicc -adm OC012645 -imsi 001010100001645 -key 00112233445566778899aabccddeeff
-opc 63bfa50ee65233 65ff14c1f45f88737d -spn „internet“ (vgl. Abbildung 11).
```

Abschließend werden diese Daten der SIM-Karte über das WebUI `http://localhost:9999` im Kernnetz gespeichert, damit die SIM-Karte als Nutzer des Netzes registriert wird (s. Abbildung 12).

```
matthias@matthias-Precision-5820-Tower-X-Series:~/Downloads/utcc-v3.3$ sudo ./program_uicc --adm 0C012645 --imsi 001010100001645 --key 00112233445566778899aabccddeeff --opc 63bfa50ee6523365ff14c1f45f88737d -spn "internet"

Existing values in USIM
ICCID: 898600611000000000645
WARNING: iccid luhn encoding of last digit not done
USIM IMSI: 001020100001645
PLMN selector: : 0x00f1207c
Operator Control PLMN selector: : 0x00f1207c
Home PLMN selector: : 0x00f1207c
USIM MSISDN: 00000645
USIM Service Provider Name: internet

Setting new values

Reading UICC values after uploading new values
ICCID: 898600611000000000645
WARNING: iccid luhn encoding of last digit not done
USIM IMSI: 001010100001645
PLMN selector: : 0x00f1107c
Operator Control PLMN selector: : 0x00f1107c
Home PLMN selector: : 0x00f1107c
USIM MSISDN: 00000645
USIM Service Provider Name: internet
matthias@matthias-Precision-5820-Tower-X-Series:~/Downloads/utcc-v3.3$
```

Abbildung 11: Ausgabe der Kommandozeile nach der erfolgreichen Konfiguration einer Open cells SIM-Karte.

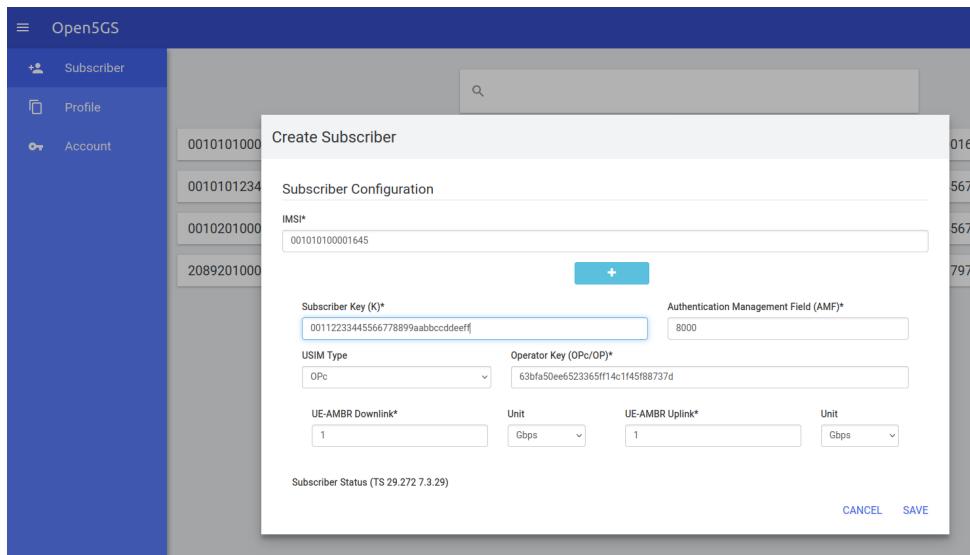


Abbildung 12: WebUI von Open5gs für die Registrierung von SIM-Karten.

3.3.3 Quectel UE (Chiara Piccolroaz)

Die erste Equipment-Alternative, die wir für das Testen der Verbindung zu unserem 5G-Netzwerk verwendet haben, war das Quectel RM500Q-AE. Dies ist ein User Equipment in Form eines Funkmoduls, das speziell für schnelle drahtlose Verbindungen in IoT- und eMBB-Anwendungen entwickelt wurde. Neben der einfachen Verfügbarkeit und Benutzbarkeit stellt die Unterstützung der von uns im Netzwerk verwendeten Generation, Frequenzen und Bänder [21] eine zentrale Entscheidung für den Einsatz dieses Geräts in unserem Projekt dar (Tabelle 2).

Tabelle 2: Generation und unterstützte Bänder.

Generation	5G NR SA/ 5G NR NSA/ LTE
5G NR SA Band	1/2/3/5/7/8/12/20/25/28/38/40/41/48/66/71/77/78/79

Für eine erfolgreiche Konfiguration muss das Quectel-UE über USB mit dem Systemrechner verbunden und mit einer in der Software OpenCell hinterlegten SIM-Karte ausgestattet sein

(Siehe Abschnitt 3.3.1). In unserem Projekt haben wir die USB2-Schnittstelle und eine SIM-Karte der dritten Generation verwendet. Es können jedoch auch andere Karten und Schnittstellen, wie z.B. USB3 für eine höhere Übertragungsgeschwindigkeit, verwendet werden [37].

Vor und während der Inbetriebnahme ist es empfehlenswert, das Gerät und seine Verbindung über AT-Befehle mithilfe von Terminalprogrammen wie Minicom [38] weiter zu konfigurieren und zu testen. AT-Befehle (Attention-Befehle) sind eine standardisierte Befehlssprache, die ursprünglich zur Steuerung von Modems entwickelt wurde. Minicom ist ein terminalbasiertes Kommunikationsprogramm für Unix-Systeme zur Kommunikation mit seriellen Geräten, wie Routern, Modems, Mikrocontrollern oder anderen Hardware-Komponenten. Die Installation kann problemlos über Paketmanager erfolgen, und die Verbindung zum seriellen Gerät kann durch den Befehl `sudo minicom -D /dev/ttyUSB<PortNr>` für den richtigen seriellen Port gestartet werden. Danach können mit dem Textterminal-Emulator die AT-Befehle eingegeben werden. Dabei ist es empfehlenswert, die Befehle `AT` oder `ATI` auszuführen, um die erfolgreiche Hardware- und Softwareverbindung zu überprüfen (Tabelle 3).

Tabelle 3: Liste der wichtigsten AT-Befehle.

AT	Grundlegender Testbefehl: überprüft, ob das Modem bereit ist
ATI	Grundlegender Testbefehl: zeigt Informationen über das Modem an
AT+CFUN=1,1	Setzt das Modem zurück und startet es neu
AT+QGMR	Zeigt die aktuelle Firmware-Version des Modems an
AT+CPIN?	Überprüft den Status der SIM-Karte
AT+COPS=?	Überprüft die verfügbare Netzbetreiber
AT+CGATT?	Überprüft, ob das Modem am Netzwerk angemeldet ist
AT+QNWINFO	Zeigt detaillierte Netzwerkverbindungsinformationen an
AT+CSQ	Zeigt die Signalstärke an
AT+QENG=servingcell	Zeigt die Details der derzeitigen verbundenen Zelle an

Nach diesen Konfigurations- und Testschritten erwarteten wir eine automatische Verbindung zwischen dem Quectel-UE und unserem 5G-Netzwerk, die jedoch nicht zustande kam. Um dieses Problem zu lösen, versuchten wir, alle Voraussetzungen für eine erfolgreiche Verbindung schrittweise und systematisch zu überprüfen.

In unserem ersten Test führten wir einige Befehle aus, um den Status des Geräts zu überprüfen, der korrekt bestätigt wurde. Dazu verwendeten wir z.B. die Befehle `AT+QGMR`, um den Status der Firmware, und `AT` sowie `AT+CPIN?`, um die Funktionalität des Moduls und der SIM-Karte zu überprüfen (Tabelle 3). Aus Sicherheitsgründen führten wir den Befehl `AT+CFUN=1,1` aus, um die durch vergangene Tests veränderten Geräteinstellungen zurückzusetzen.

In unserem zweiten Test überprüften wir die Empfangs- und Verbindungsfähigkeit des Geräts mit dem Netz, was interessantere Informationen ergab. Zuerst aktivierten wir unser Gerät im Flur, das problemlos öffentliche Netze empfangen konnte (Abb. 13). Da das Gerät jedoch keinen Internetzugang hatte, tauschten wir die SIM-Karte gegen eine aus unserem privaten Mobiltele-

fon aus und konnten so eine erfolgreiche und stabile Internetverbindung herstellen (Abb. 14). Die empfangenen Netze waren jedoch ausschließlich 4G-Netze, sodass wir die beiden anderen Teams baten, ihre beiden 4G-Netze zu aktivieren. Entgegen unseren Erwartungen wurde nur das 4G-Netz einer der beiden Gruppen gefunden (Abb. 15). Aus unserer Sicht bestand der einzige relevante Unterschied zwischen den beiden Netzen in den verwendeten 4G-Bändern (n7 und n78), die aber laut offizieller Spezifikation [39] beide gut unterstützt werden sollten. Da wir weder die Zeit noch die technischen Möglichkeiten hatten, die Netzwerkeigenschaften unseres 5G-Netzes flexibel zu verändern, konnten wir keine schlüssige Erklärung und Lösung für dieses Ergebnis finden.

```
AT+COPS=?  
ATI  
  
+COPS: (3,"Telekom.de","TDG",26201",7),(1,"o2 - de","o2 - de",26203",7),(1,"vodafone.de","Vodafone",26202",7),,(0-4),(0i
```

Abbildung 13: Informationen über die im Flur erreichbaren Mobilfunkanbieter.

```
AT+QENG="SERVINGCELL"  
  
+QENG: "servingcell","LIMSRV","LTE","FDD",262,03,186B817,267,6200,20,3,3,C936,-109,-20,-70,5,0,-,16  
  
OK  
AT+QENG="NEIGHBOURCELL"  
  
OK  
AT+CSQ  
  
+CSQ: 21,99
```

Abbildung 14: Informationen über die verbundene Netzwerkqualität und Zellen.

```
AT+COPS=?  
  
+COPS: (1,"Test PLMN 1-1","Test1-1","00101",7),,(0-4),(0-2)
```

Abbildung 15: Informationen über die im Labor erreichbaren Mobilfunkanbieter.

Als letzten Test versuchten wir, die Konfiguration des Quectel RM500Q-UA mit anderen AT-Befehlen genau an unsere Bedürfnisse anzupassen, jedoch ohne Erfolg. Dabei konzentrierten wir uns auf fehlende oder falsch eingestellte Parameter für APN, Bänder und Netzwerkregistrierung, die in verschiedenen Online-Quellen und Foren erwähnt wurden. Am Ende der ersten Woche beschlossen wir, das Modul nicht weiter zu verfolgen, da das Netzwerk grundsätzlich funktionierte und zumindest zwei Mobiltelefone und zwei srsUEs unser Netzwerk empfangen konnten.

3.3.4 srsUE - Installation (Johann Rittenschober)

Nach den Verbindungsproblemen mit dem Quectel-UE wurde ein srsUE als Alternative aufgesetzt. Das srsUE ist ein komplett in Software implementiertes UE-Modem, das sich über eine Software-Defined-Radio (SDR) Hardware (hier ein zweites B210) zu dem 5G Netzwerk verbindet [34]. Das srsUE ist ein Teil von srsRAN, allerdings nicht von der für das RAN verwendeten 5G

Variante (srsRAN Project), sondern von srsRAN 4G. Dennoch stellt das srsUE dort auch seit den neueren Versionen 5G Funktionalitäten bereit [35].

Die Installation des srsUE wurde für bessere Flexibilität und verschiedene Tests (siehe zum Beispiel 4.4) auf mehreren Geräten vorgenommen. Darunter ein MSI Laptop (AMD Ryzen 9, Ubuntu 24.10 Beta), ein MacBookPro (M1 Max, Linux Fedora 40) und ein Dell Workstation PC (Intel i9, Ubuntu 22.04.5). Auf den verschiedenen Geräten hat bereits die Installation einige (und je Gerät teilweise unterschiedliche) Probleme bereitet.

Im Laufe der Installationen hat sich herausgestellt, dass die in der Dokumentation beschriebene einfache Installations-Variante via package-manager auch auf Ubuntu (dem einzigen dafür unterstützten System) nicht funktioniert. Auf Ubuntu 22 ist die Version des `srsue` Befehls veraltet und unterstützt keine 5G Funktionalität und auf Ubuntu 24 wird gar kein Package bereitgestellt. Somit muss für jede Installation, nicht nur die für andere Systeme wie Fedora, der Source Code selbst gebaut werden.

Bei dem Build-Prozess kam es je nach System und Versionen der Dependencies zu verschiedenen Fehlern, so musste beispielsweise auf einem der Systeme aufgrund einer neueren `cmake` Version die Konfiguration abgeändert werden um ein benötigtes Modul zu finden.

Was auf mehreren Systemen auftrat, waren als Error behandelte Warnings. Dieses Problem konnte nach mehreren Versuchen umgangen werden, indem `cmake` angewiesen wird es bei den Warnings zu belassen und keine Errors daraus zu machen. Diese Warnings könnten natürlich trotzdem problematisch sein, es ließ sich aber keine direkte Beeinträchtigung oder Abstürze dadurch feststellen. Auch die mit `make test` ausgeführten Checks waren auf allen Systemen erfolgreich.

Die genauen bei den verschiedenen Installationen getätigten Installationsschritte sowie die zugehörigen Probleme wurden detailliert im Anhang A dokumentiert. Diese Dokumentation enthält ebenfalls weitere aufgetretene Fehler, deren Behebung/Umgehung, Test-System Spezifikationen, Versionsnummern und weitere Hinweise. So zum Beispiel auch, dass Wireshark erst entsprechend konfiguriert werden muss um die vom srsUE gespeicherten Aufzeichnungen korrekt dekodieren zu können. Dies wird auch in Abschnitt 4.2 beschrieben.

3.3.5 srsUE - Konfiguration (Sarah König)

Als initiale Konfiguration für das srsUE wurde die für ein 5G FDD-UE-Device [33] übernommen. Damit der Verbindungsaufbau erfolgreich ist, ist es wichtig, dass das B210 über USB 3.0 angegeschlossen ist. Wenn das Gerät lange angeschlossen war, wurde teilweise auf USB 2.0 gewechselt. In diesem Fall muss für USB 3.0 das USB-Kabel neu angeschlossen werden.

Bei der Konfiguration und den einzelnen in Abschnitt 3.1.3 beschriebenen Änderungen war es wichtig immer `srate`, `band`, `max_nof_prb` und `nof_prb` (entsprechend der Bandbreite [32]) und APN-Daten an die Werte der gNB anzupassen. Außerdem konnten in der srsUE-Konfiguration

IMSI, IMEI, OPC und Key Werte für die SIM-Karte gesetzt und verschiedene Log-Level und Pcap-Dateien aktiviert werden. Bei den Pcap-Dateien darf die Variabel `enable` nicht wie in mehreren Beispiel-Konfigurationen auf `enable / true` gesetzt werden, da so keine Aufzeichnungen erstellt werden. Stattdessen müssen die Elemente aufgelistet werden, die aufgezeichnet werden sollen (z.B. `mac_nr`, `nas`). Die Carrier bei Eutra müssen `0` sein, um sich nicht zu einem 4G Netz zu verbinden. Der Device-Name ist entsprechend dem Verbindungsprotokoll `uhd`. Für eine bessere Synchronisation wurde ein GPSDO Signal als Referenz für die Clock angeschlossen. In der Konfiguration wurde dafür `device_args= clock=gpsdo, sync=gpsdo` gesetzt. Ohne diese Synchronisation kommt kein RRC-Connected zustande.

Nach der Konfiguration und Starten des UEs konnte nach einigen in Abschnitt 3.1.3 beschriebenen Modifikationen eine RRC-Verbindung aufgebaut werden. Die PDU-Session, der nächste Schritt in dem Verbindungsaufbau, kam hier nicht zustande. Als Grundlage für die im Folgenden beschriebenen Änderungen wurden verschiedene Bug-Fix-Reports, Beispiel-Konfigurationen, Log-Ausgaben und Wireshark-Aufzeichnungen (über die pcap-Dateien oder tcpdump) verwendet. Aufgrund der Fehlermeldungen, Timeouts und den teilweise auftretenden Over- und Underflows wurden in Kombination mit den Anpassungen in der gNB auch die Konfigurationen des srsUE geändert. Unter anderem wurden verschiedene Bänder, auf welchen trotz Bandbreiten-, Frequenz- und Antennen-Änderungen kein RRC-Connect zustande kam, und die Kabelverbindung getestet. Außerdem wurden verschiedene Kombinationen von RX- und TX-Gain getestet und der Device Name auf Auto gesetzt. Die Änderungen haben das Ergebnis entweder verschlechtert oder nicht beeinflusst.

Auch Performance Anpassungen im BIOS oder andere Konfigurationen haben auf dem initial verwendeten Rechner keine PDU Verbindung erzeugen können. Nachdem letztendlich auf einem anderen Rechner mit gleicher srsUE-Konfiguration eine erfolgreiche Verbindung mit PDU-Session und IP-Adresse möglich war, lag das Verbindungsproblem an dem Rechner und nicht an einer falschen Konfiguration. Mögliche Ursachen könnten dabei an der Installation des srsUEs, der Ubuntu-Version, die Kombination von verschiedenen Versionen, zu wenig Rechenressourcen oder weiterem liegen.

3.3.6 Handy (Chiara Piccolroaz)

Aufgrund der fehlgeschlagenen Verbindungsversuche mit der Quectel UE und der instabilen Verbindung mit den srsUEs wurde versucht, eine dritte alternative UE zu nutzen.

Dabei haben wir uns auf Mobiltelefone konzentriert, die während der zweiwöchigen Projektlaufzeit als primäres Nutzerequipment für die Experimente und Tests unseres 5G-Netzes dienten, die in den folgenden Abschnitten beschrieben werden. Der Grund dafür ist die insgesamt stabilere und stärkere Netzwerkverbindung im Vergleich zu der sehr instabilen und unvorhersehbaren Verbindung der srsUEs (Abschnitt 3.3.4) und dem fehlenden Netzempfang der Quectel-UE (Ab-

schnitt 3.3.3). Ein weiterer Vorteil ist die einfache und schnelle Nutzung der Geräte, da lediglich eine in der OpenCell-Software registrierte SIM-Karte eingelegt werden muss.

Ein Nachteil ist jedoch, dass spezielle Software installiert werden müssen, um mit den Geräten zu kommunizieren oder genauere Informationen zu erhalten wie z. B. PingTools oder Google Speedtest. Darüber hinaus konnten nur zwei der im Labor zur Verfügung gestellten und der persönlichen Mobiltelefone unser 5G-Netz empfangen und auf das Internet zugreifen. Die verwendeten Modelle waren das OnePlus Nord 2T und das Pixel 6A, wobei das Pixel 6A für die meisten Tests verwendet wurde. Beide Geräte wurden 2022 auf den Markt gebracht, unterstützen sowohl 4G als auch 5G (SA und NSA) und eine Vielzahl von Bändern und Frequenzen. Um einen genaueren Überblick über ihre Eigenschaften zu erhalten, hat sich die Website *GSMarena* als sehr hilfreich erwiesen. Diese bietet eine umfangreiche Datenbank mit Mobilfunkspezifikationen, wie in den beiden folgenden Abbildungen (16, 17) dargestellt.

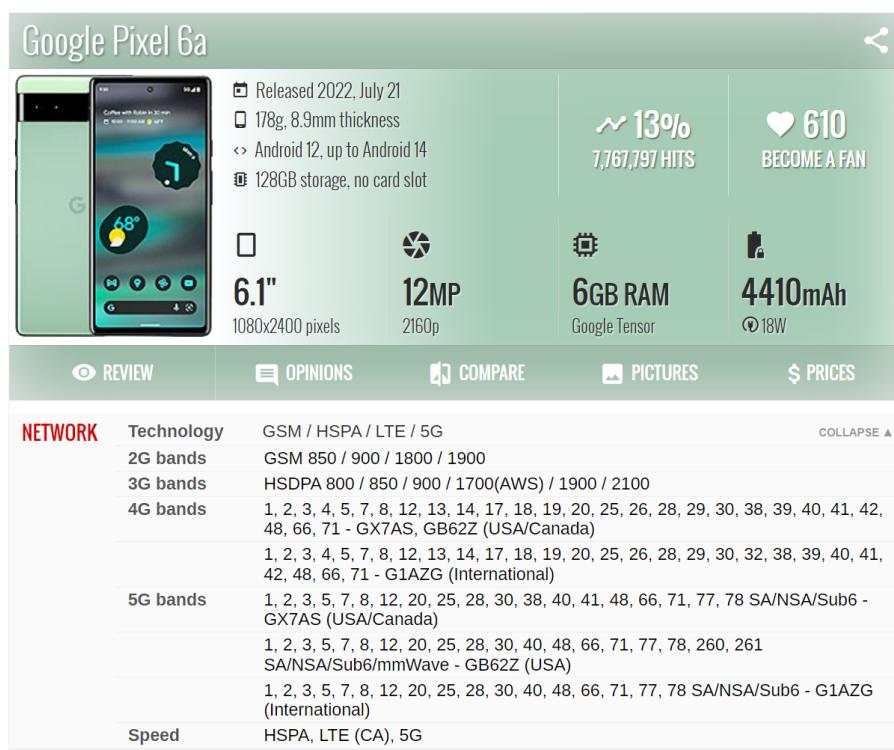


Abbildung 16: Informationen über Pixel 6A.

Plausible Gründe dafür, dass Mobiltelefone stabilere und stärkere Verbindungen bieten können, liegen wahrscheinlich in ihrer Robustheit, Flexibilität und Benutzerfreundlichkeit. Mobiltelefone sind in der Regel sofort einsatzbereit. Im Vergleich zu srsUEs müssen die Benutzer keine speziellen Konfigurationen oder Einstellungen vornehmen, was das Fehlerrisiko verringert. Die meisten Mobiltelefone können sich automatisch an verschiedene Netze und Frequenzen anpassen sowie Störungen und Interferenzen, was den Einsatz in unterschiedlichen Umgebungen erleichtert. Außerdem verfügen Mobiltelefone bereits über integrierte Einstellungen und Protokolle, die für den Zugang zu Mobilfunknetzen optimiert sind. Das bedeutet, dass die wichtigsten Para-

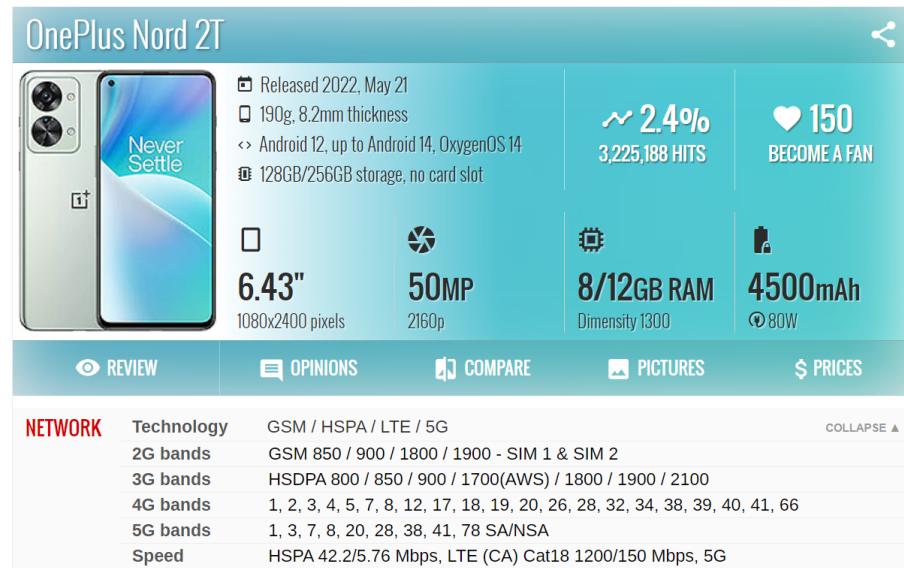


Abbildung 17: Informationen über OnePlus Nord 2T.

meter für den Netzzugang wie APN und Netzauswahl, die wir in unserem Quectel UE manuell anzupassen versuchten, oft bereits richtig konfiguriert sind [27, 26].

4 Analyse der durchgeführten Versuche

Nachdem alle Komponenten des 5G-Netzes erfolgreich installiert worden waren, wurden mehrere Messungen durchgeführt, um die Qualität und Stabilität der Verbindung zu testen. Diese Messungen werden in diesem Kapitel näher beschrieben und analysiert.

4.1 Spektrum-Analyse (Johann Rittenschober)

Einer der ersten Versuche war die Messung der Funk-Frequenzen mittels Spektrum-Analyse. Dies diente hauptsächlich der Fehlersuche und um zu verifizieren, dass die gNB tatsächlich ein Funksignal sendet. Zu diesem Zeitpunkt konnte das Netz noch von keinem der anderen Testgeräte wahrgenommen werden.

Für die Messung der Frequenzen kann ebenfalls ein Radio-Interface wie für die gNB verwendet werden, in unserem Fall wieder ein USRP B210. Als Software wird hierfür das Open-Source Projekt GNU Radio [40] verwendet. Damit lässt sich mittels Flow-Graph ein software-definiertes Funkgerät mit entsprechender Benutzeroberfläche zusammenbauen (vgl. Abbildung 18), die dann ausgeführt werden und die empfangenen Frequenzen graphisch darstellen kann.

Somit konnten wir feststellen, dass unsere gNB in der Tat ein Signal sendet. In Abbildung 19 ist dieses bei 1842,5 MHz zu sehen, was mit der eingestellten ARFCN von 368500 für Band n3 korrespondiert. Die im Wasserfall (Abbildung 19b) sichtbaren Linien sind vermutlich eine Form von Signalstörung, da es bei beliebigen Frequenzen über das gesamte Frequenz-Spektrum hinweg

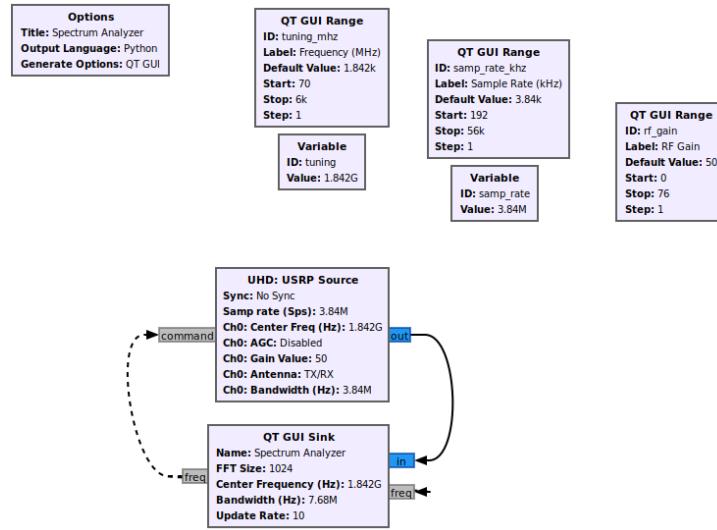


Abbildung 18: Der Flow-Graph für den Spektrum-Analysator in GNU Radio.

periodisch auftritt.

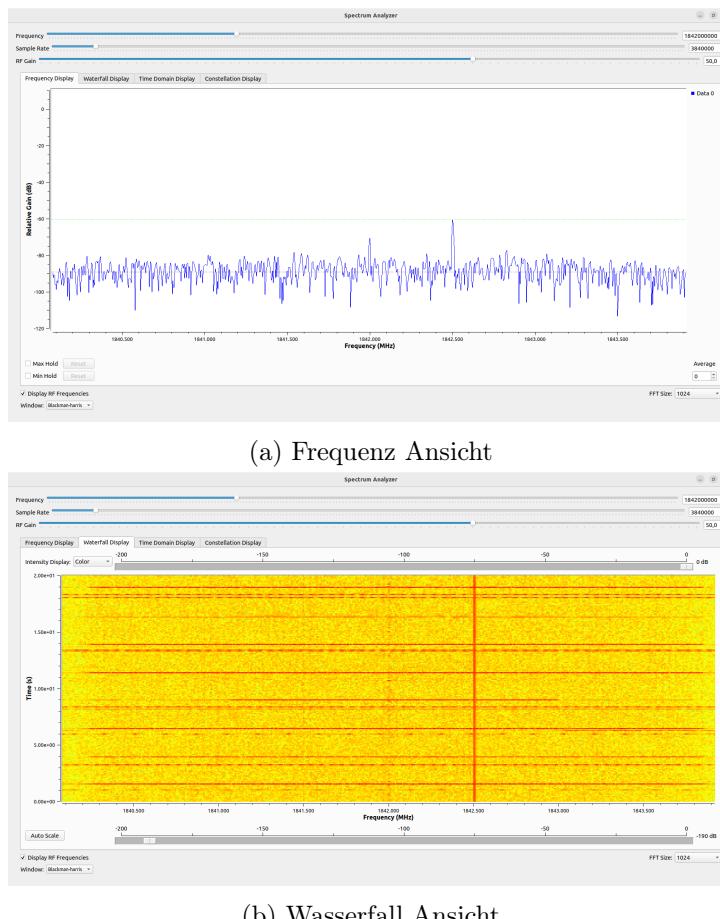


Abbildung 19: Frequenz der gNB sichtbar in GNU Radio.

Die Schritte um einen Spektrum-Analysator mit GNU Radio zu erstellen sind ebenfalls detaillierter in Anhang B dokumentiert.

4.2 Wireshark (Chiara Piccolroaz)

Ein zentraler Ansatz zur Analyse des Verbindungsverkehrs war die Nutzung von PCAP-Dateien (Packet Captures) und deren Untersuchung mit Wireshark. PCAP-Dateien erfassen den Netzwerkverkehr und enthalten detaillierte Informationen über die gesendeten und empfangenen Datenpakete, darunter Protokolldaten, Quell- und Zieladressen sowie Ports [41]. Wireshark ist ein leistungsstarkes und quelloffenes Tool, das speziell für die Analyse und Überwachung von Netzwerkpaketen entwickelt wurde. Es ist besonders nützlich für Netzwerkadministratoren und Entwickler, kann jedoch auch für die Sicherheitsüberwachung und die Analyse von Cyberangriffen eingesetzt werden.

In unserem Projekt stellt Wireshark eine gute Alternative dar, da es moderne Protokolle wie NGAP, GTP und NAS unterstützt und auf die spezifischen Herausforderungen von 5G-Netzen eingeht [48], wie z.B. die höhere Datenraten und geringere Latenzen. Wireshark wird in unserem Fall hauptsächlich dazu verwendet, PCAP-Dateien einzulesen, um Verbindungsprobleme und Paketverluste zwischen unserem gNB und den verwendeten UE-Typen aufzuspüren [47].

Die Installation von Wireshark auf Linux ist über den apt-Paketmanager sehr einfach durchführbar. Die Nutzung, insbesondere das Auslesen von PCAP-Logs von erfordert jedoch einige Schritte. Erstens müssen die richtigen Netzwerkschnittstellen ausgewählt werden, sodass Wireshark in einem Netzwerk eingesetzt wird, in dem die relevanten Geräte vorhanden sind. Zweitens müssen die PCAP-Dateien den gesamten Netzwerkverkehr des Rechners aufzeichnen, was mithilfe des Befehls `sudo tcpdump -i any -w 5g.pcap` geschieht, wie in Abschnitt 3.1 beschrieben. Drittens müssen einige Einstellungen in Wireshark vorgenommen werden, um die Dateien korrekt anzuzeigen. Nach der Generierung der PCAP-Dateien, die im kompakten MAC-LTE- und MAC-NR-Format im Ordner `/tmp` gespeichert werden, ist es hilfreich, die DLT_USER-Dissektoren gemäß den Anweisungen auf der offiziellen srsRAN-Webseite [36] anzupassen. Kurz zusammengefasst setzt man für MAC-LTE und MAC-NR DLT=149, für NAS DLT=148 und für S1AP zwischen MME und eNodeBs DLT=150. Da die Menge der angezeigten Daten in Wireshark sehr umfangreich sein kann, ist es sinnvoll, Filter zu setzen, um gezielt bestimmte Datenpakete zu analysieren. Zum Beispiel kann der Filter „ip“ verwendet werden, um nur den Internetverkehr auf Basis von IP-Adressen zu sehen, „ngap“ für das Next Generation Application Protocol oder „gtp“ für das GPRS Tunneling Protocol, das in 5G- und LTE-Netzwerken häufig genutzt wird. In unserem Projekt lag der Fokus auf NGAP und GTP.

Die Auswertung der PCAP-Dateien mit Wireshark lieferte zahlreiche wertvolle Informationen für unseren Netzaufbau. Wie in Abschnitt 3.1 beschrieben, gehörten zu den identifizierten Problemen Authentifizierungsfehler durch falsch konfigurierte SIM-Karten, Herausforderungen bei der PDCCH- und PRACH-Konfiguration, die den erfolgreichen Aufbau einer RRC-Verbindung verhinderten, sowie fehlende NGAP-Kontexte und Acknowledgment-Nachrichten für das UE.

4.3 File-Sharing Service (Sarah König)

Nachdem wir in der Lage waren mehrere UEs (sowohl ein physikalisches als auch mehrere srsUEs) mit unserem 5G Netzwerk zu verbinden, konnten wir diese Verbindung testen. Zuerst wurde die Konnektivität zwischen der gNB und den einzelnen UEs über einen `ping` in beide Richtungen sichergestellt. Danach wurde ebenfalls erfolgreich getestet, dass zwei mit dem Netzwerk verbundene UEs sich gegenseitig pingen können. Der nächste logische Test-Schritt war, dass die UEs sich nicht nur gegenseitig pingen können, sondern dass auch andere kleine Applikationen auf IP-Ebene zwischen ihnen ausgeführt werden können.

Eine Applikation, die basierend auf IP-Adressen zwischen zwei über das 5G-Netzwerk verbundenen UEs implementiert wurde, ist ein File-Sharing-Server mit dem Python Modul `http.server` [14]. Der HTTP-Server ermöglicht das Teilen von statischem Inhalt, wie Dateien und Ordner. Standardmäßig exportiert der Server den Inhalt des Working-Directories, in dem der Befehl gestartet wurde [14, 49].

Für den Versuchsaufbau wurden ein srsUE sowie das Google Pixel 6a verwendet, welche über die gNB mit dem 5G-Netzwerk verbunden waren. Auf dem Rechner des verbundenen srsUE wurde über den Befehl `python3 -m http.server 8000` der File-Sharing-Server gestartet (siehe Abbildung 20). Der Befehl startet den HTTP-Server auf Port 8000 und bindet alle verfügbaren Netzwerk-Interfaces des Rechners, die eine IP-Adresse besitzen [49]. Also über das srsUE auch das 5G-Netzwerk. So kann jeder, der sich im gleichen Netzwerk befindet, über die IP-Adresse auf den soeben gestarteten Server zugreifen.

Navigiert man im Browser des zweiten UEs zu der IP-Adresse des srsUEs, kann auf die freigegebenen Dateien und Ordner des Rechners zugegriffen werden (siehe 21). In den Ordner kann frei navigiert und die Dateien bzw. Bilder heruntergeladen werden. Der Zugriff auf die freigegebenen Inhalte war über alle mit dem 5G-Netzwerk verbundenen Komponenten möglich, also sowohl über den Rechner mit der gNB als auch über weitere Rechner, die über ein srsUE mit dem 5G-Netzwerk verbunden waren.

```
chiara@chiaraAlpha:~/config/srsran$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.45.0.59 - - [23/Sep/2024 15:45:03] "GET / HTTP/1.1" 200 -
10.45.0.59 - - [23/Sep/2024 15:45:03] code 404, message File not found
10.45.0.59 - - [23/Sep/2024 15:45:03] "GET /favicon.ico HTTP/1.1" 404 -
10.45.0.59 - - [23/Sep/2024 15:45:32] "GET /enb.conf HTTP/1.1" 200 -
```

Abbildung 20: Terminal-Output des http.servers bei dem srsUE.

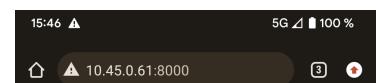


Abbildung 21: Browser-Ausschnitt File-Sharing.

- [enb.conf](#)
- [epc.conf](#)
- [mbms.conf](#)
- [rb.conf](#)
- [rr.conf](#)
- [sib.conf](#)
- [ue.conf](#)
- [user_db.csv](#)

4.4 NetCat (Chiara Piccolroaz)

Netcat ist ein vielseitiges Netzwerk-Tool, das die Kommunikation zwischen zwei Geräten über TCP oder UDP ermöglicht. TCP (Transmission Control Protocol) und UDP (User Datagram Protocol) sind zwei der wichtigsten Protokolle aus der Familie der Internetprotokolle. In unserem Projekt wurde Netcat hauptsächlich für den Nachrichtenaustausch über TCP innerhalb unseres 5G Netzwerks verwendet. Netcat kann jedoch auch für andere Funktionen wie das Lesen und Schreiben von Daten, das Öffnen von Ports und das Scannen von Netzwerken verwendet werden [46].

Die grundlegenden Voraussetzungen für den Einsatz von Netcat sind die Installation der Software sowie die Erreichbarkeit der beteiligten Geräte. Auf den meisten Linux-Distributionen ist Netcat bereits vorinstalliert oder lässt sich bei Bedarf schnell durch `sudo apt install netcat` nachinstallieren. Aufgrund möglicher Kompatibilitätsprobleme ist es jedoch empfehlenswert, die neueste Version auf allen beteiligten Rechnern zu installieren. Die Erreichbarkeit der Geräte über IP-Adressen muss gewährleistet sein, was sich beispielsweise durch einen einfachen Ping-Test überprüfen lässt. Befinden sich beide Geräte im selben Netzwerk, gestaltet sich der Datenaustausch besonders unkompliziert, da sie direkt über private IP-Adressen miteinander kommunizieren können. Sollten sich die Geräte jedoch in unterschiedlichen Netzwerken befinden, ist die Verbindung ebenfalls möglich, erfordert jedoch öffentliche IP-Adressen, Port-Forwarding oder die Einrichtung eines VPNs. Zudem muss sichergestellt werden, dass keine Firewall den verwendeten Port blockiert.

Sind diese Voraussetzungen erfüllt, kann der Datenaustausch problemlos stattfinden. Dies wird durch zwei einfache Befehle ermöglicht. Auf einem Gerät wird der Server- bzw. Listening-Modus mit `nc -l [Port]` oder `nc -lu [Port]` aktiviert, während auf dem anderen Gerät der Client- bzw. Verbindungsmodus mit `nc [IP-Adresse] [Port]` oder `nc -u [IP-Adresse] [Port]` eingerichtet wird. Um das Protokoll direkt festzulegen, können die Optionen `-t` für TCP oder `-u` für UDP verwenden. Ein Abschnitt der Kommunikation kann in Abbildung 22 gesehen werden.

```

^C
chiara@chiaraAlpha:~/config/srsran$ nc -l 1234
hi

so...essen?

pk, dann ab geht

```

Abbildung 22: Beispiel für einen Nachrichtenaustausch per NetCat.

In unserem Fall verwendeten wir zwei private Rechner mit Linux-Distributionen, die über die neuesten Netcat-Versionen verfügten und jeweils mit einem srsUE (Siehe Abschnitt 3.3.4) in

unser 5G-Netzwerk verbunden waren. Der Vorteil dieser Umsetzung lag in der einfachen Testdurchführung dank der Softwarekompatibilität und des gemeinsam genutzten Netzwerks. Ein großer Nachteil war jedoch die erwähnte instabile Verbindung, bei der der Nachrichtenaustausch gelegentlich unterbrochen wurde (Siehe Abschnitt 3.3.4).

4.5 Netzwerktest durch Simulation von Netzwerkverkehr (Bennet Kießling)

Die Durchführung von Netzwerktests ist essenziell, um die Leistungsfähigkeit eines Kommunikationsnetzes zu evaluieren, Schwachstellen zu identifizieren und Potenziale für Optimierungen zu erkennen. Insbesondere in 5G-Netzwerken, wie dem im Rahmen dieses Projekts verwendeten Open5GS 5G SA Netzwerk, stellen Parameter wie Jitter, Latenz, Paketverlust und maximale Bitrate zentrale Indikatoren für die Netzqualität dar. Diese Metriken sind besonders relevant für Anwendungen mit hohen Anforderungen an die Verbindungsstabilität und -qualität, beispielsweise für Echtzeitkommunikation (VoIP) und latenzkritische Dienste wie autonomes Fahren [13]. Die durchgeführten Tests zielen darauf ab, die Effizienz, Kapazität und Stabilität des 5G-Netzwerks zu bewerten.

4.5.1 Getestete Parameter (Bennet Kießling)

Jitter beschreibt die Schwankungen in der Verzögerung von Datenpaketen während der Übertragung. In einem 5G-Netzwerk kann eine unregelmäßige Übertragung zu erheblichen Qualitätseinbußen führen. Eine stabile Datenübertragung erfordert möglichst geringe Jitter-Werte. Mit iPerf3 lässt sich Jitter durch den Versand von UDP-Datenpaketen messen, wobei die Schwankungen in den Antwortzeiten analysiert werden. In einem optimal funktionierenden 5G-Netzwerk sollten diese Schwankungen minimiert werden, um eine gleichbleibend hohe Kommunikationsqualität zu gewährleisten [20]. Latenz bezeichnet die Verzögerung, die ein Datenpaket von der Quelle bis zum Ziel benötigt. In einem 5G SA-Netzwerk wird eine niedrige Latenz erwartet. Mithilfe von iPerf3 kann die Latenz durch den Einsatz von Pings sowie durch die Übertragung von TCP- oder UDP-Paketen gemessen werden. Dabei wird die Reaktionszeit des Netzwerks erfasst, was eine zentrale Kennzahl für die Netzwerkleistung darstellt. Paketverlust tritt auf, wenn Datenpakete während der Übertragung nicht an ihrem Ziel ankommen. In drahtlosen 5G-Netzwerken kann dies durch Störungen im Signalweg oder eine Überlastung des Netzwerks verursacht werden [20]. Ein niedriger Paketverlust ist entscheidend für die Sicherstellung einer hohen Dienstgüte (QoS) [16]. iPerf3 ermöglicht die Messung des Paketverlusts insbesondere durch UDP-Tests. Diese geben Einblick in die Zuverlässigkeit des Netzwerks, indem der Prozentsatz der verlorenen Pakete bestimmt wird. Die maximale Datenübertragungsrate eines Netzwerks spiegelt dessen Bandbreitenkapazität wider. In einem 5G-Netzwerk, das für hohe Datenraten und geringe Latenzen ausgelegt ist, ist die Ermittlung der maximalen Datenrate ein weiterer Schritt zur Bewertung der Netzwerkeffizienz. Hier wurde iPerf3 verwendet um die maximale Bandbreite über UDP-Tests

zu messen und die erzielten Datenraten zwischen unterschiedlichen Endgeräten wie srsUE und handelsüblichen Smartphones zu vergleichen [28]. Dies ermöglicht eine detaillierte Analyse der Netzwerkkapazität und hilft dabei, Engpässe in der Netzwerkperformance zu identifizieren, die zu Lastspitzen oder ineffizienten Übertragungen führen könnten [16].

4.5.2 Getestete Komponenten (Bennet Kießling)

In der Netzarchitektur eines 5G SA-Netzwerks, wie sie in diesem Projekt zum Einsatz kommt, spielen mehrere Komponenten eine zentrale Rolle bei der Datenübertragung. Dazu gehören das UE, die gNB, das AMF, das SMF sowie das UPF. Die iPerf3-Tests konzentrieren sich vorwiegend auf die Schnittstellen zwischen dem UE, der gNB und der UPF, da diese den Hauptteil der Datenübertragung bestimmen. Die Verbindung zwischen diesen Komponenten beeinflusst maßgeblich die Datenraten, Latenz, den Jitter und den Paketverlust, die durch die Tests bewertet werden [8]. Durch die Simulation von Netzwerklast zwischen dem UE und der gNB sowie die weiterführende Kommunikation über die UPF wird eine umfassende Analyse des User Plane-Verkehrs und der Datenpfade durchgeführt.

4.5.3 iPerf3 (Bennet Kießling)

Zur Durchführung der Netzwerktests kommt das Tool iPerf3 zum Einsatz. Es handelt sich um ein weitverbreitetes Open-Source-Tool zur Netzwerkanalyse, das die Simulation von Netzwerkverkehr unter realistischen Bedingungen ermöglicht und sowohl TCP- als auch UDP-Tests unterstützt. Während TCP-Tests hauptsächlich zur Messung des maximalen Durchsatzes dienen, wird UDP bevorzugt für die Analyse von Jitter und Paketverlust verwendet [17]. Bei der Einrichtung der Testumgebung wird zunächst sichergestellt, dass alle Komponenten korrekt konfiguriert sind. Dies umfasst die Installation von iPerf3 auf den entsprechenden Geräten, sowohl auf dem Server als auch auf dem UE. Hierfür werden die Befehle `sudo apt update` und `sudo apt install iperf3` genutzt. Zur Messung des maximalen Durchsatzes wird iPerf3 sowohl auf einem Server im Internet als auch auf dem UE installiert. Die Tests erfolgen in beide Richtungen, um sowohl die Download- als auch die Upload-Geschwindigkeit zu bestimmen. Ein typischer Befehl zur Durchführung eines Download-Tests lautet: `iperf3 -c <server-ip> -u -b 10M -t 30`. Dabei steht `-c <server-ip>` für die IP-Adresse des iPerf3-Servers, `-u` für die Nutzung von UDP, was besonders für die Messung von Jitter und Paketverlust wichtig ist. Mit `-b 10M` wird die zu sendende Bandbreite (in diesem Fall 10 Mbps) festgelegt, und `-t 30` spezifiziert die Testdauer von 30 Sekunden.

4.5.4 Ergebnisse (Bennet Kießling und Johann Rittenschober)

Upload-Tests mit srsUE (Bennet Kießling) In den Upload-Tests mit dem srsUE werden drei Versuche unter verschiedenen Testbedingungen durchgeführt, um die maximale stabile

Upload-Bitrate sowie die Auswirkungen auf Jitter und Paketverlust zu untersuchen. Im ersten Versuch wird eine sehr niedrige Bitrate von 1 Kbit/s bei einer Paketgröße von 16 Bytes verwendet. Unter diesen Bedingungen werden insgesamt 7,33 KBytes an Daten übertragen, was einer durchschnittlichen Bitrate von 999 bits/s entspricht. Der Jitter beträgt bis zu 9,679 ms, während kein Paketverlust auftritt. Dies zeigt, dass trotz der niedrigen Bitrate eine zuverlässige Übertragung ohne Datenverluste möglich ist, allerdings mit deutlichen Schwankungen in der Verzögerung. Im zweiten Versuch wird die Bitrate auf 11,2 Kbit/s erhöht und die Paketgröße auf 1400 Bytes angepasst. Die übertragene Datenmenge steigt auf 83,4 KBytes bei einer durchschnittlichen Bitrate von 11,4 Kbits/s. Der Jitter ist mit maximal 0,172 ms deutlich geringer als im ersten Test, was auf eine stabilere Übertragung bei dieser Bitrate hinweist. Auch hier wird kein Paketverlust festgestellt, was eine stabile Verbindung unter diesen Bedingungen zeigt. Im dritten und letzten Versuch wird die Bitrate weiter auf 700 Kbit/s erhöht, um die maximale Leistungsfähigkeit des srsUE zu testen. Es werden 5,01 MBytes an Daten übertragen, was einer durchschnittlichen Bitrate von 699 Kbits/s entspricht. Der Jitter liegt bei bis zu 4,952 ms, und erneut tritt kein Paketverlust auf. Die Tests zeigen jedoch, dass die Verbindung zum srsUE bei höheren Bitraten instabil wird und ein Verbindungsabbruch stattfindet, was darauf hinweist, dass 700 Kbit/s die maximale stabile Bitrate für das srsUE darstellt. Insgesamt zeigen die Upload-Tests mit dem srsUE, dass die Verbindung bei niedrigeren Bitraten (bis 700 Kbit/s) stabil ist, ohne dass Paketverluste auftreten. Der Jitter variiert stark, insbesondere bei sehr niedrigen Bitraten, stabilisiert sich jedoch mit zunehmender Bitrate.

Testversuch	Testbedingungen	Übertragene Daten	Durchschnittliche Bitrate	Jitter	Paketverlust
Upload mit SRSUE - Erster Versuch	Bitrate: 1 Kbit/s, Paketgröße: 16 Bytes, iPerf3: iperf3 -c <srsRAN_IP> -u -b 1k -t 60 -n 16	7,33 KBytes	999 bits/sec	9,679 ms	0% (0/469)
Upload mit SRSUE - Zweiter Versuch	Bitrate: 11,2 Kbit/s, Paketgröße: 1400 Bytes, iPerf3: iperf3 -c <srsRAN_IP> -u -b 11.2k -t 60 -n 1400	83,4 KBytes	11,4 Kbits/sec	0,172 ms	0% (0/61)
Upload mit SRSUE - Dritter Versuch	Bitrate: 700 Kbit/s, iPerf3: iperf3 -c <srsRAN_IP> -u -b 700k -t 60	5,01 MBytes	699 Kbits/sec	4,952 ms	0% (0/3895)
Upload mit Smartphone	Maximum Bitrate	425 MBytes	59,1 Mbits/sec	0,159 ms	0% (0/330620)

Abbildung 23: Upload-Tests.

Upload-Tests mit Smartphone (Bennet Kießling) Im Vergleich dazu zeigen die Upload-Tests mit einem handelsüblichen Smartphone eine deutlich höhere Leistungsfähigkeit. Es wird eine hohe Bitrate ohne spezifische Begrenzung verwendet. Dabei können 425 MBytes an Daten übertragen werden, was einer durchschnittlichen Bitrate von 59,1 Mbits/s entspricht – rund das 85-fache der Leistung des srsUE. Der gemessene Jitter ist mit maximal 0,159 ms sehr niedrig, was auf eine stabile Verbindung hinweist. Auch bei diesen Tests wird kein Paketverlust festgestellt,

was die Zuverlässigkeit der Verbindung bestätigt.

Download-Tests mit srsUE (Bennet Kießling) Bei den Download-Tests zeigt das srsUE im Vergleich zu den Upload-Tests eine deutlich schlechtere Leistung. Die maximale gemessene Download-Bitrate liegt bei unter 10 Kbit/s, was sehr niedrig ist und darauf hindeutet, dass entweder das Gerät selbst oder die Konfiguration des Netzwerks eine bessere Leistung verhindert. Der durchschnittliche Jitter liegt bei etwa 0,5 ms, was auf eine grundsätzlich stabile, jedoch langsame Verbindung hinweist.

Download-Tests mit Smartphone (Johann Rittenschober) Beim Smartphone zeigt sich hier wieder im Vergleich zum srsUE eine deutlich bessere Leistung, allerdings – analog zum srsUE – wieder schlechter als der zugehörige Upload-Test. So wird hier eine maximale Bitrate ohne Verluste von 23,3 MBit/s erreicht, der durchschnittliche Jitter ist ca. 0,9 ms.

Testversuch	Testbedingungen	Übertragene Daten	Durchschnittliche Bitrate	Jitter	Paketverlust
Download mit SRSUE	Bitrate unbegrenzt		Max. < 10 Kbit/s	0,5 µs	
Download mit Smartphone - Erster Versuch	Bitrate: unbegrenzt, iPerf3: iperf3 -c 10.45.0.56 -u -b 0 -t 30	93,8 MBytes (Empfänger) 2	24,6 Mbits/sec	0,651 ms	100% (16339023/16412020 Pakete)
Download Maximum bei dem keine Packets verloren gehen	Bitrate: 23,4 Mbit/s, iPerf3: iperf3 -c 10.45.0.57 -u -b 23.4M -t 30	83,7 MBytes	23,3 Mbits/sec	0,919 ms	0% (0/65095)

Abbildung 24: Download-Tests.

Leistungsunterschiede (Bennet Kießling) Die durchgeführten Tests verdeutlichen die Leistungsunterschiede zwischen dem srsUE und einem modernen Smartphone. Bei den Upload-Tests erreicht das srsUE eine maximale stabile Bitrate von 700 Kbit/s, während das Smartphone eine durchschnittliche Upload-Bitrate von 59,1 Mbits/s erreicht, was etwa das 85-fache der srsUE-Leistung entspricht. Bei den Download-Tests ist das ähnlich. Das srsUE erreicht nur eine sehr geringe Download-Bitrate von unter 10 Kbit/s, was die Nutzbarkeit stark einschränkt, während das Smartphone bei angepasster Testkonfiguration eine stabile Bitrate von 23,3 Mbits/s erreicht.

Die Testparameter, insbesondere die Bitrate-Einstellungen, spielen eine wesentliche Rolle bei den Ergebnissen. Zu hohe Bitraten führen sowohl beim srsUE als auch beim Smartphone zu Instabilitäten und Verbindungsabbrüchen. Angepasste Bitraten ermöglichen hingegen stabile Verbindungen ohne Paketverlust. Auch die Wahl der Paketgröße hat Einfluss auf die Testergebnisse: Kleinere Paketgrößen führen tendenziell zu höheren Jitter-Werten, was die Bedeutung einer optimalen Konfiguration der Testparameter unterstreicht. Jitter ist in den Tests generell niedriger bei Smartphones, was auf eine bessere Netzwerkkonfiguration hinweist. Beim srsUE ist der Jitter insbesondere bei niedrigen Bitraten höher. In Bezug auf den Paketverlust gibt es bei

angepassten Bitraten keine signifikanten Verluste, während unrealistische Testbedingungen, wie die Verwendung einer unbegrenzten Bitrate, zu einem hohen Paketverlust führen.

Leistungsunterschiede nach Antennen-Gain (Johann Rittenschober) Zusätzlich zu den zuvor beschriebenen Tests, wurden manche davon mit unterschiedlichen Werten für den RX und TX Gain der gNB Antennen wiederholt. Dafür wurde das Smartphone als zuverlässigeres und schnelleres UE gewählt, auch da es die eigenen Antennen automatisch anpasst um die weitere Variable der srsUE Antennen auszuschließen. Die Annahme war, dass sich die Änderung des RX Gain hauptsächlich auf die Upload-Leistung, und die des TX Gain auf die Download-Leistung auswirkt. Diese Annahme konnte bestätigt werden, indem durch eine (moderate) Erhöhung des RX Gains der Upload leicht verbessert werden konnte, durch eine Reduzierung aber ebenfalls deutlich verschlechtert wurde. Der TX Gain ist mit 80 bereits nahe am oberen Limit (89), eine Reduzierung hat die Download Leistung deutlich verschlechtert. Das Handy hat dann kaum noch Netz-Empfang angezeigt und die Verbindung war zunehmend instabil.

5 Erfolglose Tests

In diesem Kapitel werden durchgeführte Experimente beschrieben, die nicht erfolgreich abgeschlossen werden konnten.

5.1 Handover

Eines der Experimente, die im Rahmen dieses Projektes durchgeführt wurden, ist die Realisierung des Handovers in einem 5G SA Netz.

5.1.1 Theorie (Marina Trinz)

Das Handover ist ein entscheidendes Verfahren für die Aufrechterhaltung der Verbindung eines UE zum mobilen Netz. Dabei wird vom Netz für das UE das Wechseln von einer Zelle zur anderen ermöglicht. Das Wechseln kann aufgrund von schwacher Signalstärke, hoher Netzauslastung, Bewegung des UE oder zur Gewährleistung von Quality of Service-Anforderungen durchgeführt werden. Solche Daten wie Signalstärke werden vom UE gemessen und als ein Measurement Report an das Netz geschickt, welches das Wechseln der Zelle initiiert [43, p. 187]. Ein reibungsloses Handover ist essenziell, um Unterbrechungen in der Verbindung zu vermeiden, insbesondere bei Echtzeitanwendungen wie VoIP oder Videostreaming. Die kontinuierliche Verbindung stellt sicher, dass Nutzer nahtlos von einer Zelle zur nächsten wechseln können, ohne Qualitätsverluste zu erleben.

Beim Handover in 5G-Netzen unterscheidet man zwischen Intra- und Inter-Handover, wobei das Intra-Handover das Wechseln innerhalb einer Basisstation und das Inter-Handover das

Wechseln zwischen zwei unterschiedlichen Basisstationen kennzeichnet.

Zwei wichtige Komponenten einer Basisstation in 5G-Netzen, genannt gNodeB (gNB), sind die CU und die DU [1]. Die CU übernimmt Steuerung und Verwaltung von Verbindungen, und die DU ist für die Verarbeitung der Daten verantwortlich. Eine Basisstation kann aus mehreren DUs zusammengesetzt werden, die von einer CU gesteuert werden. Dabei kann eine DU mehrere Zellen kontrollieren [12].

Beim Intra-Handover unterscheidet man zwischen dem Wechseln zwischen zwei DUs einer Basisstation oder zwei Zellen einer DU. Das Wechseln zwischen zwei DUs wird als Inter-gNB-DU-Handover bezeichnet, während das Wechseln zwischen zwei Zellen einer DU als Intra-gNB-DU-Handover genannt wird [6]. Die Aufteilung der gNB-Architektur in CU und DU ermöglicht beim Intra-Handover eine flexible und effiziente Verwaltung der Netzwerkressourcen, da keine zusätzliche Verbindung zur AMF benötigt wird.

Das Inter-Handover kann in Xn-Interface-based und N2-Interface-based aufgeteilt werden [4, p. 169]. Ob ein Xn-based Inter-Handover oder ein N2-based Inter-Handover verwendet wird, hängt davon ab, ob die Verbindung zwischen zwei Basisstationen über das Xn-Interface besteht. Wenn die Verbindung zwischen zwei Basisstationen fehlt, wird das Handover über das AMF und das N2-Interface gesteuert.

5.1.2 Inter Handover (Chiara Piccolroaz und Sarah König)

Ein zentraler Bestandteil unseres Projekts war also der Versuch, ein Handover in unserem 5G-Netz zu implementieren. Dazu haben wir zunächst einen automatischen Handover des Pixel 6A getestet, indem wir das Handy zwischen zwei möglichst weit voneinander entfernten gNBs bewegt haben. Wir erwarteten einen erfolgreichen Handover, da unser implementiertes 5G-Netz laut der offiziellen Open5GS-Website über die notwendigen Zugangs- und Mobilitätsmanagementfunktionen [18] und spezifische Funktionen für einen N2-basierten Handover verfügt [19]. Leider ist dieser erwartete Handover nicht eingetreten, weswegen wir einige Einstellungen unserer gNB überprüft und angepasst haben.

Zuerst haben wir die Option `trigger_handover_from_measurements` aktiviert, jedoch ohne die erhoffte Verbesserung. Daraufhin konfigurierten wir zusätzliche Parameter, wie Zellen, Nachbarzellen und die ereignisgesteuerte Berichtskonfiguration vom Typ `a3`. Dies führte zu mehreren Fehlermeldungen. Unter anderem wurden alle für den Typ `a3` spezifischen Parameter nicht erkannt, obwohl diese in der Konfigurationsreferenz [29] dokumentiert sind. Außerdem sollte der nicht dokumentierte Parameter `event_triggered_report_type` gesetzt werden.

Nach Hinzufügen dieses Parameters trat eine neue Fehlermeldung auf, die fehlende Parameter wie PCI, Band und SSB für externe Zellen bemängelte. Wir fügten diese Parameter zusammen mit der `gnb_id` ein und ergänzten im Abschnitt `cells` weitere Parameter wie PCI, ARFCN und PRACH für beide gNBs.

Trotz aller Anpassungen blieb die Fehlermeldung bestehen, dass das `execution_profile` nicht gesetzt und die A3-Report-Konfiguration ungültig sei, was zu einem „bad optional access“ führte und den Start der gNB verhinderte. Dies war logisch, da die entsprechenden `a3_`-Parameter nicht akzeptiert wurden. Auch nach Setzen des `execution_profile` auf `quad` blieb die Fehlermeldung bestehen - ein Problem, das auch bei normalen Startversuchen der gNB auftrat. Ein erneutes Setzen der `a3_`-Parameter führte wieder zu der Fehlermeldung, dass die Parameter nicht analysiert werden konnten. Daraufhin haben wir die Report-Typen A1 bis A6 ausprobiert, aber alle Konfigurationen führten zu ungültigen Report-Einstellungen, da die notwendigen Parameter fehlten.

Unsere Vermutung ist, dass weder Inter-Handover noch unser USRP B210 unterstützt werden. Nach der offiziellen srsRAN-Website wird derzeit nur ein Intra-Cell-Handover mit USRPs der X- oder N-Serie unterstützt [31], was erklären könnte, warum die Konfigurationseinstellungen entweder nicht erkannt oder falsch interpretiert wurden.

5.1.3 Intra Handover (Marina Trinz)

Im zweiten Experiment mit dem Handover in 5G-Netzen haben wir versucht, das Intra-Handover umzusetzen. Laut der offiziellen Seite von srsRAN wird das Inter-gNB-DU-Handover von der gNB-Software unterstützt [31]. Dabei wird ein gNB auf zwei DUs aufgeteilt, die jeweils eine Funkzelle kontrollieren. Die DUs werden weiterhin nur von einer CU gesteuert, welche die Verbindung zur AMF herstellt (vgl. Abbildung 25).

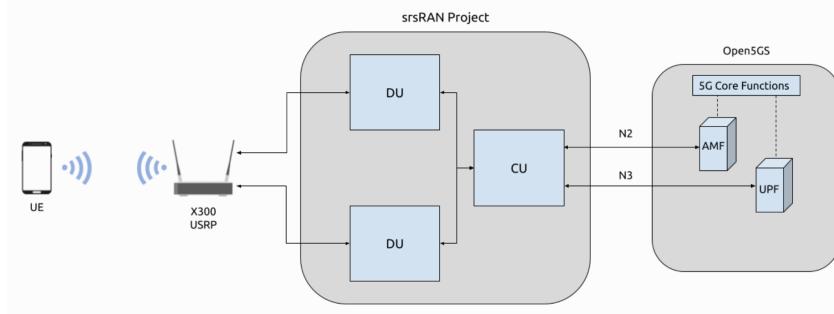


Abbildung 25: Architektur der Implementierung für Intra-Handover von srsRAN (Quelle [31]).

Für die Durchführung dieses Inter-gNB-DU-Handover von srsRAN wird ein Funksystem benötigt, das zwei unabhängige Radiofrequenzen für jedes DU realisieren kann. Dafür haben wir das softwaredefinierte Funksystem USRP X310 verwendet.

Die Konfiguration eines gNB wird durch eine Konfigurationsdatei definiert, die beim Starten des gNB als Übergabeparameter angegeben wird. srsRAN stellt eine fertige Konfigurationsdatei `handover_x310.yml` für das Intra-Handover zur Verfügung. Diese Datei wird in diesem Experiment verwendet. Im Vergleich zur Konfigurationsdatei für ein gNB ohne Aufteilung auf zwei DUs werden in der Konfigurationsdatei für das Intra-Handover weitere Parameter wie `cells` und

`cu_cp mobility` definiert. Über den Parameter `cells` wird die Anzahl von Funkzellen angegeben. Im Bereich `cu_cp mobility` werden die Funkzellen näher beschrieben, und der Parameter `trigger_handover_from_measurements` wird auf `true` gesetzt.

Nach dem Aktualisieren aller Konfigurationsdateien wurde das Funksystem mit dem Rechner über das PCI-Kabel verbunden. Alle benötigten Treiber für das Funksystem USRP X310 wurden auf dem System installiert. Um die Verbindung zum Funksystem zu überprüfen, wurde in der Kommandozeile der Befehl `uhd_find_devices` eingegeben. Die Ausgabe auf der Kommandozeile lautete `No devices found`. Nach dem Aktualisieren der Treiber konnte das Funksystem weiterhin nicht mit dem Rechner verbunden werden. Das Verbindungsproblem konnte nicht eliminiert werden.

5.2 SMS (Chiara Piccolroaz)

Eine weitere Anwendungsidee war der Austausch von SMS über unser 5G-Netzwerk. Nach unseren Recherchen werden SMS in 5G jedoch nicht nativ über das Netz übertragen, sondern über das IP Multimedia Subsystem (IMS) oder über die SMS Function (SMSF) im Kernnetz [22]. Die Implementierung in unserem Projekt wurde erheblich erschwert, da die Unterstützung der SMSF in Open5GS für 5G-Netze noch in der Entwicklung zu sein scheint [25]. Außerdem würde die IMS-Implementierung eine separate Infrastruktur erfordern, wie z.B. einen SIP-Server oder die Integration von 4G/LTE [24]. Aufgrund fehlender Dokumentation und zeitlicher Ressourcen wurden diese Ansätze in unserem Projekt nicht weiter verfolgt.

Für zukünftige Arbeiten könnten jedoch VoIP oder Instant Messaging über MQTT sowie WebSocket-basierte Lösungen als praktikable Alternativen für die Kommunikation in Betracht gezogen werden. VoIP (Voice over Internet Protocol) ermöglicht die Übertragung von Sprachkommunikation über das Internet und stellt damit eine kostengünstige und flexible Alternative zu herkömmlichen Telefonleitungen dar. Instant Messaging über MQTT ist ein leichtgewichtiges Protokoll, das für den Austausch von Nachrichten zwischen Geräten in Echtzeit optimiert ist. WebSocket-basierte Lösungen ermöglichen eine bidirektionale Kommunikation zwischen Client und Server über eine einzige, persistente Verbindung, die Echtzeitanwendungen wie Chats oder Benachrichtigungen unterstützt.

6 Diskussion (Alle)

5G ist eine relativ neue Technologie, die sich noch in der Entwicklungsphase befindet. Viele Technologien von 5G wurden erst vor Kurzem durch 3GPP standardisiert. Somit ist es schwer für die Entwickler von Open-Source-Forschungssoftware wie open5gs und srsRAN, die in diesem Projekt verwendet wurden, eine Implementierung bereitzustellen, die dem aktuellen Stand der 3GPP-Standards entspricht. Dadurch erhöht sich der Zeitaufwand für die Installation der

Netzkomponenten enorm. Viele 5G-Technologien wie SMS oder Handover, die in diesem Projekt untersucht wurden, waren nur teilweise oder manchmal gar nicht für die SA-Version von 5G implementiert. Für ein zeitlich sehr begrenztes Projekt wird dieser zusätzliche Zeitaufwand schnell zu einem Hindernis. Die Hälfte der für dieses Projekt zugeteilten Zeit wurde allein für den Aufbau einer intakten Verbindung zwischen dem Kernnetz, dem gNB und dem UE benötigt.

Ein zusätzlicher Aufwand in diesem Projekt entstand durch die Hardware. Im Rahmen des aktuellen Projekts wurden die bereitgestellten Rechner bereits für zahlreiche Installationen verwendet. Um sicherzustellen, dass keine Konflikte mit bestehenden Installationen auftreten, ist es ratsam, die Rechner vor Beginn des Projekts neu zu booten. Dies gewährleistet eine saubere Arbeitsumgebung und minimiert potenzielle Störungen durch vorherige Softwareinstallationen.

Ein weiterer wichtiger Aspekt ist die Leistungsfähigkeit der Rechner. Nicht alle zur Verfügung stehenden Geräte sind für die Installation von 5G-Komponenten geeignet, da einige unserer Meinung nach nicht über ausreichende Rechenleistung verfügen. Auf einem der Rechner sind sowohl die Installation eines srsUE als auch eines gNB für das Handover fehlgeschlagen, während es auf den anderen Rechnern reibungslos funktionierte. Darüber hinaus konnten einige Experimente, wie beispielsweise das Intra-Handover, aus Zeitgründen nicht abgeschlossen werden. Dies lag daran, dass die benötigte Hardware, das Funksystem USRP X310, von einer anderen Gruppe verwendet wurde.

Nach der erfolgreichen Installation von Netzkomponenten war ein zentrales Problem bei der Nutzung von srsUE die instabile und unvorhersehbare Verbindung, die hauptsächlich durch RRC- (Radio Resource Control) und RLF- (Radio Link Failure) Fehlermeldungen verursacht wurde. In einigen Fällen konnte die Verbindung für einige Minuten aufrechterhalten werden, in anderen Fällen brach sie nach wenigen Sekunden ab.

Ein weiteres mögliches Hindernis waren Frequenzinterferenzen. Da andere Gruppen teilweise die gleichen Bänder und Frequenzen nutzten, war es schwierig festzustellen, ob die bisher augetretenen Fehlermeldungen und Fehlversuche - einschließlich der instabilen und unvorhersehbaren Netzwerkverbindung zu srsUE - auf Frequenzinterferenzen oder auf die bestehende Hardware- oder Softwareimplementierung unseres Projektes zurückzuführen waren.

Weiterhin war der eigentlich geplante Umfang der Netzwerktests durch den Versuchsaufbau eingeschränkt, da es nicht möglich war eine Verbindung auf verschiedenen Bändern mit unterschiedlichen Frequenzen aufzubauen, weshalb diese auch nicht getestet werden konnten.

Die Netzwerktests, die durchgeführt werden konnten, lassen softwareseitige Einschränkungen mit dem srsRAN Projekt bzw. dem srsUE vermuten, da die Performace damit in unserem 5G Netz entgegen den Erwartungen um ein Vielfaches schlechter war als mit dem Handy. Auch dass die Download Datenrate im Gegensatz zum Upload – sowohl auf srsUE als auch auf dem Handy – deutlich schlechter war, war so nicht erwartet.

Die verschiedenen Installationen haben gezeigt, dass ein Build from Source in der Regel die

Beste Wahl ist um die neuesten Versionen zu erhalten bzw. die meiste Kontrolle und Flexibilität zu haben, auch wenn es nicht immer problemlos funktioniert. Besonders hier in dem relativ neuen Bereich des 5G Standalone war das wichtig.

Insgesamt lässt sich festhalten, dass es eine sehr große Menge an Fehlerquellen gibt, wie zum Beispiel die verschiedenen Konfigurationsparameter und deren Wechselwirkung, die Eigenschaften der verwendeten Antennen und deren Ausrichtung, Frequenzinterferenzen, mögliche Reflektionen der Funksignale im Raum, die Clock der Funkgeräte, die Prozessor Leistung und deren Konfiguration, Betriebssystem Verhalten im Batteriebetrieb, und mehr. Eine Fehlerquelle ist dadurch sehr schwer zu finden, vor allem wenn in all diesen Bereichen die Expertise fehlt. Das kann einerseits zu vermeintlich dummen Fehlern führen, wo eine unwissentlich schlechte Konfiguration große Auswirkungen hat. Andererseits kann auch passieren, dass man beim Debugging mit einem hohen Zeitaufwand viele Varianten methodisch testet, das eigentliche Problem aber an einer ganz anderen Stelle zu finden ist.

7 Fazit (Alle)

Die Implementierung eines 5G-Netzwerks stellte für uns eine Herausforderung dar, insbesondere aufgrund der noch relativ neuen Architektur und unserer Verwendung von Open-Source-Tools. Die verwendeten Technologien befinden sich noch in einem fortlaufenden Entwicklungsprozess, was bedeutet, dass einige Komponenten nicht vollständig getestet oder nur teilweise implementiert sind. Das erschwert die Entwicklung im Vergleich zu länger etablierten und gründlich erprobten Lösungen. Zudem sind die verfügbaren Quellen begrenzt, und auf häufige Probleme gibt es oftmals keine zufriedenstellenden Antworten.

Das Identifizieren potenzieller Fehlerquellen wird auch durch die Vielzahl an Variablen zusätzlich kompliziert, was die letztendliche Lokalisierung von Problemen erheblich erschwert. Dadurch hatten wir auch kein zuverlässiges, stabiles System als Basis für weiteres Debugging.

Somit verlängert sich die Zeit, bis man ein funktionierendes Netzwerk hat und UEs, die sich erfolgreich verbinden können, und man muss auf dem Weg dahin viel Geduld aufbringen und viele verschiedene Ansätze ausprobieren. Dennoch war es dann sehr bereichernd, als unser Netzwerk schließlich erfolgreich lief.

Um jedoch die Stabilität unseres aufgesetzten Netzwerks langfristig zu gewährleisten, wäre ein deutlich größerer Zeitaufwand für das Debugging notwendig, mehr Kenntnisse in den relevanten Fachbereichen sowie die Möglichkeit, Störfaktoren wie konkurrierende Netzwerke auszuschließen.

Literatur

- [1] 3GPP. *5G System Overview*. URL: <https://www.3gpp.org/technologies/5g-system-overview> (besucht am 30.09.2024).
- [2] 3GPP. *Authentication and Key Management for Applications (AKMA) in 5G*. URL: <https://www.3gpp.org/technologies/akma> (besucht am 30.09.2024).
- [3] 3GPP. *Technical Specification 23.501*.
- [4] 3GPP. *Technical Specification 23.502*.
- [5] 3GPP. *Technical Specification 38.300*.
- [6] 3GPP. *Technical Specification 38.401*.
- [7] Ettus Research AnNIBrand. *VERT900 Antenna*. 2024. URL: <https://www.ettus.com/all-products/vert900/> (besucht am 27.09.2024).
- [8] T.O. Atalay u. a. „Demystifying 5G Traffic Patterns with an Indoor RAN Measurement Campaign“. In: *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*. Kuala Lumpur, Malaysia, 2023, S. 1185–1190. DOI: [10.1109/GLOBECON54140.2023.10437330](https://doi.org/10.1109/GLOBECON54140.2023.10437330).
- [9] Nuradio Concepts. *5G Network*. 2024. URL: https://github.com/Nuradioconcepts/5G_Network (besucht am 27.09.2024).
- [10] Nuradio Concepts. *Home 5G Network - srsRAN gNB, srsUE, and Open5gs*. 2024. URL: <https://www.youtube.com/watch?v=Av640iJS99M> (besucht am 27.09.2024).
- [11] DeLOCK. *Delock 5G 3,4 - 3,8 GHz Antenne SMA Stecker 5 dBi 20 cm omnidirektional mit Kippgelenk und flexiblem Material schwarz*. 2024. URL: <https://www.delock.de/produkt/12638/merkmale.html?f=s> (besucht am 27.09.2024).
- [12] Devopedia. *5G Handover*. URL: <https://devopedia.org/5g-handover> (besucht am 30.09.2024).
- [13] X. Fan und Y. Huo. *An Overview of Low Latency for Wireless Communications: An Evolutionary Perspective*. 2021. DOI: [10.48550/arXiv.2107.03484](https://arxiv.org/abs/2107.03484).
- [14] Python Software Foundation. *http.server - HTTP servers*. 2024. URL: <https://docs.python.org/3/library/http.server.html> (besucht am 27.09.2024).
- [15] F. Hofbauer. „5G in München“. In: *Digitale Welt* 3 (2019), S. 14–20. DOI: [10.1007/s42354-019-0144-4](https://doi.org/10.1007/s42354-019-0144-4).
- [16] M. Hoppari u. a. „Performance of the 5th Generation Indoor Wireless Technologies: Empirical Study“. In: *Future Internet* 13.180 (2021). DOI: [10.3390/fi13070180](https://doi.org/10.3390/fi13070180).

- [17] K.-L. Lee, C.-N. Lee und M.-F. Lee. „Realizing 5G Network Slicing Provisioning with Open Source Software“. In: *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Tokyo, Japan, 2021, S. 1923–1930.
- [18] Open5Gs - Sukchan Lee. *Quickstart*. 2024. URL: <https://open5gs.org/open5gs/docs/guide/01-quickstart/> (besucht am 27.09.2024).
- [19] Open5Gs - Sukchan Lee. *v2.1.5 - 5G Core N2 based handover*. 2021. URL: <https://open5gs.org/open5gs/release/2021/02/02/release-v2.1.5.html> (besucht am 30.09.2024).
- [20] N. Mesbahi und H. Dahmouni. „Delay and jitter analysis in LTE networks“. In: *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. Fez, Morocco, 2016, S. 122–126. DOI: [10.1109/WINCOM.2016.7777202](https://doi.org/10.1109/WINCOM.2016.7777202).
- [21] miniPC. *Quectel RM500Q-AE 3G/4G/LTE/5G M.2 NGFF Modem*. 2023-2024. URL: <https://www.minipc.de/de/catalog/il/3020> (besucht am 27.09.2024).
- [22] Nick vs Networking - Telco Network Engineering. *How to Use Netcat Commands: Examples and Cheat Sheets*. 2023-2024. URL: <https://nickvsnetworking.com/sms-in-5gc/>.
- [23] Open cells project. *UICC/SIM programing*. URL: <https://open-cells.com/index.php/uiccsim-programing/> (besucht am 30.09.2024).
- [24] Github - open5gs. *Failure to attach after enabling Sounding reference signal on srsue*. 2023-2024. URL: https://github.com/srsran/srsRAN_4G/issues/937 (besucht am 27.09.2024).
- [25] Github - open5gs. *Is it possible to connect SMSF to Open5GS (support for N20/21 interface)?* 2023-2024. URL: <https://github.com/open5gs/open5gs/discussions/3243> (besucht am 27.09.2024).
- [26] Github - open5gs. *SrsUE can't maintain a connection to cell for more than a few seconds*. 2023-2024. URL: https://github.com/srsran/srsRAN_4G/issues/966 (besucht am 27.09.2024).
- [27] Github - open5gs. *srsUE cannot connect to gNb*. 2023-2024. URL: https://github.com/srsran/srsRAN_Project/issues/732 (besucht am 27.09.2024).
- [28] C. Park und A. Balasubramanian. „A Look Into the Performance of mmWave 5G, 3G, 4G/LTE using PC Application, iPerf3 and DASH“. In: *Journal of Student Research* 12.1 (2023). DOI: [10.47611/jsrhs.v12i1.4109](https://doi.org/10.47611/jsrhs.v12i1.4109).
- [29] srsRAN Project - Software Radio Systems. *Configuration Reference*. 2023-2024. URL: https://docs.srsran.com/projects/project/en/latest/user_manuals/source/config_ref.html#id1 (besucht am 27.09.2024).

- [30] srsRAN Project - Software Radio Systems. *Installation Guide*. 2023-2024. URL: https://docs.srsran.com/projects/project/en/latest/user_manuals/source/installation.html (besucht am 27.09.2024).
- [31] srsRAN Project - Software Radio Systems. *srsRAN gNB Handover*. 2023-2024. URL: <https://docs.srsran.com/projects/project/en/latest/tutorials/source/handover/source/index.html> (besucht am 30.09.2024).
- [32] srsRAN Project - Software Radio Systems. *srsRAN gNB with srsUE*. 2023-2024. URL: <https://docs.srsran.com/projects/project/en/latest/tutorials/source/srsUE/source/index.html> (besucht am 27.09.2024).
- [33] srsRAN Project 4G - Software Radio Systems. *5G SA srsUE*. 2019-2024. URL: https://docs.srsran.com/projects/4g/en/latest/app_notes/source/5g_sa_amari/source/index.html (besucht am 27.09.2024).
- [34] srsRAN Project 4G - Software Radio Systems. *Introduction SRS UE*. 2019-2024. URL: https://docs.srsran.com/projects/4g/en/latest/usermanuals/source/srsue/source/1_ue_intro.html (besucht am 27.09.2024).
- [35] srsRAN Project 4G - Software Radio Systems. *Release Notes – srsRAN 4G 23.11 Documentation*. URL: https://docs.srsran.com/projects/4g/en/latest/general/source/2_release_notes.html (besucht am 02.10.2024).
- [36] srsRAN Project 4G - Software Radio Systems. *Troubleshooting - Examining PCAPs with Wireshark*. 2023-2024. URL: https://docs.srsran.com/projects/4g/en/latest/general/source/4_troubleshooting.html#examining-pcaps-with-wireshark (besucht am 27.09.2024).
- [37] Quectel. *Quectel_RM50xQ_Series_Hardware_Design_V1.3*. 2023-2024. URL: https://www.quectel.com/download/quectel_rm50xq_series_hardware_design_v1-3/ (besucht am 27.09.2024).
- [38] Quectel. *RG50xQ&RM5xxQ Series updated AT Commands Manual*. 2023-2024. URL: https://www.quectel.com/download/quectel_rg50xqrm5xxq_series_at_commands_manual_v1-2/ (besucht am 27.09.2024).
- [39] Quectel. *RM50xQ Series Hardware Design*. 2023-2024. URL: https://sixfab.com/wp-content/uploads/2022/05/Quectel_RM50xQ_Series_Hardware_Design_V1.2.pdf (besucht am 27.09.2024).
- [40] GNU Radio. *Wiki Home*. URL: https://wiki.gnuradio.org/index.php?title=Main_Page (besucht am 02.10.2024).
- [41] Endace - Record.Response. *What is a PCAP file?* 2023-2024. URL: <https://www.endace.com/learn/what-is-a-pcap-file> (besucht am 27.09.2024).

- [42] 5G Tools for RF Wireless. *5G NR ARFCN*. 2024. URL: <https://5g-tools.com/5g-nr-arfcn-calculator/> (besucht am 27.09.2024).
- [43] Martin Sauter. *Grundkurs Mobile Kommunikationssysteme - 5G New Radio und Kernnetz, LTE-Advanced Pro, GSM, Wireless LAN und Bluetooth*). Springer Fachmedien Wiesbaden GmbH, 2022. ISBN: 9783658369620.
- [44] Software & Support Media GmbH. *Schwachstellen in SIM- und Smartcards auf der Black Hat USA 2015*. 2016. URL: <https://entwickler.de/mobile/schlechte-karten> (besucht am 30.09.2024).
- [45] Sqimway. *5G NR frequency band*. 2024. URL: https://www.sqimway.com/nr_band.php (besucht am 27.09.2024).
- [46] Varonis. *How to Use Netcat Commands: Examples and Cheat Sheets*. 2023-2024. URL: <https://www.varonis.com/blog/netcat-commands> (besucht am 27.09.2024).
- [47] Wireshark. *Chapter 5. File Input, Output, And Printing*. 2023-2024. URL: https://www.wireshark.org/docs/wsug_html_chunked/Ch10openSection.html (besucht am 27.09.2024).
- [48] Ulf Lampert Wireshark - Richard Sharpe Ed Warnicke. *Wireshark User's Guide - Version 4.5.0*. 2023-2024. URL: https://www.wireshark.org/docs/wsug_html/#ChIntroWhatIs.
- [49] Bartosz Zaczynski. *How to Launch an HTTP Server in One Line of Python Code*. 2024. URL: <https://realpython.com/python-http-server/> (besucht am 27.09.2024).

Anhang

Die im Folgenden aufgelisteten Anhänge sind separate PDFs, deren Seiten in der gegebenen Reihenfolge am Ende dieses Dokuments angefügt sind.

A srsUE Installation Guide (Johann Rittenschober)

B Spectrum Analyser using GNU Radio (Johann Rittenschober)