Nils Beyrle*

Betreuer:
Prof. Dr. Johannes Maucher
Moritz Seltmann

Ein Projekt im Studiengang Medieninformatik an der Hochschule der Medien Stuttgart

8. August 2010

Zusammenfassung

Das Ziel des Projektes ist es die Steuerungskanäle von Global System for Mobile Communications (GSM) mitzuschneiden. Bei diesen Kanälen handelt es sich um unverschlüsselte Broadcast-Kanäle, die von allen Teilnehmern in einer Zelle zur Koordinierung genutzt werden. Darüber gehen u.A. periodisch ausgestrahlte Informationen der Basisstation, Daten beim Einbuchen eines Endgerätes in ein Netz, oder beim Wechsel (Handover) einer Funkzelle. Zu diesen Kanälen zählt beispielhaft der Broadcast Control Channel (BCCH). Das Ergebnis des Dumps soll sich an dem für WLAN bekannten *airodump* orientieren. Dazu zählt z.B. nach einem Durchlauf eine Übersicht zu gefunden Netzen auszugeben.

Die verschiedenen mitgeschnittenen Pakete müssen dafür zuerst einmal auch als solche auf den PC kommen. Diese werden anschließend dort so aufbereitet, dass sie etwa im weit verbreiteten *Wireshark* betrachtet und näher analysiert werden können.

Das für den Dump eingesetzte Gerät muss dazu die in der Luftschnittstelle vorhandenen Informationen zunächst am Rechner verfügbar machen. Dazu werden im ersten Schritt verschiedene Geräte evaluiert:

- Telit EVK2
- Nokia 3310
- Universal Software Radio Peripheral (USRP)

Aus diesen Geräten wird das am besten geeignete Gerät ausgewählt und mit diesem dann der Mitschnitt und die nötige Aufbereitung der Daten realisiert.

*E-Mail: nb031@hdm-stuttgart.de

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Ein Beispielmitschnitt im Wireshark-Format findet sich unter http://www.gsmdump.de/downloads/arfcn_877_1878.2M.pcap

Die aktuelle Version dieses Dokuments findet sich unter http://www.gsmdump.de/downloads/gsmdump.pdf

Die aktuelle Version dieses Dokuments in einer Druckversion findet sich unter http://www.gsmdump.de/downloads/gsmdump_print.pdf

Die Live-CD zum Projekt findet sich unter http://www.gsmdump.de/downloads/gsmdump.iso

In Beispielpaketen wurden IMSIs oder ähnliche Geräten/Personen zuordenbare Daten im Idealfall anonymisiert.

Im Folgenden werden zumindest grundlegende Kenntnisse über GSM (den Aufbau des Netzes, der Luftschnittstelle und der Signalisierung) vorrausgesetzt. Zum Nachschlagen sei hier auf [9], [7] und [8] verwiesen.

2 Geräteauswahl

2.1 Voraussetzungen

Für die gestellte Aufgabe ist es wichtig, dass das Gerät nicht auf einen bestimmten Anbieter festgelegt ist. Es sollen alle von GSM verwendeten Frequenzen, ohne SIM-Karten von allen Providern zu benötigen, untersucht werden können. Das verwendete Gerät sollte zudem möglichst viele der verfügbaren Daten an seiner Schnittstelle zur Analyse bereitstellen.

2.2 Die Geräte

2.2.1 Telit EVK2

Das *Telit EVK2* besitzt neben einem GSM-Modul auch noch einen GPS-Empfänger. Es gibt seine Daten über vier virtuelle serielle Schnittstellen, welche über eine USB-Schnittstelle an den Rechner angebunden werden, preis. Sein primärer Einsatzzweck ist der mobile Einsatz, um etwa Karten für die Netzabdeckung zu erstellen oder Daten für eine Positionsbestimmung zu gewinnen (wurde bereits im WS 2008/2009 von Christoph Graf im Rahmen des Projektes "GSM und GPS Ortung im Vergleich" an der HdM realisiert).

Gegen eine Verwendung für GSMdump spricht, dass das Gerät eine SIM-Karte benötigt und damit die gewonnen Daten immer an das verwendete Netz geknüpft sind. Es ist also nicht einfach ein Wechsel auf die Frequenzen eines beliebigen Mobilfunkanbieters möglich.

2.2.2 Nokia 3310

Das *Nokia 3310* ist zwar ein schon in die Jahre gekommenes Handy, es hat gegenüber einem iPhone jedoch einen entscheidenden Vorteile: es besitzt einen

Debug-Modus über den es an einer seriellen Schnittstelle sehr viele Informationen über Paktet an sich und sein Netz bereitstellt. So erhält man darüber auch einen Mitschnitt der Übertragenen Daten, jedoch nur aus dem, durch die verwendete SIM-Karte festgelegt, eigenen Netz und an das im speziellen Fall verwendete Handy adressierte Daten.

2.2.3 USRP (Universal Software Radio Peripheral)

Homepage: `http://www.ettus.com/order`

Ein USRP bietet eine sehr flexible Hardware Plattform, die es ermöglicht per Software auf verschiedenen Funk-Frequenzen empfangen und senden zu können. So können für verschiedene Frequenzen verschiedene Empfangs-/Sendemodule in die schwarze Box eingebaut werden und somit theoretisch beliebige Frequenzen mitgehört oder Daten darauf versendet werden.

Diese Offenheit befreit auch von der bei den vorherigen Lösungen nötigen SIM-Karte und von der Beschränkung auf den eigenen Mobilfunkanbieter. Mit dem eingebauten *DBSRX* Modul können Frequenzen von 800 MHz bis 2400 MHz empfangen werden. Das deckt das ganze in Deutschland für GSM (bei 900 und 1800 MHz) verwendete Frequenzspektrum ab.

Bisher ist es mit *airprobe* nur möglich den Downlink-Kanal[1] (also alles was von der Basisstation zum Mobilteil gesendet wird) zu empfangen. Der Grund hierfür ist, dass die Software sich im ersten Schritt auf die Downlink-Frequenz synchronisieren muss, da dort die dafür benötigten Pakete vorhanden sind. Hierfür gibt es nun zwei Lösungsansätze (siehe [4, 5]):

- Im Anschluss an eine erfolgreiche Synchronisierung wäre ein Wechsel auf die Uplink-Frequenz nötig, allerdings ohne die Synchronisation zu verlieren. Dies wäre praktisch fast unmöglich, da ein sofortiges umschalten nicht möglich ist und auch die Dauer eines Umschaltvorgangs nicht im Vorhinein exakt bestimmt werden kann
- Mit einem zweiten Empfangsmodul im USRP wäre es möglich gleichzeitig auf einer Frequenz den Downlink-Kanal und auf einer zweiten den Uplink-Kanal mitschneiden zu können und somit die Synchronisierung aus dem Downlink-Kanal auf den Uplink-Kanal zu übertragen. Diese Funktion ist bisher allerdings noch nicht in *airprobe* implementiert.

Für die Realisierung der Aufgabenstellung wird im folgenden das USRP verwendet.

Ausblick: Mit dem verwendeten Modul wären sogar noch Versuche im 2400 MHz Bereich von WLAN denkbar.

Mit einem weiteren Modul, um auf den GSM-Frequenzen zusätzlich auch senden zu können, wäre mit *OpenBTS*[2] damit der Betrieb einer eigenen Basisstaion möglich.

[1] Siehe auch Abbildung 6 auf Seite 24 eines Paging Request
[2] Für Details zu *OpenBTS* siehe Kapitel 5.2 auf Seite 20

3 Verwendung des USRP

3.1 Überblick über die verwendete Software

Für den Betrieb des USRP sind eine Reihe von verschiedenen Softwarepaketen nötig:

- *GNU Radio:* Eine Software-Sammlung, die sich um den Funklayer kümmert, also alles was irgendwie nötig ist um am Ende rein binäre Daten vorliegen zu haben. Sie spricht das USRP an und bereitet die Daten soweit auf, dass sie von anderen Anwendungen aus *airprobe* weiterverarbeitet werden können. *GNU Radio* implementiert große Teile der Funk-Logik in Software und fällt damit in die Kategorie *Software Defined Radio*[3]
- *airprobe:* Eine Sammlung verschiedener Tools um GSM analysieren und aufbereiten zu können. *airprobe* verarbeitet die Daten von *GNU Radio* weiter.
- *Wireshark:* Das weitverbreitete Werkzeug um Paketmitschnitte komfortabel analysieren zu können. Ist in einer angepassten Version nötig um die GSM Daten zu verstehen.

Abbildung 1 stellt das Zusammenwirken der verschiedenen Tools dar.

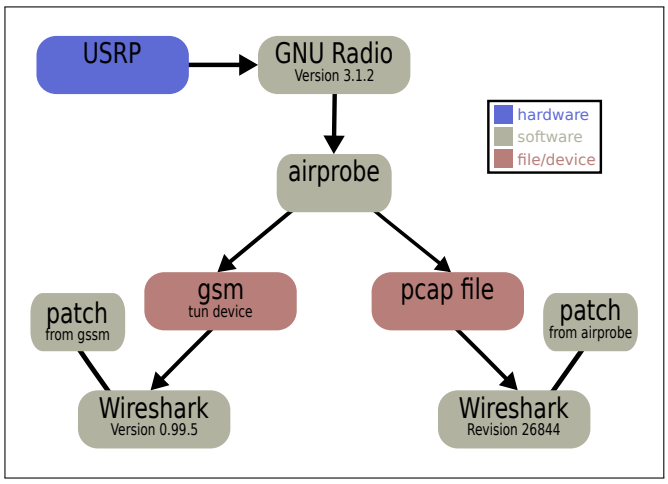

Abbildung 1: Zusammenwirken der einzelnen Programme

3.2 Versionschaos zwischen den Programmen

Die Inbetriebnahme des USRP stellt sich als nicht trivial heraus. Ein großes Problem bis es überhaupt irgend ein Paket zu sehen gibt ist das Versionschaos der verschiedenen verwendeten Programme. So funktionieren benötigte Pachtes nur bei bestimmten Versionen und Programme spielen nur mit bestimmten Versionen anderer Programme zusammen. Alte Versionen machen dann aber

[3] zu *Software Defined Radio* siehe auch http://en.wikipedia.org/wiki/Software-defined_radio

teilweise wieder Probleme mit neueren *GCC* Versionen (für Details siehe auch Kapitel 3.3). Abbildung 2 demonstriert in vereinfachter Form die verschiedenen Abhängigkeiten.

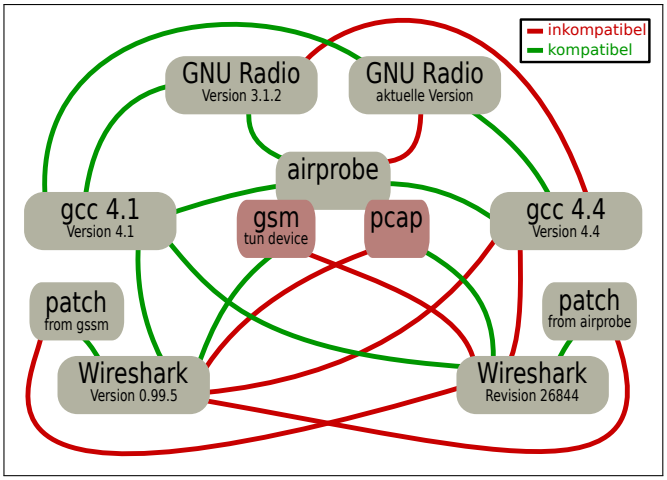

Abbildung 2: Versionschaos zwischen den Programmen

3.3 Installation der Software

Als Entwicklungs- und Test-System kommt ein Ubuntu 9.10 unter 32 Bit zum Einsatz. Es sind einige Programme zum kompilieren von eigener Software nötig. Sollte einer der folgenden Schritte daher fehlschlagen fehlt vermutlich ein benötigtes Programm oder eine beötigte Bibliothek, welche dann bei Bedarf über den Paketmanager nachzuinstallieren ist.

GNU Radio setzt in deren Wiki [4] die folgenden Programme voraus. Es empfiehlt sich diese für die weiteren Schritte zu installieren.

```
sudo aptitude install swig g++ automake libtool python-dev libfftw3-dev libcppunit-dev
     libboost1.38-dev libusb-dev fort77 sdcc sdcc-libraries libsdl1.2-dev python-wxgtk2.8
     subversion git-core guile-1.8-dev libqt4-dev python-numpy ccache python-opengl libgsl0-
     dev python-cheetah python-lxml doxygen qt4-dev-tools libqwt5-qt4-dev libqwtplot3d-qt4-
     dev pyqt4-dev-tools
```

GCC wird zudem in Version 4.1 benötigt:

```
sudo aptitude install gcc-4.1
```

Sollten sich ein paar Python Skripte über fehlende libs beschweren, so liegt das daran, dass Ubuntu nicht mehr in /usr/local/lib/python2.6/site-packages danach sucht, z.B. *GNU Radio* hierhin jedoch installiert wurde. Zwei Symlinks dienen hier als einfache Abhilfe:

```
cd /usr/local/lib/python2.6/dist-packages
sudo ln -s ../site-packages/gnuradio gnuradio
sudo ln -s ../site-packages/usrpm usrpm
```

Anschließend sollten die entsprechenden libs vom System gefunden werden.

[4] http://gnuradio.org/redmine/wiki/gnuradio/UbuntuInstall

3.3.1 GNU Radio

Homepage: http://gnuradio.org/redmine/wiki/gnuradio [2]

Bei *GNU Radio* stellte es sich als sehr schwer heraus die zu *airprobe* passende Version zu finden. Die beiden Programme arbeiten nicht in jeder Version zusammen. Nach unzähligen Kompilier-Vorgängen zeigte sich die Verison 3.1.2 als funktionsfähig. Zu all dem lässt sich die *GNU Radio* Version 3.1.2 nicht mit *GCC* in der Version 4.4 bauen, daher ist das configure-Skript anzuweisen den *GCC* in der Version 4.1 zu verwenden. Installieren lässt es sich dann mit:

```
wget ftp://ftp.gnu.org/gnu/gnuradio/gnuradio-3.1.2.tar.gz
tar xvfz gnuradio-3.1.2.tar.gz
cd gnuradio-3.1.2.tar.gz
./configure CC=gcc-4.1 CXX=g++-4.1
make
sudo make install
```

Wichtig ist, dass in der Auflistung nach dem configure die Unterstützung für das USRP aufgeführt wird (siehe Listing 1). Dies sollte zwar der Fall sein, ansonsten müssen jedoch die entsprechenden Meldungen durchgesehen werden, um herauszufinden an was es scheitert.

Listing 1: Ausgabe von *./configure CC=gcc-4.1 CXX=g++-4.1* bei *GNU Radio*

```
**********************************************************************
The following GNU Radio components have been successfully configured:

config
omnithread
gnuradio-core
usrp
gr-usrp
gr-audio-alsa
gr-audio-oss
gr-atsc
gr-gpio
gr-gsm-fr-vocoder
gr-pager
gr-radar-mono
gr-radio-astronomy
gr-trellis
gr-video-sdl
gr-wxgui
gr-sounder
gr-utils
gnuradio-examples

You my now run the make command to build these components.

**********************************************************************
The following components were skipped either because you asked not
to build them or they didn't pass configuration checks:

gr-audio-jack
gr-audio-osx
gr-audio-portaudio
gr-audio-windows
gr-comedi

These components will not be built.
```

3.3.2 airprobe

Homepage: https://svn.berlin.ccc.de/projects/airprobe/ [1]

airprobe erhält man mit einem

```
git clone git://svn.berlin.ccc.de/airprobe
```

damit lässt sich gsmdecode

```
cd airprobe/gsmdecode
./bootstrap
./configure
make
```

und gsm-tvoid bauen

```
cd ../gsm-tvoid
./bootstrap
./configure
make
```

3.3.3 Wireshark

Homepage: http://wiki.wireshark.org/Development [3]

Wireshark wird in Revison 26844 benötigt, diese erhält man mit

```
svn co -r 26844 http://anonsvn.wireshark.org/wireshark/trunk/ wireshark
```

anschließend wendet man den Patch aus *airprobe* an (der Pfad muss evtl. entsprechend angepasst werden)

```
cd wireshark
patch -p0 < ~/airprobe/wireshark/wireshark-wtap-gsm.patch
```

um zum Abschluß Wiresark zu konfigurieren und zu kompilieren. Hier macht wieder der *GCC* in Verison 4.4 Probleme, weshalb auf die Version 4.1 ausgewichen wird:

```
./autogen.sh
./configure CC=gcc-4.1 CXX=g++-4.1
make
```

3.3.4 airprobe - gssm

Homepage: http://thre.at/gsm/

gssm wird nicht immer zwangsläufig benötigt. Für einen Live-Mitschnitt ist jedoch *Wireshark* in der Version 0.99.5 und *mktun* aus *gssm* nötig (siehe auch Kapitel 3.4.4 auf Seite 12).

Leider kommt *gssm* nicht mit *GNU Radio* Version 3.1.2 klar, hierfür wird die Revision 5220 benötigt. Diese Revision bezieht sich auf SVN, inzwischen verwendet *GNU Radio* jedoch Git, dort lautet die entsprechende Revision 28259329. Zum kompilieren ist hierfür folgende Vorgehensweise notwendig:

```
git clone git://gnuradio.org/gnuradio
cd gnuradio
git checkout 28259329
./configure CC=gcc-4.1 CXX=g++-4.1
make
sudo make install
```

anschließend wird *Wireshark* in Version 0.99.5 benötigt. Hier führen folgende Befehle zum Erfolg:

```
wget http://www.wireshark.org/download/src/all-versions/wireshark-0.99.5.tar.bz2
tar xvfz wireshark-0.99.5.tar.bz2
cd wireshark-0.99.5
patch -p1 < ~/airprobe/gssm/patch/wireshark-0.99.5-gssm.patch
./configure CC=gcc-4.1 CXX=g++-4.1
make
sudo make install
```

zu letzt muss noch *gssm* kompiliert werden

```
cd airprobe/gssm
./bootstrap
./configure
make
```

3.4 Beschreibung von airprobe

3.4.1 gsmdecode

gsmdecode dient dazu die in hexadezimaler Schreibweise empfangenen Pakete in ein menschenlesbares Form zu bringen und auf der Shell auszugeben. Die Daten erhält es von *gsm-tvoid*.

3.4.2 gsm-tvoid

gsm-tvoid empfängt mithilfe von *GNU Radio* die GSM-Daten und leitet sie beispielsweise an *gsmdecode* zur Ausgabe weiter

```
cd airprobe/gsm-tvoid/src/python
./gsm_scan.py -R B -p d -c 24 -r e | ../../../gsmdecode/src/gsmdecode -i
```

- **-R B** weißt *gsm-tvoid* an das Modul B zu verwenden. Das USRP besitzt zwei Einschübe für Sende-/Empfangsmodule. In unserem Fall ist in Steckplatz B das *DBSRX* eingebaut.
- **-p d** gibt die Ausgabeform an. In unserem Fall d, was eine Ausgabe für *gsmdecode* in hexadezimaler Form erzeugt.
- **-c 24** gibt den GSM-ARFCN Kanal an, der mitgeschnitten werden soll.
- **-r e** definiert die Region. Hier e für Europa.

gsm-tvoid stellt in einer Grafik (siehe Abbildung 3 auf der nächsten Seite) zudem die Empfangsstärke der aktuell gewählten Frequenz dar.

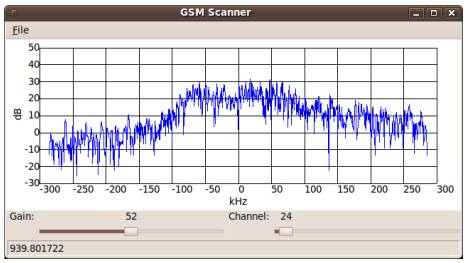

Abbildung 3: Grafische Darstellung der Empfangsstärke bei *gsm-tvoid*

3.4.3 gsm-receiver

gsm-receiver bietet eine ähnliche Funktionsweise wie *gsm-tvoid*. Viele der Funktionen wurden inzwischen auch unter den Programmen ausgetauscht, wodurch kein allzu großer Unterschied mehr zwischen ihnen besteht. In diesem Projekt wird im Allgemeinen *gsm-tvoid* verwendet.

3.4.4 gssm

Mit *gssm* ist es möglich die Daten aus der Luft live in *Wireshark* über ein TUN-Device zu analysieren.

Dafür muss der TUN-Kernel-Treiber geladen und das gsm TUN-Device erstellt werden

```
sudo modprobe tun
cd src/mktun
sudo ./mktun gsm
```

Nachdem in der Datei ../python/gssm_usrp.py der Wert c0 auf die gewünschte Frequenz gesetzt wurde, kann der Mitschnitt gestartet

```
cd ../python
./gssm_usrp.py
```

und in *Wiresahrk* das Interface gsm ausgewählt werden.

Da für *gssm* selbst eine andere *GNU Radio* Version als für *gsm-tvoid* oder *gsm-receiver* benötigt wird, kann von *gssm* auch nur *mktun* verwendet werden um das TUN-Device zu erstellen und *Wireshark* entsprechen gepatcht werden. *gsm-tvoid* kann das damit erstellte TUN-Device dann auch nutzen und ermöglicht auch so ohne *gssm* einen Live-Mitschnitt.

4 Live-CD

4.1 Einleitung

Homepage: http://www.gsmdump.de

Um den Einstieg möglichst einfach zu machen entstand während des Projekts eine Live-CD auf der bereits alle nötigen Programme installiert und eingerichtet sind. Somit muss nur noch diese CD gestartet, das USRP angeschlossen und das

entsprechende Programm gestartet werden.

Die einzelnen im Folgenden beschriebenen Programme und Skripte können über eine Konsole in einem beliebigen Pfad gestartet werden. Die Programme und Skripte sind entsprechend gelinkt.

Zum Einstieg empfiehlt sich zunächst ein Blick auf *gsmdump.sh* (siehe Kapitel 4.3.1 auf der nächsten Seite), welches einen guten Überblick über die gesamten Möglichkeiten bietet. Damit ergibt sich auch recht einfach ein erster Überblick über die empfangenen Mobilfunkstationen in der Nachbarschaft.

Die Live-CD basiert auf Ubuntu 10.04 (i386). An diesem wurden einige Änderungen vorgenommen, so wurden zahlreiche nicht direkt benötigte Pakete entfernt um Platz zu sparen und die CD unter der 700 MB Grenze einer realen CD zu halten. Die benötigte Software wurde auf einem Ubuntu 9.10 kompiliert und zu Debian-Paketen verpackt, welche dann in dem Live-System installiert wurden.

Die aktuellste Version der CD findet sich unter `http://www.gsmdump.de/downloads/gsmdump.iso`

4.2 Programme

4.2.1 GNU Radio

GNU Radio[5] ist in Version 3.1.2 enthalten.

4.2.2 airprobe

Auf der Live-CD sind alle Programmteile von *airprobe*[6] enthalten. Diese befinden sich im Verzeichnis ~/Desktop/airprobe

4.2.3 Wireshark

Wireshark[7] ist in Revison 26844 auf der Live-CD installiert. Diese Version dient dem Betrachten von zuvor aufgezeichneten pcap Dateien. Abbildung 4 auf der nächsten Seite zeigt *Wireshark* mit einer zur näheren Analyse geöffneten pcap Datei.

Weiterhin ist *Wireshark*[8] auch in Version 0.99.5 mit einem Patch von *gssm* auf der Live-CD vorhanden. Dieser wird für Live-Mitschnitte benötigt und kann als *gsmshark* gestartet werden.

Erläuterungen zu den einzelnen Paketen finden sich in Anhang C auf Seite 25.

[5] Installation von *GNU Radio* siehe Kapitel 3.3.1 auf Seite 9
[6] Installation von *airprobe* siehe Kapitel 3.4 auf Seite 11
[7] Installation von *Wireshark* in Revison 26844 siehe Kapitel 3.3.3 auf Seite 10
[8] Installation von *Wireshark* in Version 0.99.5 siehe Kapitel 3.3.4 auf Seite 11

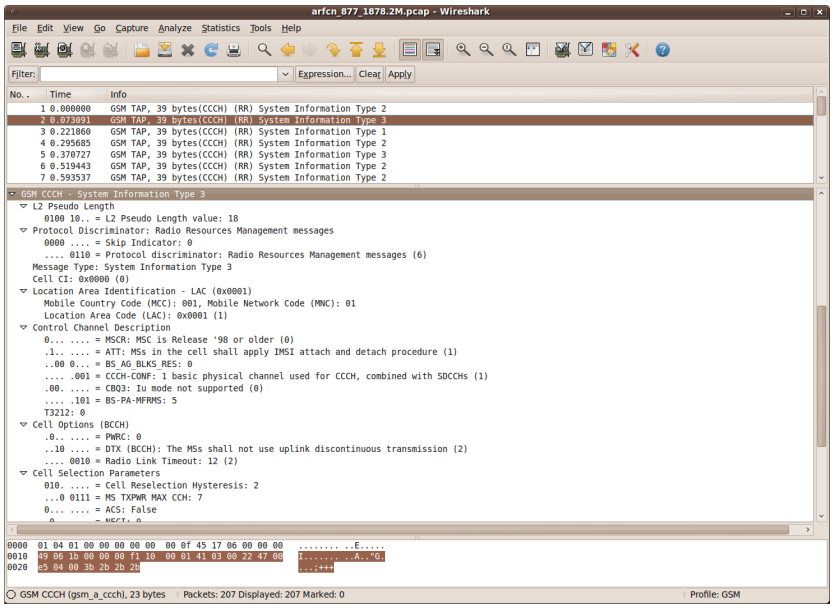

Abbildung 4: pcap Datei in *Wireshark*

4.2.4 GSMdump

Im Ordner ~/Desktop/gsmdump befinden sich Skripte um einen einfachen Einstieg zu ermöglichen. Die Programme befinden sich alle im Pfad, können also direkt in einer Shell an jedem Ort auch ohne explizite Pfadangabe gestartet werden. Die einzelnen Programme werden in Kapitel 4.3 näher beschreiben.

4.3 Beschreibung der GSMdump Skripte

4.3.1 gsmdump.sh

Quellcode siehe Anhang D.1 auf Seite 36

gsmdump.sh erstellt eine Übersicht über alle in der Umgebung empfangbaren Basisstaionen. Für den Einstieg lohnt sich zunächst ein Blick auf dieses Skript.

Dazu hört es auf allen Frequenzen nacheinander für eine definierte Zeit auf Daten und erstellt daraus Dateien im pcap Format. Diese wandelt *pcap4gsmdecode.py*[9] daraufhin in ein hexadezimales Format um, welches *gsmdecode*[10] interpretieren kann. Die Ausgabe von *gsmdecode* wird von *analyse.py*[11] schließlich analysiert und in CSV Dateien abgelegt. Im letzten Schritt erstellt *summary.py*[12] aus diesen CSV Dateien lesbare Text Versionen, welche mit den pcap Dateien zu der entsprechenden Frequenz (und Absolute Radio Frequency Channel Number (ARFCN)) in einem nach dem aktuellen Datum benannten Ordner abgelegt werden.

[9] *pcap4gsmdecode.py* siehe Kapitel 4.3.2 auf Seite 16
[10] *gsmdecode* siehe Kapitel 3.4.1 auf Seite 11
[11] *analyse.py* siehe Kapitel 4.3.3 auf Seite 17
[12] *summary.py* siehe Kapitel 4.3.4 auf Seite 17

Hilfe zur Verwendung von *gsmdump.sh* findet sich in Listing 2.

Listing 2: Verwendung von gsmdump.sh

```
$ gsmdump.sh -h
This script captures GSM traffic and creates a summary and pcap file. By default it iterates
      all ARFCNs.

SYNOPSIS
      gsmdump.sh [ -s TMPDIR ] [ -t SECONDS ] [ -d DECIM ] [ -g GAIN ] [ -c ARFCN ] [ -R
           BOARD ] [ -o DSTDIR ] [ -k ] [ -l ] [ -h ]

OPTIONS
      -s TMPDIR
          Sets the TMPDIR.
      -t SECONDS
          Duration in SECONDS to stay on each ARFCN.
      -d DECIM
          Sets the DECIM value.
      -g GAIN
          Sets the GAIN value.
      -c ARFCN
          Scan only on ARFCN, don't iterate all.
      -R BOARD
          Specifies which BOARD (A or B) of the USRP to use.
      -o DSTDIR
          Sets the DSTDIR. Only if -l is not set.
      -k
          Keeps the raw dump files. Be carfull with this option, this can be much data.
      -l
          Run in an infinite loop. Not in combination with -o.
      -h
          Prints this help.
```

Abbildung 5 auf der nächsten Seite zeigt eine beispielhafte Ausgabe eines Scandurchgangs beim Aufruf von *gsmdump.sh*. Ohne explizite Parameter scannt das Skript dabei alle ARFCNs nacheinander auf vorhandene Basisstaionen. Dabei sind im Ergebnis Zellen aller vier großen deutschen Mobilfunkanbieter zu sehen. Die empfangenen IMSIs wurden in der Abbildung geschwärzt, empfangene TMSIs wurden aus Platzgründen nicht aufgelistet.

Bei Aufruf dieses Skripts wird ein Ordner mit dem aktuellen Datum erstellt (z.B. 20100725204351). Darin werden die pcap Datein (z.B. arfcn_24_939.8M.pcap) und die zugehörige Zusammenfassung in einer TXT Datei (z.B. arfcn_24_939.8M.txt) abgelegt. Zudem wird in diesem Ordner noch eine Zusammenfassung von allen ARFCNs erstellt (summary.txt).

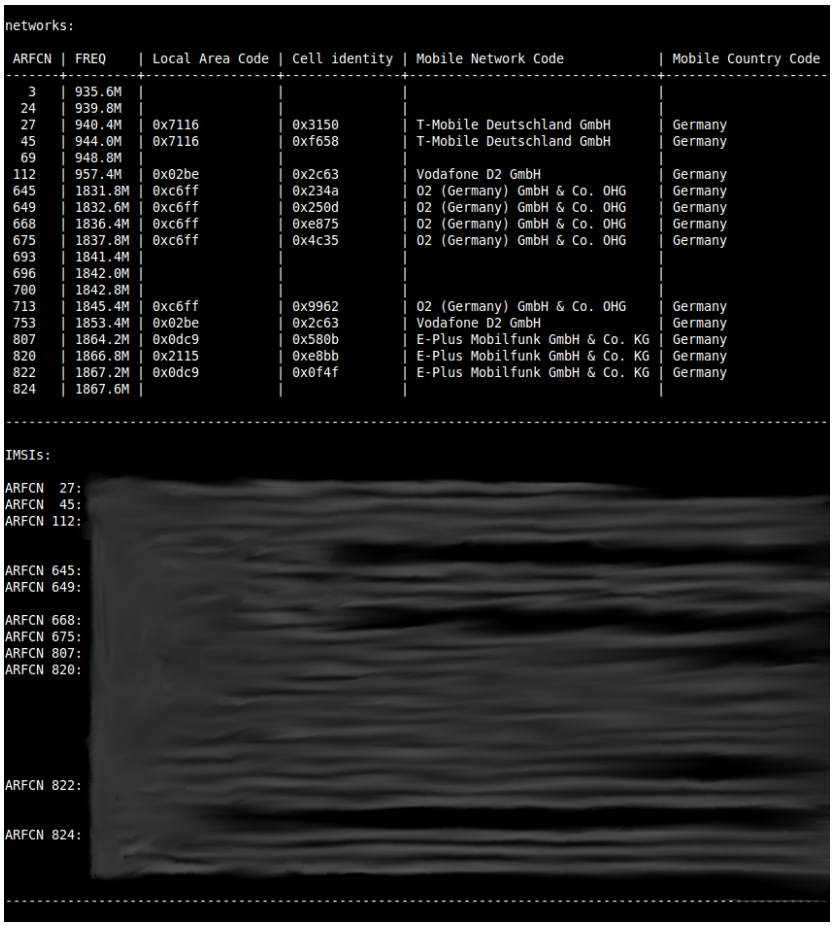

Abbildung 5: Beispielausgabe eines Scans von *gsmdump.sh*

4.3.2 pcap4gsmdecode.py

Quellcode siehe Anhang D.2 auf Seite 41

pcap4gsmdecode.py wandelt eine Datei im pcap Format in ein von *gsmdecode*[13] lesbares Format um.

Dazu wird es mit der entsprechenden pcap Datei als Parameter aufgerufen und gibt diese in einem hexadezimalen Format aus. Diese Ausgabe kann dann mit *gsmdecode* weiterverarbeitet werden.

Ein Beispiel zur Verwendung von *pcap4gsmdecode.py* findet sich in Listing 3.

Listing 3: Verwendung von pcap4gsmdecode.py

```
$ pcap4gsmdecode.py 20100725204351/arfcn_24_939.8M.pcap | gsmdecode -i
```

[13] *gsmdecode* siehe Kapitel 3.4.1 auf Seite 11

4.3.3 analyse.py

Quellcode siehe Anhang D.3 auf Seite 42

analyse.py erstellt aus der Ausagbe von *gsmdecode*[14] eine Zusammenfassung über in der Nachbarschaft gefundene Netze, IMSIs und TMSIs im CSV Format.

Hilfe zur Verwendung von *analyse.py* findet sich in Listing 4.

Listing 4: Verwendung von analyse.py

```
$ analyse.py -h
Usage: analyse.py [options]

Options:
  -h, --help show this help message and exit
  -a ARFCN, --arfcn=ARFCN
                        Input is an ARFCN
```

Die CSV Datei kann mit *summary.py*[15] in ein für die Bildschirmausgabe optimiertes Format gebracht werden.

4.3.4 summary.py

Quellcode siehe Anhang D.4 auf Seite 44

summary.py erstellt aus den CSV Dateien von *analyse.py*[16] eine menschenlesbare Ausgabe.

Hilfe zur Verwendung von *summary.py* findet sich in Listing 5.

Listing 5: Verwendung von summary.py

```
$ summary.py -h
Usage: summary.py [options]

Options:
  -h, --help show this help message and exit
  -s, --single Read from a single ARFCN file
```

[14] *gsmdecode* siehe Kapitel 3.4.1 auf Seite 11
[15] *summary.py* siehe Kapitel 4.3.4
[16] *analyse.py* siehe Kapitel 4.3.3

4.3.5 arfcn.py

Quellcode siehe Anhang D.5 auf Seite 48

arfcn.py berechnet zu einer gegebenen ARFCN die zugehörige Frequenz, welche z.B. für die Zusammenarbeit mit *GNU Radio* benötigt wird.

Hilfe zur Verwendung von *arfcn.py* findet sich in Listing 6.

Listing 6: Verwendung von arfcn.py

```
$ arfcn.py -h
Usage: arfcn.py [options]

Options:
  -h, --help show this help message and exit
  -a ARFCN, --arfcn=ARFCN
                        Input is an ARFCN
```

Eine Beispielausgabe von *arfcn.py* findet sich in Listing 7.

Listing 7: Beispielausgabe von *arfcn.py -a 24*

```
$ arfcn.py -a 24
939.8M
```

4.3.6 gsmlive.sh

Quellcode siehe Anhang D.6 auf Seite 49

gsmlive.sh ermöglicht einen Live-Mitschnitt mit Wireshark. Das Programm öffnet automatisch Wireshark und startet den Mitschnitt.

Beim booten der Live-CD wird mit *mktun* aus *gssm*[17] über

```
mktun gsm
```

das tun Device gsm erstellt, über welches dann die Daten live an Wireshark geleitet werden können.

Für den Live-Mitschnitt wird Wireshark (mit einem Patch von *gssm*) in Version 0.99.5 benötigt. Dieser ist auf der Live-CD zusätzlich als *gsmshark* verfügbar.

Hilfe zur Verwendung von *gsmlive.sh* findet sich in Listing 8 auf der nächsten Seite.

[17] *gssm* siehe Kapitel 3.4.4 auf Seite 12

Listing 8: Verwendung von gsmlive.sh

```
$ gsmlive.sh -h
This script captures an specified ARFCN. It creates a live dump to the gsm tun device and a
     pcap file for later analysis.

SYNOPSIS
      gsmlive.sh [ -c ARFCN ] [ -d DECIM ] [ -R BOARD ] [ -s ] [ -h ]

OPTIONS
      -c ARFCN
          Sets the ARFCN.
      -d DECIM
          Sets the DECIM value.
      -R BOARD
          Specifies which BOARD (A or B) of the USRP to use.
      -s
          Don't start Wireshark. So there will be an output to the shell, the pcap file and
                the gsm tun device. You can capture the gsm device with any other tool of
               course.
      -h
          Prints this help.
```

4.3.7 capture.sh

Quellcode siehe Anhang D.7 auf Seite 51

capture.sh schreibt die Rohaten von einer bestimmten ARFCN oder, sollte keine explizite Angabe erfolgen, von allen ARFCNs in Dateien. Diese können im nächsten Schritt manuell mit *analysecfile.sh*[18] ausgewertet werden.

Hilfe zur Verwendung von *capture.sh* findet sich in Listing 9.

Listing 9: Verwendung von capture.sh

```
$ capture.sh -h
This script captures GSM traffic and stores it in a cfile. By default it iterates all ARFCNs
     .

SYNOPSIS
      capture.sh [ -t SECONDS ] [ -d DECIM ] [ -g GAIN ] [ -c ARFCN ] [ -R BOARD ] [ -d
           DSTDIR ] [ -h ]

OPTIONS
      -t SECONDS
          Duration in SECONDS to stay on each ARFCN.
      -d DECIM
          Sets the DECIM value.
      -g GAIN
          Sets the GAIN value.
      -c ARFCN
          Scan only on ARFCN, don't iterate all.
      -R BOARD
          Specifies which BOARD (A or B) of the USRP to use.
      -d DSTDIR
          Sets the DSTDIR.
      -h
          Prints this help.
```

[18] *analysecfile.sh* siehe Kapitel 4.3.8 auf der nächsten Seite

4.3.8 analysecfile.sh

Quellcode siehe Anhang D.8 auf Seite 53

analysecfile.sh wertet mitgeschnittene Rohdaten analog zu *gsmdump.sh*[19] aus.

Es eignet sich dazu zuvor mitgeschnittene Rohdaten, z.B. von *capture.sh*[20], auszuwerten.

Hilfe zur Verwendung von *analysecfile.sh* findet sich in Listing 10.

Listing 10: Verwendung von analysecfile.sh

```
$ analysecfile.sh -h
This script creates a summary and pcap file. It reads from a captured cfile.

SYNOPSIS
      analysecfile.sh [ -s TMPDIR ] [ -d DECIM ] [ -h ] CFILE

OPTIONS
      -s TMPDIR
          Sets the TMPDIR.
      -d DECIM
          Sets the DECIM value.
      -h
          Prints this help.
```

5 Weitere Software-Projekte

5.1 OpenBSC

Homepage: http://openbsc.osmocom.org/trac/

OpenBSC ist eine offene Implementierung eines Base Station Controller (BSC). Es kann im aktuellen Entwicklungsstand zwei verschiedene Base Transceiver Station (BTS), die *Siemens BS-11 microBTS* und die *ip.access nanoBTS*, ansprechen. Netzseitig implementiert *OpenBSC* zudem in Software noch das Mobile Switching Center (MSC), Home Location Register (HLR), Authentication Center (AuC), Visitor Location Register (VLR) und Equipment Identity Register (EIR).

Das Projekt möchte eine Plattform schaffen um mit GSM Netzseitig experimentieren zu können und etwa die Sicherheit von GSM zu analysieren. Es möchte nicht GSM bis ins letzte Detail implementieren oder eine produktiv nutzbare Lösung schaffen.

5.2 OpenBTS

Homepage: http://openbts.sourceforge.net/

[19] *gsmdump.sh* siehe Kapitel 4.3.1 auf Seite 14
[20] *capture.sh* siehe Kapitel 4.3.7 auf der vorherigen Seite

OpenBTS bietet eine Schnittstelle zwischen GSM und der Software-Telefonanlage Asterisk. Es verwendet mit *GNU Radio* und einem USRP auch Hardwareseitig eine offene und sehr flexible Plattform. Das Projekt möchte im Vergleich zu *OpenBSC* mehr eine günstige Lösung für GSM-Netze bieten als eine möglichst Standardkonforme Lösung zum experimentieren mit GSM sein. So lagert es z.B. das HLR und MSC komplett an Asterisk aus.

5.3 OsmocomBB

Homepage: http://bb.osmocom.org/trac/

OsmocomBB möchte eine komplett freie Implementierung eines Treibers für den Chip und den zugehörigen GSM-Stack für ein Telefon realisieren. Das Projekt konzentriert sich dabei auf Telefone mit Ti Calypso/Iota/Rita Chipsätzen, z.B. dem MotorolaC123. Im Gegensatz zu *OpenBSC* und *OpenBTS* ist die Arbeit hier auf die Mobilseite gerichtet.

6 Fazit

Mit *GSMdump* ist es möglich den BCCH, Paging channel (PCH) und Access grant channel (AGCH) von GSM mitzuschneiden und anschließend zu analysieren. Dabei handelt es sich um Informationen einer Basisstation an ein bestimmtes Handy (z.B. über einen aus dem Netz kommenden anstehenden Anruf) oder auch an alle empfangenden Handys im Umkreis (z.B. über die Eigenschaften und Fähigkeiten der betreffenden Zelle, oder in der Nachbarschaft vorhandene weitere alternative Zellen). Es ist kein Werkzeug um unerlaubt fremde Telefongespräche mithören zu können, dazu sind Momentan auch noch einige technische Schwierigkeiten zu bewältigen, man darf aber gespannt sein was sich in dieser Richtung in Zukunft noch so alles Bewegt.

Die während des Projekts enstandenen Skripte vereinfachen die Nutzung der verwendeten Programme wie *airprobe*, da sie sich im Hintergrund um die Zusammenarbeit der einzelnen beteiligten Tools kümmern. Die Daten werden dabei entsprechend umgewandelt und aufbereitet, um sie anschließend komfortabel analysieren zu können.

Durch die entstandene Live-CD ist es möglich ohne viel Vorarbeit einen GSM Mitschnitt zu realisieren, ohne das dafür Änderungen auf einem eigenen System nötig sind oder zuerst Software kompiliert und installiert werden muss und die ganzen zuvor schon im Detail erwähnten Besonderheiten und Stolpersteine dabei berücksichtigt werden müssen. Die für Mitschnitt und Analyse benötigten Programme sind dazu bereits auf der Live-CD installiert und eingerichtet. Zudem sind bereits alle nötigen Schritte am System vorgenommen um direkt loslegen zu können.

An dieser Stelle vielen Dank an meine Betreuer Moritz Seltmann und Prof. Dr. Johannes Maucher für deren Unterstützung.

A Abkürzungsverzeichnis

AGCH Access grant channel

ARFCN Absolute Radio Frequency Channel Number

AuC Authentication Center

BCCH Broadcast Control Channel

BSC Base Station Controller

BTS Base Transceiver Station

CCCH Common Control Channel

EIR Equipment Identity Register

GPRS General Packet Radio Service

GSM Global System for Mobile Communications

HLR Home Location Register

HSN Hopping Sequence Number

IMSI International Mobile Subscriber Identity

LAC Location Area Code

MAIO Mobile Allocation Index Offset

MNC Mobile Network Code

MOC Mobile Country Code

MS Mobile Station

MSC Mobile Switching Center

NCC National Color Code/Network Color Code

PCH Paging channel

RACH Random Access Channel

SACCH Slow associated control channel

SDCCH Stand alone dedicated control channel

SIM Subscriber Identity Module

TMSI Temporary Mobile Subscriber Identity

USRP Universal Software Radio Peripheral

VLR Visitor Location Register

B Sequenzdiagramm

B.1 Paging Request

Abbildung 6 zeigt einen Paging Request für einen vom Mobilfunknetz kommenden Anruf. Pakete in Downlink-Richtung können mit dem USRP empfangen werden, in der Abbildung sind diese zusätzlich mit einem * gekennzeichnet.

Die Grafik wurde mit Hilfe des Nokia 3310[21] erstellt, da dort auch die Pakete vom Telefon in Uplink-Richtung zur BTS sichtbar sind.

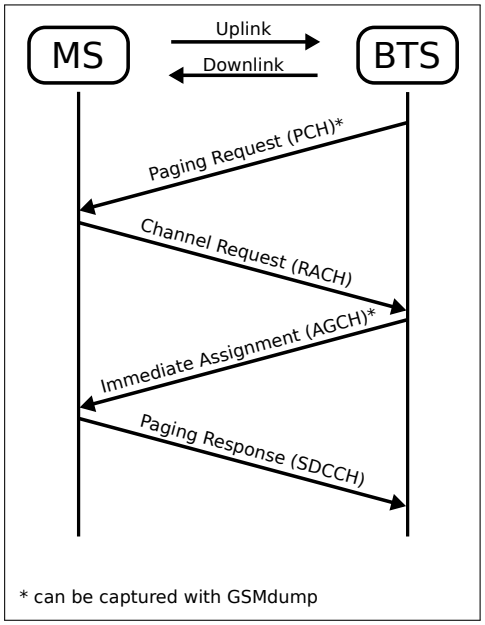

Abbildung 6: Sequenzdiagramm eines Paging Request

[21] siehe Kapitel 2.2.2 auf Seite 5

C Analysierte Mitschnitte

Im folgenden werden nun einige empfangene Pakete und deren Details näher erläutert.

C.1 BCCH

C.1.1 System Information Type 1

Zellweiter Broadcast vier mal in der Sekunde auf dem BCCH.

```
1  GSM CCCH - System Information Type 1
2      L2 Pseudo Length
3          0101 01.. = L2 Pseudo Length value: 21
4      Protocol Discriminator: Radio Resources Management messages
5          0000 .... = Skip Indicator: 0
6          .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7      Message Type: System Information Type 1
8      00.. 000. = Format Identifier: bit map 0 (0x00)
9      List of ARFCNs = 83 24
10     RACH Control Parameters
11         10.. .... = Max retrans: Maximum 4 retransmissions (2)
12         ..10 01.. = Tx-integer: 12 slots used to spread transmission (9)
13         .... ..0. = CELL_BARR_ACCESS: The cell is not barred (0)
14         .... ...1 = RE: True
15         0000 0000 0000 0000 = ACC: 0x0000
16     SI 1 Rest Octets
17         NCH position: not present
18         Band Indicator: 1800
```

- **Zeile 9:** Traffic Channel auf ARFCN 83
- **Zeile 9:** Kanalnummer des BCCH auf ARFCN 24
- **Zeile 10ff:** RACH Control Prameters
  - **Zeile 11:** max. Wiederholungen des Zugriffsversuchs: 4
  - **Zeile 12:** zugewiesene Zufallszahl zwischen 3 und 50 für Warte-Zeitschlitze: 12
  - **Zeile 13:** erlaubte Nutzerklassen: 0 (alle)
  - **Zeile 14:** Verbindungaufbau nach Abriss in selber Zelle erlaubt: Ja

Siehe auch [7, Seite 46ff]

C.1.2 System Information Type 2

Zellweiter Broadcast auf dem BCCH der die Frequenzen der Nachbarzellen des Providers (der selbe National Color Code/Network Color Code (NCC)) enthält.

```
1  GSM CCCH - System Information Type 2
2      L2 Pseudo Length
3          0101 10.. = L2 Pseudo Length value: 22
4      Protocol Discriminator: Radio Resources Management messages
5          0000 .... = Skip Indicator: 0
6          .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7      Message Type: System Information Type 2
8      BCCH Frequency List
9      ..0. .... = EXT-IND: The information element carries the complete BA (0)
10     ...0 .... = BA-IND: 0
11     00.. 000. = Format Identifier: bit map 0 (0x00)
12     List of ARFCNs = 102 92 90 45 33 29 27 24
13     NCC Permitted
14         0000 1000 = NCC Permitted: 0x08
15     RACH Control Parameters
16         10.. .... = Max retrans: Maximum 4 retransmissions (2)
17         ..10 01.. = Tx-integer: 12 slots used to spread transmission (9)
18         .... ..0. = CELL_BARR_ACCESS: The cell is not barred (0)
19         .... ...1 = RE: True
20         0000 0000 0000 0000 = ACC: 0x0000
```

- **Zeile 12:** Liste benachbarter ARFCNs: 102 92 90 45 33 29 27 24
- **Zeile 15ff:** RACH analog zu System Information Type 1, siehe Kapitel C.1.1 auf der vorherigen Seite

Siehe auch [7, Seite 48]

C.1.3 System Information Type 2ter

Zellweiter Broadcast auf dem BCCH, der Informationen zu den Nachbarzellen enthält.

```
GSM CCCH - System Information Type 2ter
    L2 Pseudo Length
        0000 00.. = L2 Pseudo Length value: 0
    Protocol Discriminator: Radio Resources Management messages
        0000 .... = Skip Indicator: 0
        .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
    Message Type: System Information Type 2ter
    Extended BCCH Frequency List
    .00. .... = Multiband Reporting: 0
    ...0 .... = BA-IND: 0
    10.. 111. = Format Identifier: variable bit map (0x47)
    List of ARFCNs = 596
    SI 2ter Rest Octets
        Empty
```

Siehe auch [6, Seite 295f]

C.1.4 System Information Type 2quater

Zellweiter Broadcast auf dem BCCH, der Informationen zu den Nachbarzellen enthält.

```
GSM CCCH - System Information Type 2quater
    L2 Pseudo Length
        0000 00.. = L2 Pseudo Length value: 0
    Protocol Discriminator: Radio Resources Management messages
        0000 .... = Skip Indicator: 0
        .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
    Message Type: System Information Type 2quater
    SI 2quater Rest Octets
        0... .... = BA-IND: 0
        .0.. .... = 3G BA-IND: 0
        ..1. .... = Measurement Parameter Change Mark: 1
        ...0 000. = SI2quater Index: 0
        .... ...0 000. .... = SI2quater Count: 0
        3G Neighbour Cell Description
            UTRAN FDD Description
                .101 0100 1010 100. = FDD UARFCN: 10836
                FDD Indic0: 0
                Nr of FDD Cells : 31
                UTRAN FDD Description
                    Field is 0 bits long
        3G Measurement Parameters Description
            0111 .... = Qsearch I: Always (7)
            .... 0... = QSearch C Initial: use Qsearch I
            .... ..00 00.. .... = FDD Qoffset: always select a cell if acceptable (0)
            ..0. .... = FDD Rep Quant: RSCP
            ...0 0... = FDD Multirat Reporting: 0
            .... .101 = FDD Qmin: -10 dB (5)
        GPRS 3G Measurement Parameters Description
            ..01 11.. = Qsearch P: Always (7)
            .... ..1. = 3G Search Prio: 3G cells may be searched when BSIC decoding is
                 required
        3G Additional Measurement Parameters Description
            .... .000 = FDD Qmin Offset: 0 dB (0)
            0111 .... = FDD RSCPmin: -100 dBm (7)
```

C.1.5 System Information Type 3

Zellweiter Broadcast auf dem BCCH vier mal in der Sekunde.

```
1  GSM CCCH - System Information Type 3
2      L2 Pseudo Length
3          0100 10.. = L2 Pseudo Length value: 18
4      Protocol Discriminator: Radio Resources Management messages
5          0000 .... = Skip Indicator: 0
6          .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7      Message Type: System Information Type 3
8      Cell CI: 0xf657 (63063)
9      Location Area Identification - LAC (0x7116)
10         Mobile Country Code (MCC): 262, Mobile Network Code (MNC): 01
11         Location Area Code (LAC): 0x7116 (28950)
12     Control Channel Description
13         1... .... = MSCR: MSC is Release '99 onwards (1)
14         .1.. .... = ATT: MSs in the cell shall apply IMSI attach and detach procedure (1)
15         ..01 1... = BS_AG_BLKS_RES: 3
16         .... .000 = CCCH-CONF: 1 basic physical channel used for CCCH, not combined with
               SDCCHs (0)
17         .00. .... = CBQ3: Iu mode not supported (0)
18         .... .110 = BS-PA-MFRMS: 6
19         T3212: 60
20     Cell Options (BCCH)
21         .1.. .... = PWRC: 1
22         ..01 .... = DTX (BCCH): The MSs shall use uplink discontinuous transmission (1)
23         .... 0101 = Radio Link Timeout: 24 (5)
24     Cell Selection Parameters
25         011. .... = Cell Reselection Hysteresis: 3
26         ...0 0101 = MS TXPWR MAX CCH: 5
27         0... .... = ACS: False
28         .0.. .... = NECI: 0
29         ..00 0100 = RXLEV-ACCESS-MIN: -107 <= x < -106 dBm (4)
30     RACH Control Parameters
31         10.. .... = Max retrans: Maximum 4 retransmissions (2)
32         ..10 01.. = Tx-integer: 12 slots used to spread transmission (9)
33         .... ..0. = CELL_BARR_ACCESS: The cell is not barred (0)
34         .... ...1 = RE: True
35         0000 0000 0000 0000 = ACC: 0x0000
36     SI 3 Rest Octets
37         SYSTEM INFORMATION TYPE 2ter message is available
38         Early Classmark Sending is allowed
39         GPRS Indicator
40             RA Colour: 6
41             .0.. .... = SI13 Position: SYSTEM INFORMATION TYPE 13 message is sent on BCCH
                   Norm (0)
42         3G Early Classmark Sending Restriction: Neither UTRAN, CDMA2000 nor GERAN IU MODE
               CLASSMARK CHANGE message shall be sent with the Early classmark sending
43         .... 0... = SI2quater Position: SYSTEM INFORMATION TYPE 2 quater message is sent on
               BCCH Norm
```

- **Zeile 8:** Die vom Provider für diese Zelle vergebene ID: 0xf657
- **Zeile 10:** Der Mobile Country Code (MOC) definiert das Land, hier 262 für Deutschland, der Mobile Network Code (MNC) den Provider, hier 01 für T-Mobile
- **Zeile 11:** Der Location Area Code (LAC) definiert die Location in welcher sich die Basissation befindet: 0x7116
- **Zeile 14:** Mobiltelefone sollen sich in der Zelle an-/abmelden (ATT)
- **Zeile 15:** Anzahl der verwendeten Paging Kanäle für AGCH (BS_AG_BLKS_RES)
- **Zeile 16:** Konfiguration des Common Control Channel (CCCH) (CCCH-CONF): im CCCH werden keine Zeitschlitze für einen Stand alone dedicated control channel (SDCCH) reserviert

- **Zeile 18:** Zugewiesene Paging-Untergruppe: 6 Multiframe-Perioden (BS-PA-MFRMS)
- **Zeile 19:** Zeit bis zur Aktuallisierung des Standorts: 60 Stunden
- **Zeile 21:** Leistung des MS wird geregelt (PWRC)
- **Zeile 22:** Diskontinuierliche Übertragung im Uplink, während Gesprächspausen Strom sparen und nicht senden
- **Zeile 23:** Verbindung abbrechen nach 24 nicht dekodierbaren Slow associated control channel (SACCH) Rahmen (Radio Link Timeout)
- **Zeile 25:** Beharrungsfaktor um an Zellgrenzen nicht hin und her zu wechseln (Cell Reselection Hysteresis): 3 dBm
- **Zeile 26:** max. Sendeleistung die ein Mobile auf Random Access Channel (RACH) ausstrahlen darf (MS TXPWR MAX CCH). Wert in dBm.
- **Zeile 29:** nötiger Pegel für das Mobile um auf BTS zuzugreifen (RXLEV-ACCESS-MIN): -107 dBm
- **Zeile 30ff:** RACH analog zu System Information Type 1, siehe Kapitel C.1.1 auf Seite 25
- **Zeile 36ff:** Informationen zur General Packet Radio Service (GPRS) Fähigkeit der Zelle (SI 3 Rest Octets)

Siehe auch [7, Seite 49ff]

C.1.6 System Information Type 4

```
GSM CCCH - System Information Type 4
    L2 Pseudo Length
        0011 00.. = L2 Pseudo Length value: 12
    Protocol Discriminator: Radio Resources Management messages
        0000 .... = Skip Indicator: 0
        .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
    Message Type: System Information Type 4
    Location Area Identification - LAC (0x7116)
        Mobile Country Code (MCC): 262, Mobile Network Code (MNC): 01
        Location Area Code (LAC): 0x7116 (28950)
    Cell Selection Parameters
        011. .... = Cell Reselection Hysteresis: 3
        ...0 0101 = MS TXPWR MAX CCH: 5
        0... .... = ACS: False
        .0.. .... = NECI: 0
        ..00 0100 = RXLEV-ACCESS-MIN: -107 <= x < -106 dBm (4)
    RACH Control Parameters
        10.. .... = Max retrans: Maximum 4 retransmissions (2)
        ..10 01.. = Tx-integer: 12 slots used to spread transmission (9)
        .... ..0. = CELL_BARR_ACCESS: The cell is not barred (0)
        .... ...1 = RE: True
        0000 0000 0000 0000 = ACC: 0x0000
    SI 4 Rest Octets
        SI4 Rest Octets_O
            GPRS Indicator
                RA Colour: 6
                .... ..0. = SI13 Position: SYSTEM INFORMATION TYPE 13 message is sent on BCCH
                     Norm (0)
        Break Indicator: Additional parameters "SI4 Rest Octets_S" are not sent in SYSTEM
             INFORMATION TYPE 7 and 8
```

Analog zu System Information Type 3, siehe Kapitel C.1.5 auf Seite 29

Siehe auch [7, Seite 54f]

C.1.7 System Information Type 13

Broadcast zu GPRS Fähigkeiten der BTS

```
GSM CCCH - System Information Type 13
    L2 Pseudo Length
        0000 00.. = L2 Pseudo Length value: 0
    Protocol Discriminator: Radio Resources Management messages
        0000 .... = Skip Indicator: 0
        .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
    Message Type: System Information Type 13
    SI 13 Rest Octets
        .100 .... = BCCH Change Mark: 4
        .... 0000 = SI Change Field: Update of unspecified SI message or SI messages (0)
        .01. .... = SI13 Change Mark: 1
        GPRS Mobile Allocation
            ...0 0000 1... .... = HSN: 1
            ...0 0000 1... .... = MA Length: 1
            MA Bitmap: 11
        .... 0000 0001 .... = RAC: 1
        .... 0... = SPGC CCCH Sup: SPLIT_PG_CYCLE is not supported on CCCH in this cell
        .... .110 = Priority Access Thr: Packet access is allowed for priority level 1 to 4
             (6)
        00.. .... = Network Control Order: NC0 (0)
        GPRS Cell Options
            ..00 .... = NMO: Network Mode of Operation I (0)
            .... 011. = T3168: 2000 ms (3)
            .... ...1 11.. .... = T3192: 200 ms (7)
            ..01 1... = DRX Timer Max: 4 s (3)
            .... .0.. = Access Burst Type: 8-bit format shall be used
            .... ..0. = Control Ack Type: Default format is four access bursts
            .... ...1 001. .... = BS CV Max: 9
            .... 001. = PAN Dec: 1
            .... ...0 10.. .... = PAN Inc: 2
            ..00 0... = PAN Max: maximum value allowed for counter N3102 is 4 (0)
            GPRS Cell Options Extension Information
                Extension Length: 14
                .... .0.. = EGPRS Packet Channel Request: Use EGPRS PACKET CHANNEL REQUEST
                     message for uplink TBF establishment on the PRACH
                .... ..01 10.. .... = BEP Period: 10 (6)
                ..0. .... = PFC Feature Mode: The network does not support packet flow context
                      procedures
                ...0 .... = DTM Support: The cell does not support DTM procedures
                .... 0... = BSS Paging Coordination: The cell does not support Circuit-
                     Switched paging coordination
                .... .1.. = CCN Active: CCN is enabled in the cell
                .... ..1. = NW Ext UTBF: The extended uplink TBF mode is supported by the
                     network
                .... ...0 = Multiple TBF Capability: The cell does not support multiple TBF
                     procedures
                1... .... = Ext UTBF No Data: The mobile station may refrain from sending a
                     PACKET UPLINK DUMMY CONTROL BLOCK message when there is no other RLC/MAC
                     block ready to send in an uplink radio block allocated by the network
                .0.. .... = DTM Enhancements Capability: The cell does not support enhanced
                     DTM CS establishment and enhanced DTM CS release procedures
        GPRS Power Control Parameters
            .000 0... = Alpha: 0.0 (0)
            .... .001 01.. .... = T Avg W: 2^(5/2) / 6 multiframes (5)
            ..00 011. = T Avg T: 2^(3/2) / 6 multiframes (3)
            .... ...0 = PC Meas Chan: Downlink measurements for power control shall be made
                 on BCCH
            0001 .... = N Avg I: 2^(1/2) (1)
```

C.2 CCCH

C.2.1 Immediate Assignment

Die BTS weist dem Mobile einen SDCCH Kanal zu, nachdem dieses auf dem RACH einen Channel Request gestellt hat. Den kompletten Ablauf zeigt Abbildung 6 auf Seite 24.

```
1  GSM CCCH - Immediate Assignment
2      L2 Pseudo Length
3          0011 00.. = L2 Pseudo Length value: 12
4      Protocol Discriminator: Radio Resources Management messages
5          0000 .... = Skip Indicator: 0
6          .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7      Message Type: Immediate Assignment
8      Page Mode
9          .... ..11 = Page Mode: Same as before (3)
10     Dedicated mode or TBF
11         .000 .... = Dedicated mode or TBF: This message assigns a dedicated mode resource (0)
12     Channel Description
13         0100 0... = SDCCH/8 + SACCH/C8 or CBCH (SDCCH/8), Subchannel 1
14         .... .001 = Timeslot: 1
15         010. .... = Training Sequence: 2
16         ...1 .... = Hopping channel: Yes
17         Hopping channel: MAIO 0
18         Hopping channel: HSN 1
19     Request Reference
20         Random Access Information (RA): 226
21         0010 1... = T1': 5
22         .... .010 101. .... = T3: 21
23         ...0 1001 = T2: 9
24         [RFN: 7263]
25     Timing advance value: 2
26     Mobile Allocation
27         Length: 1
28         Bitmap of increasing ARFCNs included in the Mobile Allocation: 11000000
29     IA Rest Octets
30         Data(Not decoded)
```

- **Zeile 11:** fest zugwiesener Kanal, nicht wie GPRS nur zeitweilig (Dedicated mode or TBF)
- **Zeile 13:** Kanalkonfiguration (Channel Description): SDCCH/8 + SACCH/C8 or CBCH (SDCCH/8), Subchannel 1
- **Zeile 14:** Zeitschlitz: 1
- **Zeile 16:** Frequenzsprungverfahren wird verwendet (Hopping channel): Yes
- **Zeile 17f:** Reihenfolge der nacheinander verwendeten Frequenzen und Zeitschlitze (Mobile Allocation Index Offset (MAIO)/Hopping Sequence Number (HSN))
  - **Zeile 17:** Zahl max. zur Verfügung stehender Frequenzen (MAIO)
  - **Zeile 18:** HSN zwischen 0-63 für Frequenzsprunggenerator: 1
- **Zeile 20:** aktuelle Framenummer der BTS wird dem Mobile mitgeteilt (Random Access Information)
- **Zeile 25:** Entferungsabhäniger Korrekturfaktor für Paketlaufzeit (Timing advance value): 2

Siehe auch [7, Seite 58ff]

C.3 PCH

C.3.1 Paging Request Type 1 (IMSI)

Anruf aus dem Mobilfunknetz an das Mobile. Den kompletten Ablauf zeigt Abbildung 6 auf Seite 24.
Type 1 kann gleichzeitig bis zu zwei Teilnehmer rufen. Kann aber auch leer sein.

```
1  GSM CCCH - Paging Request Type 1
2      L2 Pseudo Length
3          0011 00.. = L2 Pseudo Length value: 12
4      Protocol Discriminator: Radio Resources Management messages
5          0000 .... = Skip Indicator: 0
6          .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7      Message Type: Paging Request Type 1
8      Page Mode
9          .... ..00 = Page Mode: Normal paging (0)
10     Channel Needed
11         ..00 .... = Channel 1: Any channel (0)
12         00.. .... = Channel 2: Any channel (0)
13     Mobile Identity - Mobile Identity 1 - IMSI (262010123456789)
14         Length: 8
15         0010 .... : Identity Digit 1: 2
16         .... 1... = Odd/even indication: Odd number of identity digits (1)
17         .... .001 = Mobile Identity Type: IMSI (1)
18         BCD Digits: 262010123456789
19     P1 Rest Octets
20         Packet Page Indication 1: Paging procedure for RR connection establishment
21         Packet Page Indication 2: Paging procedure for RR connection establishment
```

- **Zeile 17:** Das Gerät wird anhand seiner IMSI gerufen
- **Zeile 18:** Die IMSI: 262 steht ist die Länderkennung für Deutschland, 01 der Mobilfunkanbieter (T-Mobile), der Rest bildet die Teilnehmernummer

Siehe auch [7, Seite 45]

C.3.2 Paging Request Type 1 (TMSI)

Anruf aus dem Mobilfunknetz an das Mobile. Den kompletten Ablauf zeigt Abbildung 6 auf Seite 24.

```
1  GSM CCCH - Paging Request Type 1
2      L2 Pseudo Length
3          0010 01.. = L2 Pseudo Length value: 9
4      Protocol Discriminator: Radio Resources Management messages
5          0000 .... = Skip Indicator: 0
6          .... 0110 = Protocol discriminator: Radio Resources Management messages (6)
7      Message Type: Paging Request Type 1
8      Page Mode
9          .... ..00 = Page Mode: Normal paging (0)
10     Channel Needed
11         ..00 .... = Channel 1: Any channel (0)
12         00.. .... = Channel 2: Any channel (0)
13     Mobile Identity - Mobile Identity 1 - TMSI/P-TMSI (0x12345678)
14         Length: 5
15         1111 .... : Unused
16         .... 0... = Odd/even indication: Even number of identity digits (0)
17         .... .100 = Mobile Identity Type: TMSI/P-TMSI (4)
18         TMSI/P-TMSI: 0x12345678
19     P1 Rest Octets
20         Packet Page Indication 1: Paging procedure for RR connection establishment
21         Packet Page Indication 2: Paging procedure for RR connection establishment
```

- **Zeile 17:** Das Gerät wird anhand seiner TMSI gerufen
- **Zeile 18:** Die gerufenen TMSI

Siehe auch [7, Seite 45]

D Quellcode der Live-CD

D.1 gsmdump.sh

```
#!/bin/bash
#
# This script captures GSM traffic and creates a summary and pcap file. By default it
    iterates all ARFCNs.
#

# predefine config values
TMP=/tmp/gsmdump
DURATION=4
DECIM=112
GAIN=52
ARFCN=""
BOARD="B"
AKTPWD=`pwd`
DST=$AKTPWD"/"`date +%Y%m%d%H%M%S`
KEEP=1
LOOP=1
DSTSET=1
ARFCNSET=1

# check for parameters and change config values
while getopts ":s:t:d:g:c:R:o:klh" optname; do
      case "$optname" in
            "s")
                  TMP=$OPTARG
            ;;
            "t")
                  DURATION=$OPTARG
            ;;
            "d")
                  DECIM=$OPTARG
            ;;
            "g")
                  GAIN=$OPTARG
            ;;
            "c")
                  ARFCN=$OPTARG
                  ARFCNSET=0
            ;;
            "R")
                  BOARD=$OPTARG
            ;;
            "o")
                  DST=$OPTARG
                  DSTSET=0
            ;;
            "k")
                  KEEP=0
            ;;
            "l")
                  LOOP=0
            ;;
            "h")
                  echo "This script captures GSM traffic and creates a summary and pcap
                      file. By default it iterates all ARFCNs."
                  echo
                  echo "SYNOPSIS"
                  echo " gsmdump.sh [ -s TMPDIR ] [ -t SECONDS ] [ -d DECIM ] [ -g GAIN ]
                       [ -c ARFCN ] [ -R BOARD ] [ -o DSTDIR ] [ -k ] [ -l ] [ -h ]"
                  echo
                  echo "OPTIONS"
                  echo " -s TMPDIR"
                  echo " Sets the TMPDIR."
                  echo " -t SECONDS"
                  echo " Duration in SECONDS to stay on each ARFCN."
                  echo " -d DECIM"
                  echo " Sets the DECIM value."
                  echo " -g GAIN"
```

```
66                      echo " Sets the GAIN value."
67                      echo " -c ARFCN"
68                      echo " Scan only on ARFCN, don't iterate all."
69                      echo " -R BOARD"
70                      echo " Specifies which BOARD (A or B) of the USRP to use."
71                      echo " -o DSTDIR"
72                      echo " Sets the DSTDIR. Only if -l is not set."
73                      echo " -k"
74                      echo " Keeps the raw dump files. Be carfull with this option, this can
                              be much data."
75                      echo " -l"
76                      echo " Run in an infinite loop. Not in combination with -o."
77                      echo " -h"
78                      echo " Prints this help."
79                      exit 0
80              ;;
81              "?")
82                      echo "Unknown option $OPTARG"
83                      echo
84                      $0 -h
85                      exit 1
86              ;;
87              ":")
88                      echo "No argument value for option $OPTARG"
89                      echo
90                      $0 -h
91                      exit 1
92              ;;
93              *)
94                      echo "Unknown error while processing options"
95                      echo
96                      $0 -h
97                      exit 1
98              ;;
99      esac
100 done
101
102 # check parameter
103 if [ $DSTSET -eq 0 ] && [ $LOOP -eq 0 ]; then
104         echo "ERROR: -o and -l set at the same time."
105         echo
106         $0 -h
107         exit 1
108 fi
109
110 # DST is relative
111 if ! [ `echo -n $DST | grep -e "^/.*"` ]; then
112         DST=$AKTPWD"/"$DST
113 fi
114
115 # check or create the destination
116 if ! [ -d $DST ]; then
117         mkdir -p $DST
118 fi
119
120 # check or create the tmp
121 if ! [ -d $TMP ]; then
122         mkdir -p $TMP
123 fi
124 cd $TMP
125
126 # Look for usrp_rx_cfile.py
127 USRP_PROG=usrp_rx_cfile.py
128 while :; do
129         which "$USRP_PROG" > /dev/null
130         if [ $? -eq 0 ]; then
131                 break
132         fi
133         USRP_PROG=/usr/share/gnuradio/usrp/usrp_rx_cfile.py
134         which "$USRP_PROG" > /dev/null
135         if [ $? -eq 0 ]; then
136                 break
137         fi
138
```

```
139         echo "ERROR: usrp_rx_cfile.py not found. Make sure it's in your PATH!"
140         exit 1
141 done
142
143 # calculate samples for the specified duration
144 SAMPLES=`expr 64000000 / $DECIM '*' $DURATION`
145
146 # rm old netlists
147 if [ -f netlist.nets ]; then
148         rm netlist.nets
149 fi
150 if [ -f netlist.imsis ]; then
151         rm netlist.imsis
152 fi
153 if [ -f netlist.tmsis ]; then
154         rm netlist.tmsis
155 fi
156
157 # loop forever?
158 LOOPING=0
159 while [ $LOOPING -eq 0 ]; do
160
161         # no single ARFCN specified
162         if [ $ARFCNSET -eq 1 ]; then
163
164                 # Now iterate all, about 1000 RFCNS, each for -t seconds
165                 # GSM 900 (ARFCN 1 - 124) && GSM 1800 (ARFCN 512 885)
166                 for ARFCN in `seq 1 124 && seq 512 885`; do
167
168                         FREQ=$(arfcn.py -a $ARFCN)
169                         FILE="$TMP/arfcn_${ARFCN}_${FREQ}.cfile"
170                         echo "capturing for $DURATION seconds ($SAMPLES samples) on module
                                $BOARD on ARFCN $ARFCN (${FREQ}Hz) to $FILE"
171                         $USRP_PROG -R $BOARD -g $GAIN -d "$DECIM" -f "$FREQ" -N $SAMPLES $FILE
                                > /dev/null 2> /dev/null
172                         if [ $? -gt 0 ]; then
173                                 echo "WARNING: something went wrong while capturing (ARFCN
                                        $ARFCN), continuing on the next ARFCN anyway ..."
174                         fi
175                         gsm_scan_light.py -SN -pd -d "$DECIM" -I "$FILE" > /dev/null 2> /dev/
                                null
176                         if [ -e tvoid.pcap ] && [ -e tvoid-burst.pcap ]; then
177                                 if [ `stat -c %s tvoid.pcap` -gt 24 ]; then
178                                         mv tvoid.pcap $DST/arfcn_${ARFCN}_$FREQ.pcap
179                                         mv tvoid-burst.pcap $DST/arfcn_${ARFCN}_$FREQ-burst.pcap
180                                         pcap4gsmdecode.py $DST/arfcn_${ARFCN}_$FREQ.pcap |
                                                gsmdecode -i | analyse.py -a $ARFCN
181                                         # check if there are any networks
182                                         if [ -f netlist.single.nets ]; then
183                                                 summary.py -s > $DST/arfcn_${ARFCN}_$FREQ.txt
184                                                 rm netlist.single.nets
185                                         fi
186                                 fi
187                         fi
188
189                         if [ $KEEP -eq 0 ]; then
190                                 mv $FILE $DST/.
191                         else
192                                 rm $FILE
193                         fi
194
195                 done
196
197         # capture only a single ARFCN
198         else
199
200                 FREQ=$(arfcn.py -a $ARFCN)
201                 FILE="$TMP/arfcn_${ARFCN}_${FREQ}.cfile"
202                 echo "capturing for $DURATION seconds ($SAMPLES samples) on module $BOARD on
                        ARFCN $ARFCN (${FREQ}Hz) to $FILE"
203                 $USRP_PROG -R $BOARD -g $GAIN -d "$DECIM" -f "$FREQ" -N $SAMPLES $FILE > /dev/
                        null 2> /dev/null
204                 if [ $? -gt 0 ]; then
205                         echo "WARNING: something went wrong while capturing (ARFCN $ARFCN) ..."
```

```
206             fi
207             gsm_scan_light.py -SN -pd -d "$DECIM" -I "$FILE" > /dev/null 2> /dev/null
208             if [ -e tvoid.pcap ] && [ -e tvoid-burst.pcap ]; then
209                     if [ `stat -c %s tvoid.pcap` -gt 24 ]; then
210                             mv tvoid.pcap $DST/arfcn_${ARFCN}_$FREQ.pcap
211                             mv tvoid-burst.pcap $DST/arfcn_${ARFCN}_$FREQ-burst.pcap
212                             pcap4gsmdecode.py $DST/arfcn_${ARFCN}_$FREQ.pcap | gsmdecode -i
                                        | analyse.py -a $ARFCN
213                             # check if there are any networks
214                             if [ -f netlist.single.nets ]; then
215                                     summary.py -s > $DST/arfcn_${ARFCN}_$FREQ.txt
216                                     rm netlist.single.nets
217                             fi
218                     fi
219             fi
220
221             if [ $KEEP -eq 0 ]; then
222                     mv $FILE $DST/.
223             else
224                     rm $FILE
225             fi
226
227     fi
228
229     # create a complete summary, if something was found
230     if [ -f netlist.nets ]; then
231             summary.py | tee $DST/summary.txt
232     else
233             echo
234             echo "I'm sorry, but no network was found"
235             echo
236     fi
237
238     # rm old netlists
239     if [ -f netlist.nets ]; then
240             rm netlist.nets
241     fi
242     if [ -f netlist.imsis ]; then
243             rm netlist.imsis
244     fi
245     if [ -f netlist.tmsis ]; then
246             rm netlist.tmsis
247     fi
248     if [ -f netlist.single.nets ]; then
249             rm netlist.single.nets
250     fi
251     if [ -f netlist.single.imsis ]; then
252             rm netlist.single.imsis
253     fi
254     if [ -f netlist.single.tmsis ]; then
255             rm netlist.single.tmsis
256     fi
257     if [ -f gsm-receiver.pcap ]; then
258             rm gsm-receiver.pcap
259     fi
260     if [ -f gsm-receiver-burst.pcap ]; then
261             rm gsm-receiver-burst.pcap
262     fi
263     if [ -f speech.gsm ]; then
264             rm speech.gsm
265     fi
266     if [ -f cfile2.out ]; then
267             rm cfile2.out
268     fi
269
270     # check if loop forever?
271     if [ $LOOP -eq 0 ]; then
272             # set new DST
273             DST=$AKTPWD"/"`date +%Y%m%d%H%M%S`
274             # check or create the destination
275             if ! [ -d $DST ]; then
276                     mkdir -p $DST
277             fi
278     else
```

```
279                 # no loop
280                 LOOPING=1
281         fi
282
283 done
284
285 exit 0
```

D.2 pcap4gsmdecode.py

```
#! /usr/bin/env python
#
# This script converts a pcap file so gsmdecode can read it.
#

import sys
import pcap

def print_packet(pktlen, data, timestamp):
        # check if len(data) is < 17, cause the next for starts at position 16
        if not data or len(data) < 17:
                return

        out = ""
        for i in range(16, len(data)-1):
                if len(hex(ord(data[i]))[2:]) == 1:
                        # convert the data in HEX, and add zero before
                        out = out + " 0" + str(hex(ord(data[i]))[2:])
                else:
                        # same, but no zero is needed
                        out = out + " " + str(hex(ord(data[i]))[2:])

        print out

def main():

        # help the user
        if len(sys.argv) < 2:
                print 'usage: pcap4gsmdecode.py <file.pcap>'
                sys.exit(1)

        # create a pcap Object
        p = pcap.pcapObject()
        # and use it for the given file
        p.open_offline(sys.argv[1])

        pkt = (1,2)
        pkt = p.next()
        # iterate the packages, if there are more packages, print them
        while type(pkt) is tuple:
                apply(print_packet,pkt)
                pkt = p.next()

if __name__=='__main__':
        main()
```

D.3 analyse.py

```
#! /usr/bin/env python
#
# This script analyses the output of gsmdecode for a summary and writes found networks,
      IMSIs and TMSIs into a CSVs.
#

import sys
import re
from optparse import OptionParser

def parse_options():
       parser = OptionParser()
       parser.add_option("-a", "--arfcn", type="int", dest="arfcn",
                                                  help="Input is an ARFCN")

       (options, args) = parser.parse_args()
       if (len(args) != 0):
              parser.print_help()
              sys.exit(1)

       return options

def freq(arfcn):

       # GSM 900
       if arfcn > 0 and arfcn <= 124:
              freq = 890 + 0.2*arfcn + 45

       # GSM 1800
       elif arfcn >= 512 and arfcn <= 885:
              freq = 1710.2 + 0.2*(arfcn - 512) + 95

       else:
              freq = 0

       return freq

def main():

       # get parameter
       options = parse_options()

       # check options
       # nothing to check

       input=sys.stdin.readlines()
       nets = []
       imsis = []
       tmsis = []
       file_nets = ""
       file_imsis = ""
       file_tmsis = ""

       for i in range(0, len(input)):
              input[i] = re.sub("\n","",input[i])
       for i in range(0, len(input)):
              if re.search("Cell identity",input[i]):
                     nets = input[i:i+4]
              if re.search("Type of identity: IMSI",input[i]):
                     imsis.append(re.sub(r".*ID\(7/odd\): (.*)$",r"\1",input[i+1]))
              if re.search("Type of identity: TMSI",input[i]):
                     tmsis.append(re.sub(r".*ID\(4/even\): (.*)$",r"\1",input[i+1]))
       if nets == []:
              if not (options.arfcn):
                     file_nets = "?" + ";" + "?" + ";;;;"
              else:
                     file_nets = str(options.arfcn) + ";" + str(freq(options.arfcn)) + "M" +
                            ";;;;"
       else:
              if not (options.arfcn):
```

```
69                      file_nets = "?" + ";" + "?" + ";" + re.sub(r".*\[([^]]*)\] Cell
                             identity.*",r"\1",nets[0]) + ";" + re.sub(r".*: [^ ]* ([^ ]*) *
                             Mobile Country Code .*$",r"\1",nets[1]) + ";" + re.sub(r".*Mobile
                              Country Code \((.*)\)$",r"\1",nets[1]) + ";" + re.sub(r".*: [^
                             ]* ([^ ]*) *Mobile Network Code .*$",r"\1",nets[2]) + ";" + re.
                             sub(r".*Mobile Network Code \((.*)\)$",r"\1",nets[2]) + ";" + re.
                             sub(r".*\[([^]]*)\] Local Area Code.*",r"\1",nets[3])
70              else:
71                      file_nets = str(options.arfcn) + ";" + str(freq(options.arfcn)) + "M" +
                              ";" + re.sub(r".*\[([^]]*)\] Cell identity.*",r"\1",nets[0]) + "
                             ;" + re.sub(r".*: [^ ]* ([^ ]*) *Mobile Country Code .*$",r"\1",
                             nets[1]) + ";" + re.sub(r".*Mobile Country Code \((.*)\)$",r"\1",
                             nets[1]) + ";" + re.sub(r".*: [^ ]* ([^ ]*) *Mobile Network Code
                             .*$",r"\1",nets[2]) + ";" + re.sub(r".*Mobile Network Code \((.*)
                             \)$",r"\1",nets[2]) + ";" + re.sub(r".*\[([^]]*)\] Local Area
                             Code.*",r"\1",nets[3])
72
73      # collect IMSIs
74      for i in range(0, len(imsis)):
75              file_imsis = file_imsis + str(imsis[i])
76              if i < len(imsis) - 1:
77                      file_imsis = file_imsis + ";"
78
79      # collect TMSIs
80      for i in range(0, len(tmsis)):
81              file_tmsis = file_tmsis + str(tmsis[i])
82              if i < len(tmsis) - 1:
83                      file_tmsis = file_tmsis + ";"
84
85      # wirte nets to file
86      file = open("netlist.nets", "a")
87      file.write(file_nets + "\n")
88      file.close()
89      file = open("netlist.single.nets", "w")
90      file.write(file_nets + "\n")
91      file.close()
92
93      # wirte IMSIs to file
94      file = open("netlist.imsis", "a")
95      file.write(file_imsis + "\n")
96      file.close()
97      file = open("netlist.single.imsis", "w")
98      file.write(file_imsis + "\n")
99      file.close()
100
101     # wirte TMSISs to file
102     file = open("netlist.tmsis", "a")
103     file.write(file_tmsis + "\n")
104     file.close()
105     file = open("netlist.single.tmsis", "w")
106     file.write(file_tmsis + "\n")
107     file.close()
108
109 if __name__=='__main__':
110     main()
```

D.4 summary.py

```
#! /usr/bin/env python
#
# This script produces a fancy summary output, it's maybe sometimes a little bit dirty, but the output looks OK.
# The script looks for the width of your shell and modifys the output, so it fits the screen.
#
# The getTerminalSize() function is from http://stackoverflow.com/questions/566746/how-to-get-console-window-width-in-python
#

import sys
import re
from optparse import OptionParser

def getTerminalSize():
	def ioctl_GWINSZ(fd):
		try:
			import fcntl, termios, struct, os
			cr = struct.unpack('hh', fcntl.ioctl(fd, termios.TIOCGWINSZ,'1234'))
		except:
			return None
		return cr
	cr = ioctl_GWINSZ(0) or ioctl_GWINSZ(1) or ioctl_GWINSZ(2)
	if not cr:
		try:
			fd = os.open(os.ctermid(), os.O_RDONLY)
			cr = ioctl_GWINSZ(fd)
			os.close(fd)
		except:
			pass
	if not cr:
		try:
			cr = (env['LINES'], env['COLUMNS'])
		except:
			cr = (25, 80)
	return int(cr[1]), int(cr[0])

def single_true():
	return

def parse_options():
	parser = OptionParser()
	parser.add_option("-s", "--single", action="store_true", dest="single", default=False,
                                                help="Read from a single ARFCN file")

	(options, args) = parser.parse_args()
	if (len(args) != 0):
		parser.print_help()
		sys.exit(1)

	return options

def main():

	# get parameter
	options = parse_options()

	arfcns = []

	# is the input a single ARFCN or the complete list?
	if options.single:
		# read netlist from file
		try:
			file = open("netlist.single.nets", "r")
		except IOError:
			print("ERROR: the netlist.single.nets file could not be found")
			sys.exit(1)
		liste = file.read()
		file.close()
```

```
68                liste = liste.split("\n")
69                liste.pop(len(liste) - 1)
70                for i in range(0, len(liste)):
71                        liste[i] = liste[i].split(";")
72
73                # read IMSISs from file
74                try:
75                        file = open("netlist.single.imsis", "r")
76                except IOError:
77                        print("ERROR: the netlist.single.imsis file could not be found")
78                        sys.exit(1)
79                imsis = file.read()
80                file.close()
81                imsis = imsis.split("\n")
82                imsis.pop(len(imsis) - 1)
83                for i in range(0, len(imsis)):
84                        imsis[i] = imsis[i].split(";")
85
86                # read TMSISs from file
87                try:
88                        file = open("netlist.single.tmsis", "r")
89                except IOError:
90                        print("ERROR: the netlist.single.tmsis file could not be found")
91                        sys.exit(1)
92                tmsis = file.read()
93                file.close()
94                tmsis = tmsis.split("\n")
95                tmsis.pop(len(tmsis) - 1)
96                for i in range(0, len(tmsis)):
97                        tmsis[i] = tmsis[i].split(";")
98        else:
99                # read netlist from file
100               try:
101                       file = open("netlist.nets", "r")
102               except IOError:
103                       print("ERROR: the netlist.nets file could not be found")
104                       sys.exit(1)
105               liste = file.read()
106               file.close()
107               liste = liste.split("\n")
108               liste.pop(len(liste) - 1)
109               for i in range(0, len(liste)):
110                       liste[i] = liste[i].split(";")
111
112               # read IMSISs from file
113               try:
114                       file = open("netlist.imsis", "r")
115               except IOError:
116                       print("ERROR: the netlist.imsis file could not be found")
117                       sys.exit(1)
118               imsis = file.read()
119               file.close()
120               imsis = imsis.split("\n")
121               imsis.pop(len(imsis) - 1)
122               for i in range(0, len(imsis)):
123                       imsis[i] = imsis[i].split(";")
124
125               # read TMSISs from file
126               try:
127                       file = open("netlist.tmsis", "r")
128               except IOError:
129                       print("ERROR: the netlist.tmsis file could not be found")
130                       sys.exit(1)
131               tmsis = file.read()
132               file.close()
133               tmsis = tmsis.split("\n")
134               tmsis.pop(len(tmsis) - 1)
135               for i in range(0, len(tmsis)):
136                       tmsis[i] = tmsis[i].split(";")
137
138       # set widths of the rows
139       width0 = max([len(row[0]) for row in liste])
140       width1 = max([len(row[1]) for row in liste])
141       width2 = max([len(row[2]) for row in liste])
```

```
142         width3 = max([len(row[3]) for row in liste])
143         width4 = max([len(row[4]) for row in liste])
144         width5 = max([len(row[5]) for row in liste])
145         width6 = max([len(row[6]) for row in liste])
146         width7 = max([len(row[7]) for row in liste])
147 
148         if width0 < 5:
149                 width_0 = 5
150         else:
151                 width_0 = width0
152         if width1 < 4:
153                 width_1 = 4
154         else:
155                 width_1 = width1
156         if width2 < 13:
157                 width_3 = 13
158         else:
159                 width_3 = width2
160         if (width3 + width4 + 3) < 19:
161                 width_5 = 19
162         else:
163                 width_5 = (width3 + width4 + 3)
164         if (width5 + width6 + 3) < 19:
165                 width_4 = 19
166         else:
167                 width_4 = (width5 + width6 + 3)
168         if width7 < 15:
169                 width_2 = 15
170         else:
171                 width_2 = width7
172 
173         # network overview
174         sys.stdout.write("\n")
175         print "networks:"
176         sys.stdout.write("\n")
177         print " ARFCN" + " " * (width_0 - 5) + " | FREQ" + " " * (width_1 - 4) + " | Local
                Area Code" + " " * (width_2 - 15) + " | Cell identity" + " " * (width_3 - 13) +
                " | Mobile Network Code" + " " * (width_4 - 19) + " | Mobile Country Code"
178         print "-" * (width_0 + 2) + "+" + "-" * (width_1 + 2) + "+" + "-" * (width_2 + 2) + "
                +" + "-" * (width_3 + 2) + "+" + "-" * (width_4 + 2) + "+" + "-" * (width_5 + 2)
179         for row in liste:
180                 print " " + " " * (3 - len(row[0])) + row[0] + " " * (width_0 - len(row[0]) -
                        (3 - len(row[0]))) + " | " + row[1] + " " * (width_1 - len(row[1])) + " |
                         " + row[7] + " " * (width_2 - len(row[7])) + " | " + row[2] + " " * (
                        width_3 - len(row[2])) + " | " + row[5] + " (" + row[6] + ")" + " " * (
                        width_4 - len(row[5]) - len(row[6]) - 3) + " | " + row[3] + " (" + row[4]
                         + ")"
181                 arfcns.append(row[0])
182 
183         # create fancy output for found IMSIs ...
184         sys.stdout.write("\n")
185         sys.stdout.write("-" * (getTerminalSize()[0])); sys.stdout.write("\n")
186         sys.stdout.write("\n")
187         print("IMSIs:")
188         sys.stdout.write("\n")
189         arfcn_c = 0
190         for row in imsis:
191                 if not row == ['']:
192                         sys.stdout.write("ARFCN " + " " * (3 - len(arfcns[arfcn_c])) + arfcns[
                                arfcn_c] + ": ")
193                         width_c = (getTerminalSize()[0]-11)/16
194                         for i in range(0, len(row)):
195                                 sys.stdout.write(row[i])
196                                 if not i == len(row) - 1:
197                                         sys.stdout.write(" ")
198                                 if (i + 1) % width_c == 0:
199                                         # problem with complete full lines
200                                         if not (getTerminalSize()[0] - 11) % 16 == 0:
201                                                 # last item?
202                                                 if not i == len(row) - 1:
203                                                         sys.stdout.write("\n")
204                                         if not i == len(row) - 1:
205                                                 sys.stdout.write(" " * 11)
206                         if not i == len(row) - 1 or not (getTerminalSize()[0] - 11) % 16 == 0:
```

```
207                             sys.stdout.write("\n")
208                 arfcn_c = arfcn_c + 1
209
210         # ... and of course TIMSIs
211         sys.stdout.write("\n")
212         sys.stdout.write("-" * (getTerminalSize()[0])); sys.stdout.write("\n")
213         sys.stdout.write("\n")
214         print("TMSIs:")
215         sys.stdout.write("\n")
216         arfcn_c = 0
217         for row in tmsis:
218                 if not row == ['']:
219                         sys.stdout.write("ARFCN " + " " * (3 - len(arfcns[arfcn_c])) + arfcns[
                                arfcn_c] + ": ")
220                         width_c = (getTerminalSize()[0]-11)/9
221                         for i in range(0, len(row)):
222                                 sys.stdout.write(row[i])
223                                 if not i == len(row) - 1:
224                                         sys.stdout.write(" ")
225                                 if (i + 1) % width_c == 0:
226                                         # problem with complete full lines
227                                         if not (getTerminalSize()[0] - 11) % 9 == 0:
228                                                 # last item?
229                                                 if not i == len(row) - 1:
230                                                         sys.stdout.write("\n")
231                                         if not i == len(row) - 1:
232                                                 sys.stdout.write(" " * 11)
233                         if not i == len(row) - 1 or not (getTerminalSize()[0] - 11) % 9 == 0:
234                                 sys.stdout.write("\n")
235                 arfcn_c = arfcn_c + 1
236
237         sys.stdout.write("\n")
238
239 if __name__=='__main__':
240         main()
```

D.5 arfcn.py

```
#!/usr/bin/env python
#
# This script calculates the frequency to a given ARFCN
#

import sys
from optparse import OptionParser

def parse_options():
	parser = OptionParser()
	parser.add_option("-a", "--arfcn", type="int", dest="arfcn",
                                                help="Input is an ARFCN")

	(options, args) = parser.parse_args()
	if (len(args) != 0):
		parser.print_help()
		sys.exit(1)

	return options

def freq(arfcn):

	# GSM 900
	if arfcn > 0 and arfcn <= 124:
		freq = 890 + 0.2*arfcn + 45

	# GSM 1800
	elif arfcn >= 512 and arfcn <= 885:
		freq = 1710.2 + 0.2*(arfcn - 512) + 95

	else:
		freq = 0

	return freq

def main():

	# get parameter
	options = parse_options()

	# check options
	# wrong usage
	if not (options.arfcn):
		print "ERROR: no ARFCN is set, it seams i've nothing to do"
		sys.exit(1)

	# correct usage
	if (options.arfcn):
		print str(freq(options.arfcn))+'M'

####################
if __name__ == '__main__':
	main()
```

D.6 gsmlive.sh

```
#!/bin/bash
#
# This script captures an specified ARFCN. It creates a live dump to the gsm tun device and
     a pcap file for later analysis.
#

ARFCN=""
DECIM=112
BOARD="B"
WIRESHARK=0

# check for parameters and change config values
while getopts ":c:d:R:sh" optname; do
       case "$optname" in
              "c")
                     ARFCN=$OPTARG
              ;;
              "d")
                     DECIM=$OPTARG
              ;;
              "R")
                     BOARD=$OPTARG
              ;;
              "s")
                     WIRESHARK=1
              ;;
              "h")
                     echo "This script captures an specified ARFCN. It creates a live dump
                          to the gsm tun device and a pcap file for later analysis."
                     echo
                     echo "SYNOPSIS"
                     echo " gsmlive.sh [ -c ARFCN ] [ -d DECIM ] [ -R BOARD ] [ -s ] [ -h ]"
                     echo
                     echo "OPTIONS"
                     echo " -c ARFCN"
                     echo " Sets the ARFCN."
                     echo " -d DECIM"
                     echo " Sets the DECIM value."
                     echo " -R BOARD"
                     echo " Specifies which BOARD (A or B) of the USRP to use."
                     echo " -s"
                     echo " Don't start Wireshark. So there will be an output to the shell,
                          the pcap file and the gsm tun device. You can capture the gsm
                          device with any other tool of course."
                     echo " -h"
                     echo " Prints this help."
                     exit 0
              ;;
              "?")
                     echo "Unknown option $OPTARG"
                     echo
                     $0 -h
                     exit 1
              ;;
              ":")
                     echo "No argument value for option $OPTARG"
                     echo
                     $0 -h
                     exit 1
              ;;
              *)
                     echo "Unknown error while processing options"
                     echo
                     $0 -h
                     exit 1
              ;;
       esac
done

# check if ARFCN is set?
if [ "$ARFCN" == "" ]; then
```

```
68        echo "ERROR: no ARFCN set."
69        echo
70        $0 -h
71        exit 1
72  fi
73
74  # start wireshark?
75  if [ $WIRESHARK -eq 0 ]; then
76        if [[ $EUID -ne 0 ]]; then
77              echo "ERROR: you need to be root to capture with Wireshark."
78              exit 1
79        else
80              gsmshark -i gsm -k &
81        fi
82  fi
83
84  gsm_scan.py -R $BOARD -S I -p d -d $DECIM -c $ARFCN -r e | gsmdecode -i
85
86  exit 0
```

D.7 capture.sh

```
#!/bin/bash
#
# This script captures GSM traffic and stores it in a cfile. By default it iterates all
    ARFCNs.
#

# predefine config values
DURATION=4
DECIM=112
GAIN=52
ARFCN=""
BOARD="B"
DST=`pwd`"/"`date +%Y%m%d%H%M%S`

# check for parameters and change config values
while getopts ":t:d:g:c:R:d:h" optname; do
       case "$optname" in
              "t")
                     DURATION=$OPTARG
              ;;
              "d")
                     DECIM=$OPTARG
              ;;
              "g")
                     GAIN=$OPTARG
              ;;
              "c")
                     ARFCN=$OPTARG
              ;;
              "R")
                     BOARD=$OPTARG
              ;;
              "d")
                     DST=$OPTARG
              ;;
              "h")
                     echo "This script captures GSM traffic and stores it in a cfile. By
                         default it iterates all ARFCNs."
                     echo
                     echo "SYNOPSIS"
                     echo " capture.sh [ -t SECONDS ] [ -d DECIM ] [ -g GAIN ] [ -c ARFCN ]
                         [ -R BOARD ] [ -d DSTDIR ] [ -h ]"
                     echo
                     echo "OPTIONS"
                     echo " -t SECONDS"
                     echo " Duration in SECONDS to stay on each ARFCN."
                     echo " -d DECIM"
                     echo " Sets the DECIM value."
                     echo " -g GAIN"
                     echo " Sets the GAIN value."
                     echo " -c ARFCN"
                     echo " Scan only on ARFCN, don't iterate all."
                     echo " -R BOARD"
                     echo " Specifies which BOARD (A or B) of the USRP to use."
                     echo " -d DSTDIR"
                     echo " Sets the DSTDIR."
                     echo " -h"
                     echo " Prints this help."
                     exit 0
              ;;
              "?")
                     echo "Unknown option $OPTARG"
                     echo
                     $0 -h
                     exit 1
              ;;
              ":")
                     echo "No argument value for option $OPTARG"
                     echo
                     $0 -h
                     exit 1
```

```
69              ;;
70              *)
71                      echo "Unknown error while processing options"
72                      echo
73                      $0 -h
74                      exit 1
75              ;;
76      esac
77  done
78
79  # check or create the destination
80  if ! [ -d $DST ]; then
81          mkdir -p $DST
82  fi
83
84  # Look for usrp_rx_cfile.py
85  USRP_PROG=usrp_rx_cfile.py
86  while :; do
87          which "$USRP_PROG" > /dev/null
88          if [ $? -eq 0 ]; then
89                  break
90          fi
91          USRP_PROG=/usr/share/gnuradio/usrp/usrp_rx_cfile.py
92          which "$USRP_PROG" > /dev/null
93          if [ $? -eq 0 ]; then
94                  break
95          fi
96
97          echo "ERROR: usrp_rx_cfile.py not found. Make sure it's in your PATH!"
98          exit 1
99  done
100
101 # calculate samples for the specified duration
102 SAMPLES=`expr 64000000 / $DECIM '*' $DURATION`
103
104 # no single ARFCN specified
105 if [ "$ARFCN" == "" ]; then
106
107         # Now iterate all, about 1000 RFCNS, each for -t seconds
108         # GSM 900 (ARFCN 1 - 124) && GSM 1800 (ARFCN 512 885)
109         for ARFCN in `seq 1 124 && seq 512 885`; do
110
111                 FREQ=$(arfcn.py -a $ARFCN)
112                 FILE="$DST/arfcn_${ARFCN}_${FREQ}.cfile"
113                 echo "capturing for $DURATION seconds ($SAMPLES samples) on module $BOARD on
                        ARFCN $ARFCN (${FREQ}Hz) to $FILE"
114                 $USRP_PROG -R $BOARD -g $GAIN -d "$DECIM" -f "$FREQ" -N $SAMPLES $FILE > /dev/
                        null 2> /dev/null
115                 if [ $? -gt 0 ]; then
116                         echo "WARNING: something went wrong while capturing (ARFCN $ARFCN),
                                continuing on the next ARFCN anyway ..."
117                 fi
118
119         done
120
121 # capture only a single ARFCN
122 else
123
124         FREQ=$(arfcn.py -a $ARFCN)
125         FILE="$DST/arfcn_${ARFCN}_${FREQ}.cfile"
126         echo "capturing for $DURATION seconds ($SAMPLES samples) on module $BOARD on ARFCN
                $ARFCN (${FREQ}Hz) to $FILE"
127         $USRP_PROG -R $BOARD -g $GAIN -d "$DECIM" -f "$FREQ" -N $SAMPLES $FILE > /dev/null 2>
                /dev/null
128         if [ $? -gt 0 ]; then
129                 echo "WARNING: something went wrong while capturing (ARFCN $ARFCN) ..."
130         fi
131
132 fi
133
134 exit 0
```

D.8 analysecfile.sh

```
#!/bin/bash
#
# This script creates a summary and pcap file. It reads from a captured cfile.
#

# predefine config values
TMP=/tmp/gsmdump
DECIM=112

# check for parameters and change config values
while getopts ":s:d:h" optname; do
        case "$optname" in
                "s")
                        TMP=$OPTARG
                ;;
                "d")
                        DECIM=$OPTARG
                ;;
                "h")
                        echo "This script creates a summary and pcap file. It reads from a
                             captured cfile."
                        echo
                        echo "SYNOPSIS"
                        echo " analysecfile.sh [ -s TMPDIR ] [ -d DECIM ] [ -h ] CFILE"
                        echo
                        echo "OPTIONS"
                        echo " -s TMPDIR"
                        echo " Sets the TMPDIR."
                        echo " -d DECIM"
                        echo " Sets the DECIM value."
                        echo " -h"
                        echo " Prints this help."
                        exit 0
                ;;
                "?")
                        echo "Unknown option $OPTARG"
                        echo
                        $0 -h
                        exit 1
                ;;
                ":")
                        echo "No argument value for option $OPTARG"
                        echo
                        $0 -h
                        exit 1
                ;;
                *)
                        echo "Unknown error while processing options"
                        echo
                        $0 -h
                        exit 1
                ;;
        esac
done

# check for CFILE
for p in "${@:$OPTIND}"; do
        if [ "$CFILE" == "" ]; then
                CFILE="$p"
        else
                echo "ERROR: Too much files specified."
                echo
                $0 -h
                exit 1
        fi
done

# no CFILE was set
if [ "$CFILE" == "" ]; then
        echo "ERROR: No file specified."
        echo
```

```
71        $0 -h
72        exit 1
73  fi
74
75  # CFILE is relative
76  if ! [ `echo -n $CFILE | grep -e "^/.*"` ]; then
77        CFILE=`pwd`"/"$CFILE
78  fi
79
80  # check or create the tmp
81  if ! [ -d $TMP ]; then
82        mkdir -p $TMP
83  fi
84  cd $TMP
85
86  # rm old netlists
87  if [ -f netlist.nets ]; then
88        rm netlist.nets
89  fi
90  if [ -f netlist.imsis ]; then
91        rm netlist.imsis
92  fi
93  if [ -f netlist.tmsis ]; then
94        rm netlist.tmsis
95  fi
96
97  FILE=`echo -n $CFILE | sed -e "s/\(.*\)\.[^.]*$/\1/"`
98  gsm_scan_light.py -SN -pd -d "$DECIM" -I "$CFILE" > /dev/null 2> /dev/null
99  if [ -e tvoid.pcap ] && [ -e tvoid-burst.pcap ]; then
100       if [ `stat -c %s tvoid.pcap` -gt 24 ]; then
101             mv tvoid.pcap $FILE.pcap
102             mv tvoid-burst.pcap $FILE-burst.pcap
103             pcap4gsmdecode.py $FILE.pcap | gsmdecode -i | analyse.py
104             # check if there are any networks
105             if [ -f netlist.single.nets ]; then
106                   summary.py -s | tee $FILE.txt
107                   rm netlist.single.nets
108             else
109                   echo
110                   echo "I'm sorry, but no network was found"
111                   echo
112             fi
113       fi
114 fi
115
116
117 # rm old netlists
118 if [ -f netlist.nets ]; then
119       rm netlist.nets
120 fi
121 if [ -f netlist.imsis ]; then
122       rm netlist.imsis
123 fi
124 if [ -f netlist.tmsis ]; then
125       rm netlist.tmsis
126 fi
127 if [ -f netlist.single.nets ]; then
128       rm netlist.single.nets
129 fi
130 if [ -f netlist.single.imsis ]; then
131       rm netlist.single.imsis
132 fi
133 if [ -f netlist.single.tmsis ]; then
134       rm netlist.single.tmsis
135 fi
136 if [ -f gsm-receiver.pcap ]; then
137       rm gsm-receiver.pcap
138 fi
139 if [ -f gsm-receiver-burst.pcap ]; then
140       rm gsm-receiver-burst.pcap
141 fi
142 if [ -f speech.gsm ]; then
143       rm speech.gsm
144 fi
```

```
145 if [ -f cfile2.out ]; then
146         rm cfile2.out
147 fi
148
149 exit 0
```

Literaturverzeichnis

[1] *airprobe Projekt-Homepage*. https://svn.berlin.ccc.de/projects/airprobe/.

[2] *GNU Radio Projekt-Homepage*. http://gnuradio.org/redmine/wiki/gnuradio.

[3] *Wireshark Projekt-Homepage*. http://wiki.wireshark.org/Development.

[4] *Mail 419: Re: [airprobe-main] Listening to the uplink with Airprobe gssm*. airprobe Mailingliste, 12 2009. https://svn.berlin.ccc.de/projects/airprobe/wiki/MailingLists.

[5] *Mail 626: Re: [airprobe-main] Capture uplink GSM packets*. airprobe Mailingliste, 05 2010. https://svn.berlin.ccc.de/projects/airprobe/wiki/MailingLists.

[6] ETSI: *GSM 04.08 version 7.7.1 Release 1998*.

[7] Göller, Joachim: *Die GSM-DM-Kanäle im Dialog. Einstieg in die Mobilfunk-Signalisierung*. Epv, 3 2003.

[8] Göller, Joachim: *Signaling in Mobile Radio Communication: According the the GSM standard*. Epv, 1 2006.

[9] Maucher, Johannes: *Skript zur Vorlesung Mobile Communications Systems*.