Mikko Suhonen

# TEST AUTOMATION IN AN RNC ENVIRONMENT

# TEST AUTOMATION IN AN RNC ENVIRONMENT

Mikko Suhonen
Bachelors Thesis
Spring 2014
Information Technology
Oulu University of Applied Sciences

# TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, Langattomat laitteet

---

Tekijä: Mikko Suhonen
Opinnäytetyön nimi: Testausautomaatio RNC ympäristössä
Työn ohjaajat: Veijo Korhonen, Markku Jurmu
Työn valmistumislukukausi ja -vuosi: Kevät 2014
Sivumäärä: 35 + 1 liitettä

---

Työn tilaajana toimii Oy L M Ericsson AB, ja työn tarkoituksena on kartoittaa mahdollisia vaihtoehtoja tukiaseman testausautomaation toteuttamiseksi RNC -ympäristössä. Testausta tehdään järjestelmätasolla.

Työn tavoitteena on tutkia eri vaihtoehtoja automaation toteuttamiseksi, jonka jälkeen ratkaisut implementoidaan järjestelmään. Työn aikana myös rakennetaan järjestelmä jossa testejä ajetaan.

Avaintekijät joiden ehdoilla järjestelmää kehitetään ovat luotettavuus sekä käytettävyys. Koska järjestelmää ohjaatavat työkalut käyttävän kometoriviin pohjautuvaa käyttöliittymää, järjestelmä kehitetään Linux ympäristöön. Suurin osa tiedosta on kerätty yli vuoden kestäneen harjoittelujakson aikana.

Automaatiojärjestelmä rakennettiin onnistuneesti, ja testauksen laajuutta kasvatettiin kehitystyön aikana. Tämä vaikutti järjestelmän kompleksisuuteen, joten suunniteltua testiympäristöä jouduttiin laajentamaan. Järjestelmän stabilisuuden kehittäminen jatkuu edelleen. Järjestelmä toimii kuitenkin niin vakaasti, että se on otettu käyttöön uusien ohjelmistopakettien testauksessa.

Järjestelmä suunniteltiin niin että siihen voidaan lisätä vaativampia testejä tarvittaessa.

---

Asiasanat: WCDMA, RNC, Testiautomaatio

# ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Wireless Devices

---

Author: Mikko Suhonen
Title of thesis: Test Automation in an RNC Environment
Supervisors: Veijo Korhonen, Markku Jurmu
Term and year of completion: Spring 2014, Pages: 35 + 1 appendices

---

This Bachelor's thesis was ordered by Oy L M Ericsson AB Oulu to help to investigate possible solutions for a base station test automation development in an RNC environment, covering the system testing area.

The objective was to research different methods of execution and how to implement the most practical solution into the test set. This includes building an environment in which the test cases were executed.

The key factors in deciding the platform on which to build the system were accessibility and reliability. Since most equipment can be controlled via command-line interface, Linux based environment is the main target of research. Most of the information was acquired as a trainee over a 1-year period and from experienced engineers working in the company.

The automation system was successfully built and the scope of the tests to include into the automation was expanded during the process. This increased the complexity of the test environment from the original plan. The development of system stability is still ongoing, as there are some issues regarding certain user equipment. Even though there are issues with the system, it is deemed stable enough for a daily test execution.

A further development of the automation system is possible, as the basic set is created in such a way that it is easy to implement into more complex test cases.

---

Keywords: WCDMA, RNC, Test automation

## CONTENTS

## LIST OF ABBREVIATIONS

ADB = Android debug bridge

CLI = Command-line interface

CS = Circuit switched

dBm = Decibel-milliwatts

FTP = File transfer protocol

GUI = Graphical user interface

HO = Handover

HSDPA = High-speed downlink packet data access

HSPA = High-speed packet data access

HSUPA = High-speed uplink packet data access

HW = Hardware

IMSI = International mobile subscriber identity

IP = Internet protocol

MRAB = Multi radio access bearer

OS = Operating system

PPP = Point-to-Point protocol

PS = Packet switched

R&D = Research and development

RAB = Radio access bearer

RBS = Radio base station

RNC = Radio network controller

SDK = Software development kit

SHO = Soft handover

SSH = Secure shell

SW = Software

UARFCN = UTRA absolute radio frequency channel number

UMTS = Universal mobile telecommunications system (3G)

UTRA = UMTS terrestrial radio access

UE = User equipment

WCDMA = Wideband code division multiple access

# 1 INTRODUCTION

This thesis work was ordered by L M Ericsson Oulu to help to investigate possible solutions for test automation system in an RNC environment. The company was founded 1876, and the global headquarters are located in Stockholm, Sweden. There are over 110,000 people working in the company, of which 25,300 are working in R&D (Research And Development) [1]. The Oulu site was founded in early 2012, and the site houses a small WCDMA (3G) and LTE (4G) R&D team.

The main aim of the thesis was to develop a test automation system for test case execution when testing an RBS (Radio Base Station) in a WCDMA network. The basic test set includes testing voice calls (CS-call), data performance (PS data access) and RBS features (HSPA etc.). If a valid method for executing the test set is discovered, it will be studied further for implementation into the automation system.

Since most of the software tools controlling the network side of the system are command line based, the decision to develop the automation on a Linux platform was made. Choosing to develop the automation in a Linux environment has certain advantages. First, Linux has an impressive amount of different command line tools already integrated to the system, and the diversity of free tools available for Linux is huge. Those tools are open source software and freely distributed, so using the SW to develop the automation does not cause any problem regarding licenses or copyrights. Second, Linux is a robust environment that does not require much processing power from the system. These points allow the automation system to be operated with shell scripts. More on the basic functionality scripts can be found in chapter 3.

The automation system is controlled by using shell –scripts. Android devices need to be controlled via ADB (Android Debug Bridge), which is an interface tool between the UE and Linux machine. Without it, the UE is not accessible from the shell. 3G dongles have also special conditions that need to be taken

into consideration. More detailed information about controlling different UEs can be found in chapter 2.3.

While the system was developed in an OTA (Over The Air) environment, the final system is implemented in a cabled system. This does not change how the system is controlled, so the basic model introduced in chapter 2.2 can be used in building the final setup.

When the basic system was nearly completed, the decision to add handover support to the automation system was made. This extended the scope in which the automation is developed, and brought more challenge to the work. The system needed to be remodeled to include a second node and more complex tracking of the system.

The focus of this thesis is in the system testing area of a base station, so the WCDMA network and its features are not described in detail. The basic mode of the network is depicted in the second chapter in the extent that is necessary for the developed automation environment to function. This is done to help the reader to understand the functionality of the whole system.

# 2 BASIC TEST ENVIRONMENT AND TERMINOLOGY

There are several abbreviations and terms that appear regularly in this thesis. Therefore the basic terms and network elements should be made known before starting to study the automation system.

## 2.1 Terminology and network components

Below is a basic depiction of a WCDMA (Wideband Code Division Multiple Access) infrastructure:

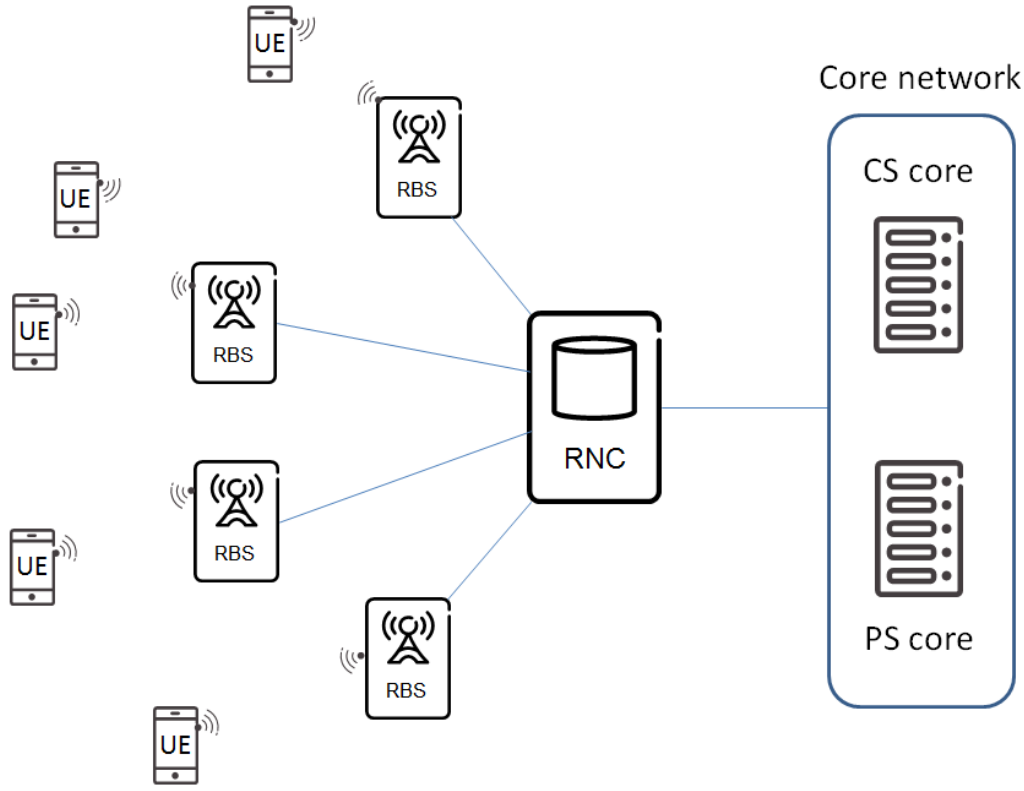


*FIGURE 1. Basic structure of a WCDMA network*

The infrastructure of a WCDMA system consists of an RNC (Radio Network Controller), an RBS (Radio Base Station), a PS (Packet Switched) and a CS (Circuit Switched) core networks. The RNC is responsible for controlling the RBSs that are connected to it. This includes controlling e.g. feature availability, transmission power, cell size. The RBS, also known as node, relays the

information to and from the network via a radio connection to an UE (User Equipment), meaning phones and 3G data dongles. PS and CS cores are the backbone of the network in which the information is transmitted from one user to another. The PS core handles the data packet transfers of the UE, while the CS core handles normal voice calls.

An RAB (Radio Access Bearer) means the connection type the UE has in the network, covering both PS and CS access. There are several different RABs available in the testing environment, but only a handful of them are implemented for testing. The automation covers basic functionality tests, so the more complex RAB combinations are not included in the test set.

MRAB (Multiple Radio Access Bearer) means that the user has several active connections simultaneously. The possibilities are limited to what kind of RAB combinations are supported by the RNC.

This description of the network components is the extent of the environment in which the automation is developed. And since the focus is on the test automation development, the WCDMA network and its features are not covered in more detail in this thesis.

### 2.2 Description of the development environment

The development environment includes:

- UE (User Equipment)
- RBS (Radio Base Station)
- RNC (Radio Network Controller)
- PC (Personal Computer)
- Server (Linux platform)

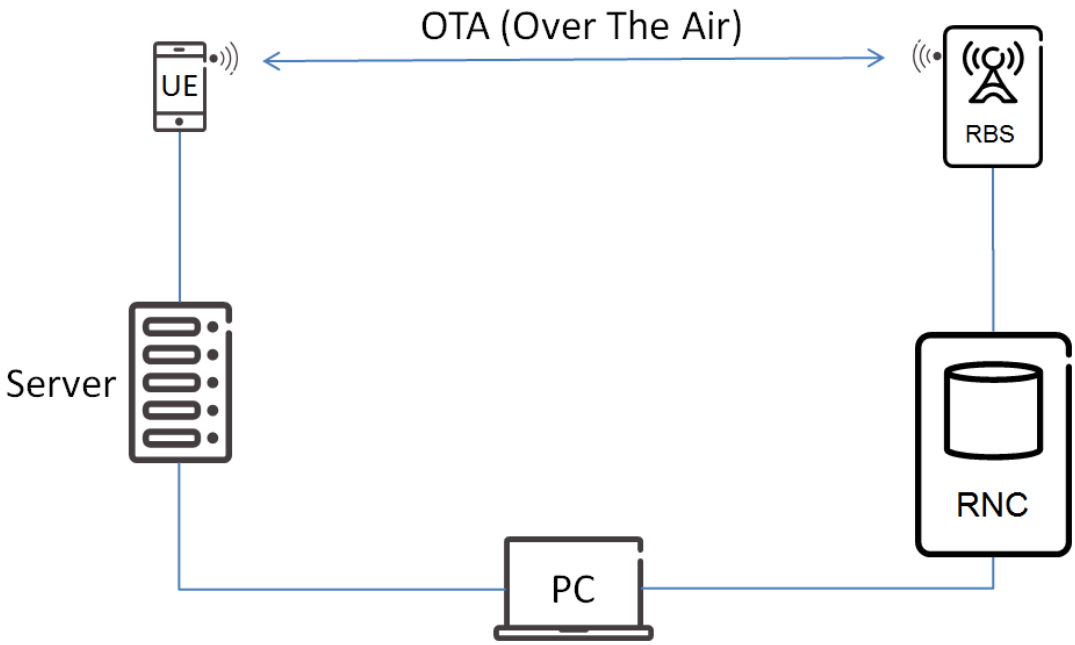


*FIGURE 2. Development environment*

Figure 2 describes the test environment where the development process was started. The setup can be further expanded by adding nodes, UEs, a remote controlled attenuator, a spectrum analyzer, etc. depending on the test case.

The Linux server environment was the only thing that needed to be set up, as the RBS and RNC were pre-configured and the PC had necessary tools installed.

The UEs that are tested are connected to the Linux server, and the server is controlled via a remote connection from the PC. The PC also has the required tools to control the RNC and node remotely. The remote access for all components is necessary while developing the automation system. This is to ensure that all devices are working properly.

The test environment was expanded during the development phase to include more complex test cases. A description of the extended environment will be given further in the thesis, after the basic functionality has been introduced.

### 2.3 Different UE’s used

While creating the automation system, several different UEs were tested to see, which would be best suited to use in the test set. These include different Nokia Asha devices, Android devices and 3G data dongles.

One key requirement for all UEs is that there is an external antenna port available as the devices and the RBS will be connected via an RF-cable. The final system has its connectivity done via cables as the radio spectrum is limited, and susceptible to interference. This also ensures that the UEs used are connected to the correct node and will not interfere with other test environments. This was not a requirement for the development environment as the system is constantly monitored, and the solutions can be implemented to the cabled environment without any modifications.

For the PS data performance USB dongles were used, while phones were used to establish basic CS calls and MRAB connections. The reason for separating the connectivity and data performance to separate UE types is that USB dongles are designed solely for a PS data access, as the device cannot perform CS calls, therefore usually having better data performance.

The next step is to take a look of pros and cons of different types of UEs.

#### 2.3.1 Android UEs

The Android UEs that are considered for automation are Samsung Galaxy S2, Galaxy SIII mini and Sony Xperia V.

The Android platform itself has some limiting factors which need to be taken into consideration when planning and setting up the system: the platform limits the usability of the modem, the reason being the way it communicates with the hardware. This means that the hardware is not freely accessible through a serial connection. Also, the system is fairly complex compared to e. g. Nokia phones, and can cause problems when bypassing the core system to control the HW directly.

One additional feature has to be enabled from the UE: the device has to be rooted. This enables the root-user account in the phone, which means that the user has administrator rights in the system. Without rooting some of the functions cannot be accessed remotely via a command line.

The Galaxy S2 has a few system impacting flaws we came across while testing with it. First, the upload from a phone to an FTP server is malfunctioning when performed via a dial-up connection. Second, the phone disconnects CS-calls seemingly randomly. The second problem was investigated thoroughly, and it was found to be UE initiated and it had nothing to do with the network or the RBS under testing. Therefore the usability in long CS stability tests is limited.

The Galaxy SIII mini performed well on preliminary tests. Data can be sent successfully both in upload and download directions while using a dial-up connection, while throughput values are not as good as when using a 3G USB dongle.

While configuring the Sony Xperia Z for use with Wvdial, the modem could not be recognized by the program even when the vendor ID was added, so this phone type was discarded as a possible candidate for the automation system. It might be possible to get the UE working with Wvdial, but the time and effort that would be needed was not feasible, as there were different models available that can be used.

One common factor for all tested Android models is that the UE needs to be rebooted occasionally, because the modem stopped answering the AT commands sent by the dial-up program. Fortunately the ADB (Android Debug Bridge) connection can be used to issue a reboot command remotely. The ADB is a command line tool that can be used to communicate with an Android device [2]. After restarting the device, the connection functioned normally. This instability needs to be taken into account when controlling the Android devices in the final system. The reboot command is easy to adapt to the execution of the test case, and the device can be rebooted in a problem situation, after a test set, and if necessary, after a single test case. The downside resulting from this reboot is that the time needed to run the whole test set will be longer, as the re-

boot and reconnection of the device takes on average about 1 minute, depending on the device.

### 2.3.2 Nokia UEs

The different types of Nokia phones were limited to the most basic models using Nokia Asha OS e. g. Nokia 311.

Nokia phones have certain advantages over Android phones while used remotely. The Asha OS does not interfere with the direct connection to the HW as the Android platform does, and the devices are more stable.

While testing the Nokia UEs, the downlink throughput values were lower than when using Android devices, but were still at a good level. Considering that the devices are of more basic model compared to the Android devices under testing, the lower throughput value is not an issue, at least not from the testing standpoint. The reason for this is that the Android devices are used primarily in the automation system. If more users are needed, then the Nokia phones can be taken into use.

### 2.3.3 3G USB dongles

The different types of dongles tested for usability are manufactured by Sierra and Huawei, and were limited to a few models.

One major complication arose while testing the 3G dongles. The dongle itself has its onboard flash memory divided into two partitions. The mounted partition depends on the detection of the installed drivers. If no driver is detected, the UE mounts the partition which has the needed files for a device driver installation. Otherwise the functional partition is mounted, which enables the device to function as a modem for the connected system. To use the dongle for testing, we needed to find a way to change the mode of the device from an installation to a functional. To achieve this, a program called USB mode switch [3] was installed in the Linux system. The program switches the UE from the installation mode to a functional mode so that Wvdial will recognize the device as a

modem, and a data connection can be made. This procedure had to be done to all devices

The first Sierra model performed extremely well in the data performance, usability and stability. The only drawback being that the model is discontinued, and that there was only one device available.

The second tested Sierra model is more compact, but the reliability is poor. While trying to get the UE to perform well enough to use in the automation system, it caused more problems than solved them. The data connection was severed while connected to the system, and the modem did not respond to further dialing attempts. The throughput achieved with the devices was also of poor quality, so the device was not taken into use.

The Huawei dongle used in the testing performs well when measuring the data performance. The stability was not as good as in the first Sierra model, but a solution was found to circumvent this problem. Rebooting the platform to which the UE is connected reboots the device. This eliminates the problems with the stability over a long period of time, so the device can be taken into use.

**Selecting the UEs for further development**

Once the UEs have been deemed usable in the automation environment, the next task is to use the required features while controlling the device remotely via a command line.

Before the features can be tested, we need to find out how to configure different UEs in the Linux environment, and what possible control issues the selected development platform has.

# 3 ESTABLISHING PS- AND CS-CALLS

Before controlling multiple UEs in various test cases, we need to establish a PS-call and a CS-call with 1 UE. The reason for doing this is that we have a reference point from which to expand the automation to include more complex tests e.g. the MRAB and several UEs. The link direction in the thesis is used from the UEs standpoint, meaning that a downlink (download) is used when the UE receives information from the network and uplink (upload) when sending information to the network.

## 3.1 Establishing a PS connection using a dial-up

There are several different software applications available which can be used to create a dial-up connection. However most of them are GUI (Graphical User Interface) based and therefore not suitable for the automation as the main goal is to perform all needed functions through a command line interface. One program which uses the command line to function is Wvdial [4]. The program has a fairly large database of supported modems pre-configured including but not limited to:

- several Nokia phones
- 3G dongles from different manufacturers

Android devices are not supported by default, so to get the modem operational, Android platform developer forums have a list of vendor IDs, which can be added manually to a Linux USB device database [5]. After adding the vendor type to the database, the modem is recognized by Wvdial. All Nokia phones used while developing the automation were supported by Wvdial by default.

Once the UE is successfully connected via the USB to the platform, the configuration for the modem is done giving a ‘wvdialconf’ -command. This configures the correct ports, modem type and baud rates which the modem uses to communicate. Then the correct APN (Access Point Name) and dial-up settings, meaning the user name, password and the number to dial, are configured to the ‘wvdial.conf’ file located in /etc/ -folder in the Linux system.

Different dial-up profiles can be created to the file, so several UEs can be used on the same platform. Below is a basic configuration example for a Samsung Galaxy S2:

[Dialer S2]
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Init3 = at+cgdcont=1,"IP","<APN-name>"
Stupid Mode = 1
Modem Type = USB Modem
ISDN = 0
New PPPD = yes
Phone = <phone number to dial>
Modem = /dev/ttyACM0
Username = '<user name>'
Password = '<password>'
Baud = 460800

In the above example, the first line defines the name of the profile for this specific setup (Dialer S2). The following lines are the necessary configuration information for the modem, PDP context information [6] and user information for the network access.

Once the configuration is done, the program itself is easy to use. In the example the configuration was named [Dialer S2], so the configuration is recognized as S2. By giving command ‘wvdial S2’ causes the program to try and create the dial-up connection with the specified configuration. The program uses a PPP (Point-to-Point Protocol) connection to establish internet connectivity. If the connection is established successfully, the device is assigned an IP address by the network DHCP (Dynamic Host Configuration Protocol) server. The UE uses the acquired address to communicate in the network e.g. data transfers. The created data connection is usable via the platform to which the phone is connected, in this case, the server.

### 3.1.1 Basic FTP commands

Once the data connection is established, we can use a command line FTP to execute the basic data transfer tests. The format can be fairly simple:

```
#!/bin/bash
HOST=<host IP address>
USER=<user name>
PASS=<password>
ftp -inv $HOST << EOF >> /<path>/<log file name>
user $USER $PASS
cd <folder>
get <desired file on FTP server>
bye
EOF
```

This is the basic model for a shell script used to execute a single file data transfer. As the scrip is done using a bash shell command set, the program can be used in UNIX and Linux environments. The required user account details and server IP address are saved to variables that can be called in the login process further down the script. The first operation is to open a FTP connection to a desired server, and to start the logging of the session. The FTP program has three options enabled when the program is started:

- i: disables interactive prompts from the server during multiple file transfers
- n: disables the auto-login feature upon an initial connection
- v: shows all responses from remote server and reports data transfer statistics

Then the user account details are sent to access the server file system. Once connected, the path is changed (if needed) to where the desired file is located, and finally ‘get’ command is sent to retrieve the specified file. After the transfer is complete, the FTP session is terminated by the ‘bye’ command. EOF (End Of File) is used to restrict the captured information to include only the messages

from the FTP session and to save them to the designated log file. For the log file to be useful the v-option is crucial, as the data transfer statistics are necessary to determine if the test case is successful or not.

The script can be run in the Linux server by giving a command ‘./<name of script>.sh’, where .sh is the file extension used for shell scripts. Before the script can be run though, the script file needs to have execution privileges. This can be accomplished with a ‘chmod’ command e.g. ‘chmod +x <file name>’.

The FTP program used reports the average data transfer rates from the session after the file transfer is completed to the log file specified in the script above.

If the user wants to upload a file into the server, the ‘get’ command is replaced with ‘put’:

```
#!/bin/bash
HOST=<host IP address>
USER=<user name>
PASS=<password>
ftp -inv $HOST << EOF >> /<path>/<log file name>
user $USER $PASS
cd <folder>
put <desired file to be uploaded>
bye
EOF
```

The designated file has to be in the current working directory, which is the folder from which the FTP program was started by default and the user account has to have the necessary access rights to write the file to the server. The working directory can be changed with an ‘lcd’ command, using basic Linux directory commands e.g. ‘lcd ..’, which changes the working directory to the previous folder in the local machine e.g. from ‘/<folder1 name>/<folder2 name>/’ to ‘/<folder1 name>/’.

### 3.1.2 Performing simultaneous file transfers

Once the basic script has been tested so that it performs the required function properly, it can be run in the background by adding an ‘&’ character after the script execution command. Once the script is running in the background, another transfer can be started by executing another script. Below is an example of a script file which performs 2 separate transfers both from a script file:

```
#!/bin/bash
for i in {1..n}
do
./<download script file>.sh &
sleep n
./<upload script file>.sh &
sleep n
wait
done
```

The script starts with a for loop counter, which causes the script to run the transfers n times. The ‘sleep’ action makes the system wait n seconds before executing the next command. This is done to give the ftp program time to log in and begin the transfer before opening the next connection. In the end the ‘wait’ command is issued to tell the system to wait that both actions have finished before starting the next loop. This way both transfers are finished before the system repeats the script so that there are not any complications from downloading or uploading the same file from the same FTP server.

## 3.2 Establishing a CS connection

Placing and ending a call is simple to perform in Nokia phones, as the phones accept an AT command for dialing. The baud rate in which the modem communicates needs to be known when issuing AT commands, which can be found using ‘wvdialconf’. When placing a call, a serial connection is opened to the phone and a number is dialed using ‘ATDT<receiving phone number>’. The

D stands for ‘Dial’ and the T for ‘Touch tone’. When ending the call command ‘ATH’ is sent, where the ‘H’ stands for ‘Hang up’

Creating a CS-call via a command line using an Android phone is a bit trickier than giving the modem basic AT commands. The modem does not directly accept commands to create a basic voice call, so the use of the ADB (Android Debug Bridge) is needed to send the voice call command to the phone.

The ADB can be found in the Android SDK (Software Development Kit), which is a freely distributed open source software [7].

The command to establish a voice call using a command line is:

‘adb_rpi shell service call phone 2 s16 "<phone number>” ‘

This uses the ADB to send a command to the phone using a Linux shell, on which the Android platform is based upon. The ‘service call’ command is a command that we want to use a certain service located in the phone, in this case ‘phone’ which means a voice call, and ‘2’ which means an outgoing call. ‘s16’ means that the message is sent using UTF-16 Unicode format which the phone can interpret. The phone number to which the call is made is the last parameter in the command.

The receiving phone needs to be configured to automatically answer calls; otherwise the call will never be successfully connected with the other UE. The current generation of mobile phones usually has an auto answer option available in the phone configuration menu.

Terminating the call on an Android phone can be done via the ADB by simulating a key press event. The command is as follows:

‘adb_rpi shell input keyevent 6’

Again the shell command is used as before when making the call. In this event, we only send the command to terminate the call, which is in the above example ‘keyevent 6’.

### 3.3 Verifying the connection and node feature availability

After either the PS or CS connection is established, we need to verify that the UE is connected to the correct node. This can be accomplished by connecting to the RNC and verifying that the correct IMSI (International Mobile Subscriber Identity) is registered to the cell of the node under test and the correct RAB has been achieved. The IMSI is a unique identifier for every single user in the network.

The interface tools used to control the RNC are private information of the company, and therefore cannot be demonstrated here.

Since the tool is command line based, the results can be parsed using a ‘grep’ command in the Linux shell environment. The basic principle is that the ‘grep’ function returns a printout from a certain IMSI number connected to the correct node and has the correct RAB combination enabled. If the string was found, the connection is successful.

While testing different features supported by the node, the same procedure can be used. When certain features e.g. an HSDPA PS RAB availability is activated or deactivated, the information can be parsed from the command line response sent by the RNC if the action was successful or not.

# 4 TEST SET

Now that the basic control scenarios are explained, we can look at the test cases that are a part of the automation system. Below is a description of basic test cases:

*TABLE 1. Descriptions and pass criteria of basic test cases*

| Test case | Pass criteria |
|---|---|
| **Voice call** | RAB enabled in RNC |
| **PS Uplink** | Measured throughput greater or equal to X |
| **PS Downlink** | Measured throughput greater or equal to X |
| **PS Uplink & Downlink simultaneously** | Measured throughput greater or equal to X |
| **Voice + PS Uplink** | RAB enabled in RNC, Measured throughput greater or equal to X |
| **Voice + PS Downlink** | RAB enabled in RNC, Measured throughput greater or equal to X |
| **Voice + PS Uplink & Downlink simultaneously** | RAB enabled in RNC, Measured throughput greater or equal to X |
| **Enabling feature N from RNC** | Node cell up, feature enabled, PS and CS tests successful |

| | |
|---|---|
| **Disabling feature N from RNC** | Node cell up, feature disabled, PS and CS tests successful |

One common requirement for all cases is that the UE is registered to the correct node. This information is available from the RNC, and is confirmed before the test case is executed.

### 4.1 One user cases

These cases include the basic functional test cases that can be executed with one UE, which include several different data performance tests, basic CS tests and MRAB test cases.

The data performance tests include downloading, uploading and performing both uplink and downlink transfers simultaneously. The throughput of the UE is measured from different file transfers, and the performance is evaluated from the collected information. Certain limits are set which the throughput needs to achieve for the test case to be passed.

The network in which the tests are run has an automated answering machine available that keeps the connection active for 5 minutes at a time, so the basic CS call test can be executed with only one UE. The system supports several connections simultaneously. Depending on the UE which is used to make the voice call to this machine, the connection is either a CS RAB or an AMRWB (Adaptive Multi-Rate Wideband) which is a 3GPP specified speech encoding [8]. The AMRWB has a better speech quality than basic CS calls and it uses more bandwidth to achieve this. One reason why the automation system uses older Android UEs is that those models do not support the AMRWB. This allows the automation test set to include a larger variety of test cases.

Data performance tests are cases where different features are verified to function properly. This means measuring the throughput values on a specific PS RAB. Although some of these features are not used in live networks, the

redundancy needs to be fully functional in case the network has some malfunc-malfunctions where the most current features cannot be activated.

### 4.2 Several user cases

Most of the several user cases cover data performance in the node when two or more users are connected, and transfer data in different directions e.g. the first one downloads data using the HSDPA access, and the other UE uploads using the HSUPA feature.

### 4.3 Node feature testing

The system also performs node feature tests, including a robustness test set, which performs a node reboot, an SW upgrade and feature enabling and disabling tests in succession. After every action, the UE connected to the node checks that the connection can be made and active features are functioning. Once this is confirmed, the script performs the next action listed.

### 4.4 Interpreting the results

Once the individual scripts have finished the task, the result from the log can be parsed using the ‘grep’ –command. The log contains information of the FTP session. Below is an example of a download log file:

Connected to <server IP>
220 Welcome message
331 Please specify the password.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
250 Directory successfully changed.
local: <file name on user system> remote: <file name on FTP server>
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for <file name on FTP server>
(XXXXXXXXX bytes).
226 File send OK.

XXXXXXXXX bytes received in XXX.XX secs (XXXX.X kB/s)
221 Goodbye.

The necessary information from a successful transfer is the throughput value ‘(XXXX.X kB/s)’. This can be collected from the log by using ‘grep’:

‘grep “kB/s” <FTP log file name> >> <grep log file name>’

The ‘>>’ parameter writes the throughput information to the next line in the log file instead of overwriting the existing information. If several loops are run in sequence, the loop counter can be used to separate the different runs from each other:

‘echo “Loop $i results” >> <grep log file name>’

where ‘$i’ is the counter value in the current loop. This is done before the results are saved into the log file, so that the values received from ‘grep’ are saved after the ‘echo’ –command.

The collected information of the transfer is compared to the limit set for the throughput of the test case. If the value is higher or equal to the limit set, the test case is passed. Each case has the limit evaluated separately, as the throughput varies depending on the features enabled in the node. The limits with which the results are compared are according to 3GPP specifications [9].

The information from all test cases can be saved into a common log. This way the results are easier to interpret, as all required information is stored in a single file. When this is done, a filter can be added. It performs checks if the throughput is high enough for passing a test case. The pass and fail reasons can be added to further help in interpreting the results e.g. a low throughput or a carrier not available, so the tester can decide which tests need a more thorough investigation.

The same method can be used to check that the RNC has the UE or node are performing certain functions successfully, e.g. RAB selection for an UE, a feature activation for the node.

The results from the tests cannot be introduced in this document, as the performance results cannot be shared externally.

### Expanding the test set

Once the system was implemented to a daily execution, another test site was decided to be built, where SHO cases can be implemented to the automation. This is done to free testers to focus on other more demanding tasks.

The setup is similar to the basic environment described in the beginning of the thesis, but the SHO test site has two nodes and a remote controlled attenuator added to the system. In the next chapter the test environment is explained, and the added test cases are described further.

# 5 EXPANDED TEST ENVIRONMENT

The basic test setup was expanded to include SHO (Soft Handover) tests; therefore a remotely controlled attenuator, a second node and several UEs were added to the setup:

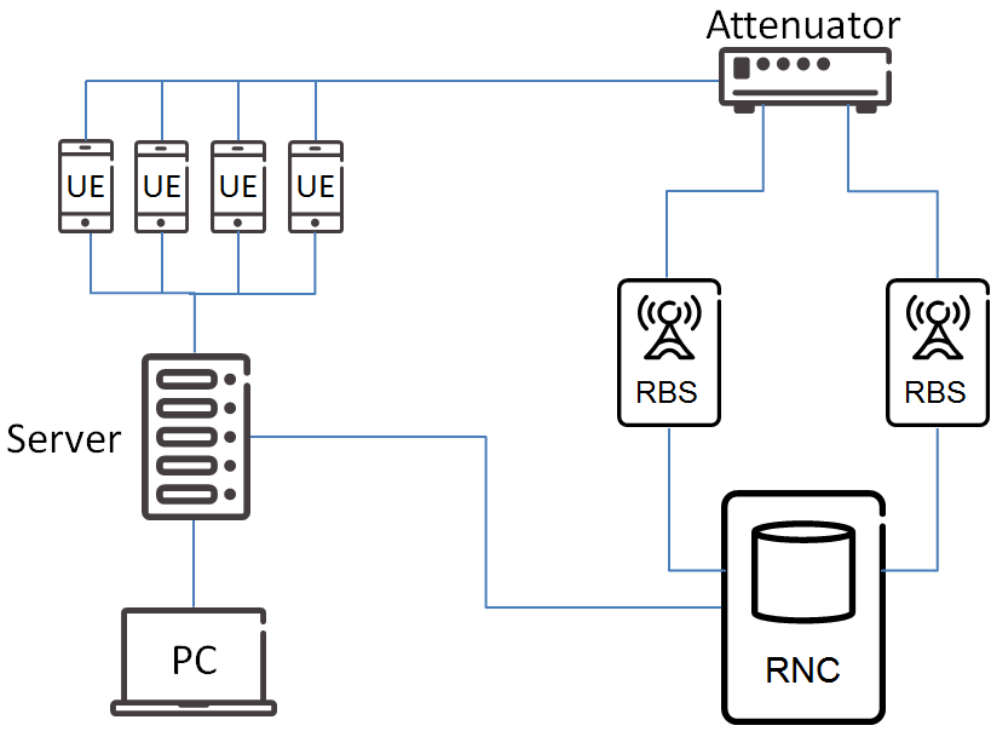


*FIGURE 3. Depiction of extended test environment*

A soft handover happens when a UE is switching between two nodes which operate on the same frequency. The remote controlled attenuator can be configured to slowly dial the attenuation up on one node, causing the signal strength on that node to decrease simulating a user walking away, and to dial down on the other node, causing the signal from that one to get stronger, as if the user walked towards that node. Once the UE connected to node 1 measures that the signal from node 2 is stronger, the handover is performed so that the active connection is maintained by node 2 instead of node 1.

First, the attenuator has to be configured into the network for the remote access to work. The initial configuration is done via a serial port connection, where an Ethernet control mode, a designated IP-address and network parameters are set. When this procedure is done, the attenuator can be controlled remotely from the network.

The attenuator has a handover support built in. This makes adapting the handover test cases to the automation much easier. The needed attenuation values were measured, to see in which level the UE has strong signal, and in which value the other nodes signal cannot be received anymore. These values are then adapted to the script. When indicating signal strength or attenuation, the unit is dB or dBm. The difference between the two units is that dBm means the measured power relation in decibels to one milliwatt (mW), and dB to one watt (W).

The nodes were configured to function in the same frequency, as is required in SHO tests. This is done from the RNC by setting the UARFCN (UTRA Absolute Radio Frequency Channel Number) [10] to the same value for both node configurations.

While controlling multiple UEs simultaneously, the devices are recognized by the PPP device number created by the dial-up program, the serial number by the ADB program or by the serial port on which the UE is connected.

The dial-up program creates the connections incrementally, the first being ppp0, the second ppp1 and so on. The system also gives the UE a local IP address and the remote IP address is given by the network DHCP server. This information can be used to add a route to which the traffic is forwarded to:

‘ip route add <destination IP address/subnet mask> via <local device IP address/subnet mask> dev pppX’

After adding the route to the device, where X is the number of the PPP connection, the traffic from the destination address is forwarded to the local address on the device ‘pppX’.

The information from the executed test set can be retrieved with the same kind of procedure as described earlier in the thesis, the only difference being an increased individual log file count. The parsing can be done in the same way, so the end results are deposited into a single file.

# 6 SUMMARY

While developing the automation system, there were several difficulties in controlling the devices. Stability issues in the whole system arose in the final steps of the current development cycle. The scripts have to be fine-tuned by changing the timing aspects of a command execution. This way the system robustness can be increased.

In retrospect, in the start of the development phase, a larger variety of different UEs would have helped with the stability issues. This can be carried out later on, and the UEs can be changed with only a small effect on the created scripts which control the system. Some of the current stability issues are circumvented by powering the system down and starting it up again. This does not solve the issue completely, but it allows the system to operate at an acceptable level, the downside being an increased run time for the entire test set. At the moment, the increased time needed is not an issue, but when the system will be expanded to include more tests, it needs to be in a more stable state to increase the time efficiency.

While comparing the achieved results to the ones set when this thesis was started, we achieved what we aimed at: creating an automated node feature test environment. Currently the basic test set has been implemented to a daily execution set. The development of the system still continues inside the company, but the criteria of the system functionality were met.

## REFERENCES


1. Ericsson. 2013. Facts & Figures. Date of retrieval 16.5.2014. Available: http://www.ericsson.com/thecompany/company_facts/facts_figures.

2. Android Debug Bridge. Available: http://developer.android.com/tools/help/adb.html. Date of retrieval 10.5.2014

3. USB_ModeSwitch - Handling Mode-Switching USB Devices on Linux. 2014. Date of retrieval 10.5.2014. Available: http://www.draisberghof.de/usb_modeswitch/.

4. Freecode: Wvdial. 2007. Date of retrieval 10.5.2014. Available: http://freecode.com/projects/wvdial.

5. Android Developers. 2014. Using Hardware Devices. USB Vendor IDs. Date of retrieval 2.5.2014. Available: http://developer.android.com/tools/device.html.

6. Stenlik, 2007. Nokia Developer Wiki. Packet Data Protocol (PDP) Context Activation. Date of retrieval 10.5.2014 Available: http://developer.nokia.com/community/wiki/PDP.

7. Android Developers. 2014. Android SDK. Date of retrieval 2.5.2014. Available: http://developer.android.com/sdk/index.html

8. 3GPP TS 26.204. 2012. 3GPP Specification detail, Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; ANSI-C code. Date of retrieval 10.5.2014. Available: http://www.3gpp.org/DynaReport/26204.htm

9. Wannstrom, Jeanette 2014. 3GPP Keywords & Acronyms, Maximum channel rate, Date of retrieval 10.5.2014. Available: http://www.3gpp.org/technologies/keywords-acronyms/99-hspa

10. 3GPP TS 25.104. 2014. 3GPP Specification detail. Base Station (BS) radio transmission and reception (FDD). Date of retrieval 12.5.2014 Available: http://www.3gpp.org/DynaReport/25104.htm

# APPENDICES

Appendix 1 Memo of initial data (in Finnish)

# LÄHTÖTIETOMUISTIO

Tekijä[1] Mikko Suhonen

Tilaaja[2] OY L M Ericsson AB

Tilaajan yhdyshenkilö ja yhteystiedot[3] Markku Jurmu

Työn nimi[4] Testausautomaatio RNC-ympäristössä

Työn kuvaus[5] Työn tarkoituksena on toteuttaa automaatio tukiaseman feature -testausta varten. Automaatiossa tullaan käyttämään puhelimia/päätelaitteita, joita ohjataan linux –ympäristössä toteutetuilla skripteillä.

Työn tavoitteet[6] -Testijärjestelmän rakentaminen
-Kartoittaa automaation eri vaihtoehdot

Tavoiteaikataulu[7] Tavoittena on saada järjestelmä valmiiksi 31.5.2014 mennessä.

Päiväys ja allekirjoitukset[8] 17.2.14

---

1 Tekijän nimi, puhelinnumero ja sähköpostiosoite.
2 Työn teettävän yrityksen virallinen nimi.
3 Sen henkilön nimi ja yhteystiedot, joka yrityksessä valvoo työn suoritusta.
4 Työn nimi voi olla tässä vaiheessa työnimi, jota myöhemmin tarkennetaan.
5 Työ kuvataan lyhyesti. Siinä esitetään muun muassa työn tausta, lähtötilanne ja työssä ratkaistavat ongelmat.
6 Esitetään lyhyesti ja selvästi työn tavoitteet.
7 Esitetään projektin tavoiteaikataulu. Silloin, kun työllä on välitavoitteita, myös ne merkitään aikatauluun. Tavoiteaikataulun ja oppilaitoksen yleisaikataulun perusteella tekijä laatii oman aikataulunsa.
8 Lähtötietomuistio päivätään ja sen allekirjoittavat tekijä ja tilaajan yhdyshenkilö