OULUN AMMATTIKORKEAKOULU

Ari Viitanen

TEST AUTOMATION IN AN RNC-ENVIRONMENT

TEST AUTOMATION IN AN RNC-ENVIRONMENT

Ari Viitanen
Bachelor’s thesis
Spring 2014
Information Technology
Oulu University of Applied Science

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikka, Langaton tiedonsiirto

Tekijä: Ari Viitanen
Opinnäytetyön nimi: Testausautomaatio RNC-ympäristössä
Työn ohjaajat: Kari Jyrkkä, Markku Jurmu
Työn valmistumislukukausi ja -vuosi: Kevät 2014 Sivumäärä: 33 + 6 liitettä

Opinnäytetyön tilaaja oli OY LM Ericsson AB. Työ tehtiin Ericssonin tiloissa Oulun teknologiakylässä. Ericssonin Oulun toimipiste keskittyy tutkimaan ja kehittämään tukiasemaratkaisuja WCDMA ja LTE-verkoissa.

Opinnäytetyön tarkoitus oli automatisoida tukiaseman testejä oikeassa RNC-ympäristössä WCDMA-verkossa. Lähinnä tiedonsiirtotestejä, jotka mittaavat keskimääräisiä siirtonopeuksia eri olosuhteissa ja asetuksissa.

Tämän automaation tärkein tavoite oli automatisoida osa testauksesta, jotta perustestaukseen kulutetut resurssit voitaisiin jakaa johonkin hyödyllisempään. Ja tietenkin automaatio voi tuoda aina kaivattua tasaisuutta ja parempaa regressiota testaukseen. Muita tavoitteita oli käytettyjen laitteiden dokumentointi ja testien valitseminen.

Tavoitteet saavutettiin koska valitut tiedonsiirto-, multi-RAB ja pehmeiden solunvaihtojen testit saatiin automatisoitua. Kaikkia testejä ei ole mahdollista automatisoida, mutta jo automatisoitujen lisäksi muutama lisätesti olisi mahdollinen. Opinnäytetyöhön varattuun aikarajaan näiden automatisointi ei ollut mahdollista. Työssä totetutettu testiautomaatio on otettu käyttöön päivittäisessä ohjelmistotestauksessa.

Tätä automaatiota voisi vielä viedä pitemmälle lisäämällä automaatioon signaalianalysaattorin, joka mahdollistaisi lähetystehojen mittaamisen ja testaamisen. Kestävyystestit olisi myös mahdollista lisätä automaatioon. Myös kolmannen tukiaseman lisääminen vaimentimeen toisi solunvaihtotestaukseen lisää variaatioita.

Asiasanat: RNC, automaatio, WCDMA, Ericsson

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Option Wireless Communications

Author: Ari Viitanen
Title of thesis: Test automation in an RNC Environment
Supervisors: Kari Jyrkkä, Markku Jurmu
Term and year of completion: Spring 2014 Pages: 33 + 6 appendices

This Bachelor’s thesis was commissioned by OY LM Ericsson AB. The work was done at Ericsson’s premises in Oulu. Oulu premises concentrate on research and development of base station solutions for WCDMA and LTE-networks.

The purpose of this thesis was to automate base station test cases in a real RNC-environment in a WCDMA-network, mainly traffic cases that measure average throughput in different conditions and settings.

The main aim of this automation was to automate some of the testing so that human resources that were consumed by this basic testing could be assigned to something more useful. And of course automation could bring much needed stability and better regression to testing. Other aims were to document the used equipment and choose test cases to automate.

The aims were achieved so that the chosen transfer multi-RAB and soft handover test cases were automated. All of the test cases are not possible to automate but few more should be possible. Unfortunately that was not possible in time reserved for this thesis. The used equipment was also documented. Automation is in use for daily software release testing now.

This automation could be taken further for example by attaching a signal analyzer with an Ethernet connection. This would enable antenna power measurements. Robustness testing cases are possible to add to this system. Adding a third base station to the attenuator would add many different variations for handovers.

Keywords: RNC, automation, WCDMA, Ericsson

PREFACE

I want to thank Ericsson for giving me the opportunity to be part of their great team and for providing me tools to do this thesis. Also, I want to thank all wonderful co-workers who helped and advised me in this process. Especially my supervisor from Ericsson Markku Jurmu deserves to be mentioned. Thanks to my supervising teacher Kari Jyrkkä for a prompt interaction and of course for his guidance.

May 2014, Oulu

Ari Viitanen

CONTENTS

ABBREVIATIONS

APN	Access Point Name
CS	Circuit Switched
DBM	DeciBel-Milliwatts
FTP	File Transfer Protocol
HSDPA	High-Speed Downlink Packet Access
HSPA	High Speed Packet Access
HSUPA	High-Speed Uplink Packet Access
IMSI	International Mobile Subscriber Identity
Layer 3	Network layer
NOOBS	New Out Of the Box Software
OS	Operating System
PS	Packet Switched
RAB	Radio Access Bearer
RF	Radio Frequency
RNC	Radio Network Controller
RND	Research and Development
SD	Secure Digital
SSH	Secure Shell
UE	User End Device
USB	Universal Serial Bus

WCDMA	Wideband Code Division Multiple Access
WLAN	Wireless Local Area Network

1 INTRODUCTION

OY L M Ericsson AB is an old Swedish company with a long history in telecommunications. It was founded in 1876 and in 2014 it has over 110,000 employees worldwide (1). Oulu site was founded in February 2012 and it is mainly concentrated on research and development of base stations in LTE- and WCDMA-networks. These automation tests are all performed with a base station in the WCDMA-network.

Lots of human resources were used for a very basic testing so there was a definite need for automation that runs basic tests with every software build. The company already has own automation programs but they are concentring more on stability testing with numerous phones and very long test runs. A test automation system designed and implemented in this thesis is used for a daily software release testing.

Other aims were to document the used equipment and choose test cases to be automated. Equipment and tools are documented in chapter 3 and all test cases explained in chapter 4.

As a result of this thesis the subscriber got a basis for an automation system for their daily software release testing. The development still continues and investigation for possible new cases is an ongoing process.

There are some appendices that contain confidential information and are only allowed for the subscriber's use.

2 BASE STATION SOFTWARE TESTING

In software testing regression testing is the most dulling and mind melting activity but at the same time it is very crucial part of testing. The point of regression testing is to repeat the same tests before and after a software fix. The tests can be used to confirm successful bug fixes and to verify that nothing was broken during a software change.

2.1 Concepts and terms

For better understanding of test cases mentioned in this thesis, it makes sense to explain some terms and concepts used in testing.

2.1.1 Base station

A base station connects a UE to a mobile network. The base station creates a cell that the UE connects to. There are cells of different sizes with radius size changing from a few meters to tens of kilometres (2).

2.1.2 Radio Access Bearer

A connection establishment is a 3-stage process that results in a Radio Access Bearer between the RNC and UE. The RAB provides connectivity to the UE. There are different RABs for data (PS) and speech (CS) that are chosen depending on what kind of service or information needs to be transported (3). When multiple RABs are used it is called a Multi-RAB. More information about the RAB can be found from Agilent (3) and telecomhall websites (4)

2.2 Base station software testing

Software testing can be grouped in different sections that test different features of base stations software. Some tests are performed for every software build and others only for main builds depending on what has changed from the last build. Practices in software developing vary from company to company. Companies deliver softwares at different phase for testing, some do this daily some weekly. And every now and then there are some bigger changes in a software when everything needs to be verified. Standards and requirements for different technologies, features and releases can be found from a 3GPP website (5).

2.2.1 Basic functionality

This very basic testing starts by checking that the base stations cell is functional and UEs can attach to it. After the cell functionality has been verified some basic features are tested. They can include a few phone calls and data transfers.

Locking and unlocking different features from network needs to be tested. In the WCDMA-network this could mean locking and unlocking HSPA-features.

2.2.2 Data performance

To ensure that transfer speeds are satisfactory and stable, downlink and uplink needs to be tested several times. The maximum channel rate and peak user data rate for the HSDPA and the HSUPA can be found in a 3GPP website (6).

2.2.3 Scheduling

Scheduling is a function that controls that a base station distributes its bandwidth correctly. If same category UEs is connected to the same cell the base station needs to share its resources equally to both UEs.

2.2.4 Handovers

Handover happens when a device that is connected to a mobile network is on the move and changes from a cell to another cell. In testing where device movements are restricted by a cable, handover is simulated with an adjustable attenuator by attenuating signal strength from one base station and by amplifying the signal from another base station. In radio technology the signal strength unit is dBm which means decibels relation in milliwatts. 0 dBm equals to 1 milliwatt and 30 dBm equals to 1,000 milliwatts.

There are soft and hard handovers. In soft handover the UE has established radio links into two or more cells at the same time and the serving cell is chosen by the strength of received signal. At least one radio link is kept active all the time. If cells are in the same base station, handover is called softer handover (7).

In hard handover old radio links are removed before a new link is established. A hard handover can be seamless or non-seamless. A seamless handover is not perceptible to the user (7).

2.2.5 Stability and capacity

Stability tests are long test runs that test software's stability and functionality for a specific period of time. These tests could include several UEs in performing dynamic tasks in different RABs.

In software testing of a base station the capacity testing checks that the software can handle the maximum amount of users in all RABs. The maximum number of users differs from a base station to a base station and from the RAB to the RAB.

2.2.6 Signal power settings

Signal transmission power tests check that the transmission and channel power adjusting function works. The base station needs to be connected to a signal analyzer to verify that that signal has changed the correct amount.

2.2.7 Robustness testing

Robustness testing is a term used when the software reliability is tested. For example, these tests can contain hundred software resets and hundred hardware resets. Also, testing for unexpected events like power outage during software upgrade is a part of robustness testing.

3 THE TEST AUTOMATION SYSTEM

When you do the same test cases hundred times a day and the results are almost the same all the time, it is hard to keep your mind sharp and your undivided attention at the results. This is where automation steps in with its infinite stamina and relentless judgment. You receive valid results seven days a week twenty four hours a day at least as long as automation is stable. Human intervention is still needed in case of unexpected events and failures. All tests are Layer 3 tests.

3.1 Test setup

Figure 1 presents a test system built in this project. Everything in this picture is connected to an Ericsson lab network. To an automation rack (a server, an attenuator, Raspberries and a power socket switch) a lab network connection comes through a network switch which is also located in the automation rack. In Figure 1 devices inside the red box are located in the automation rack.

The base stations that are used for handover testing are connected to a programmable attenuator with RF-cables. From an attenuator signal continues with RF-cables to a 2-to-1 combiner and then connected to shielded box where UEs are. Placing UEs to shielded box is very crucial so that signals or UEs do not interfere with each other. More information about the shielded box can be found in chapter 3.2.3.

Raspberry is connected to a server with the Ethernet and to UEs with a USB-cable. Raspberry handles the UE controlling and configuring.

Scripts are located in the automation server. Scripts are run by taking an ssh connection to the automation server with a PC through a lab network; the server then gives the commands to Raspberry with an ssh connection. Raspberry then uses these commands to control UEs. More about how scripts are run and to what Raspberry server connects can be found from chapter 4.2.

In cases where an RNC configuration is needed, the server takes an ssh connection through a lab network to the RNC and configures what is determined in a script. The attenuator configuration is also done the same way but with a telnet connection.

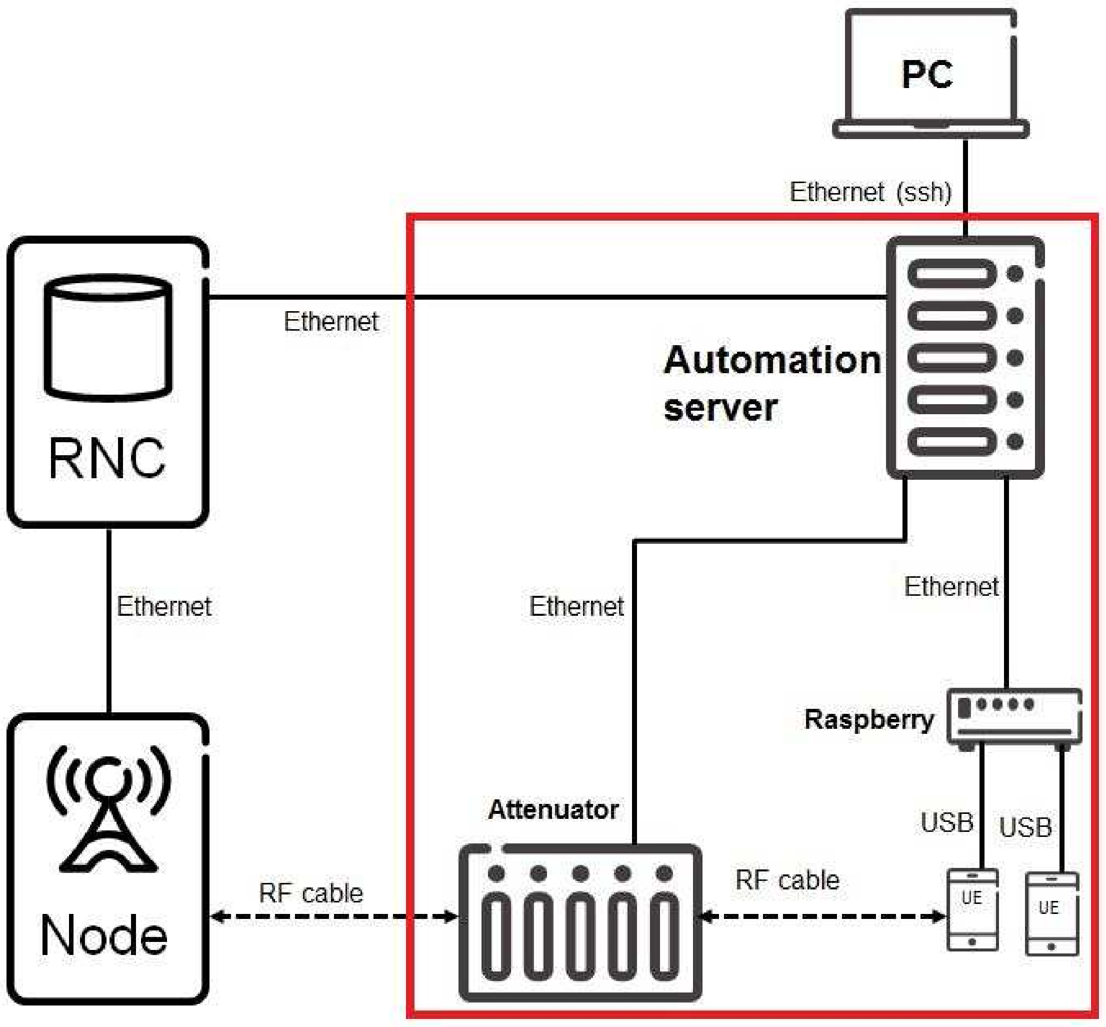


FIGURE 1. Final test setup

3.2 Equipment

When choosing equipment for this automation one of the priorities was to keep everything as simple as possible. Naturally products that offer the best value for money will always be appreciated by the one who is paying the bill.

3.2.1 Raspberry Pi

After some research Raspberry Pi running Raspbian OS (Linux) was selected as a platform for UE controlling. Raspberry was a good solution because of its price and easy operating system install. The OS is installed to a SD-card and it is easy to clone to any other Raspberry. Of course, the adjustable nature of Linux and numerous open source programs was a key factor. Linux is also quite stable after you have configured it properly. Raspberry is also a good value for money and completely silent. Raspberry’s small size can be seen in Figure 2. Raspberry takes its power from a micro-USB connection so any new mobile phone charger is suitable.

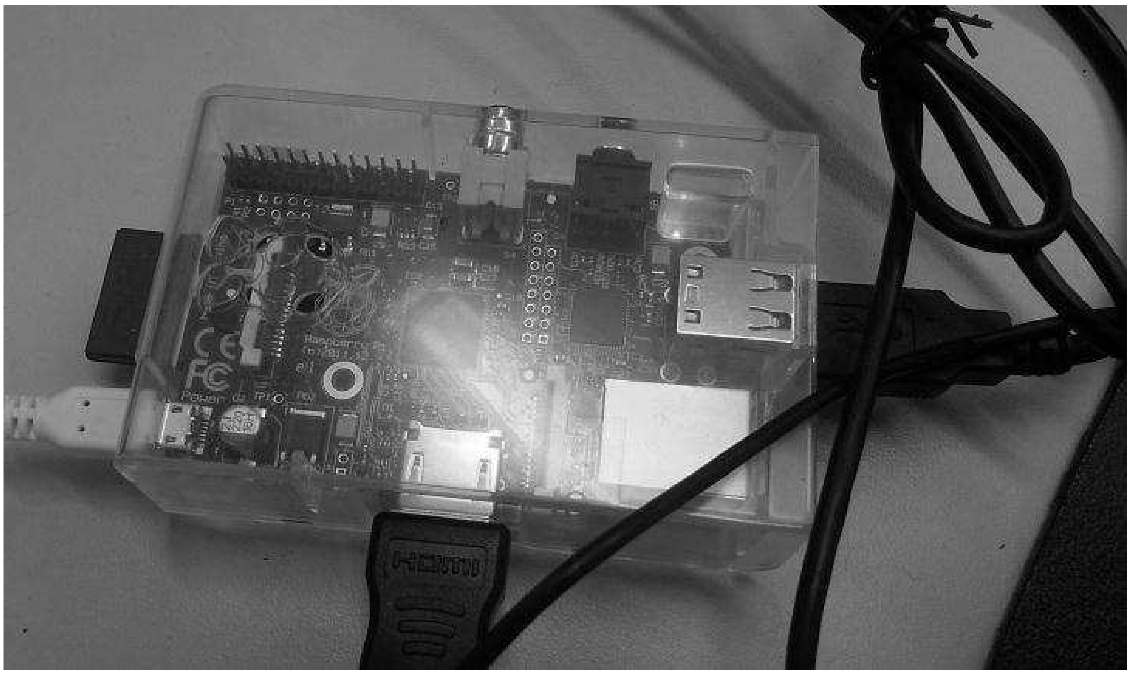


FIGURE 2. Raspberry Pi

3.2.2 User end devices

There are two kinds of UEs in this automation. Different manufacturer' wireless USB-modems were used for most of the data transfer tests because of their better and more stable data throughput.

Tests that required normal calls or multi-RAB with a call and data were performed with Android phones.

Android phones need to be rooted so that the user can control the phone with administrative rights. Rooting differs from device to device. More information of rooting can be in xda-developers message forum (8).

3.2.3 Automation rack

The heart of this automation is the rack where a server, a network switch, UEs, Raspberries, shielded boxes, a server and remote controlled attenuator and a power socket switch are all connected.

Shielded Box
This is a metal box which has pull-troughs for RF- and USB-cables. The box blocks signals from outside and RF-cables prevent UEs from connecting over-the-air to a wrong base station. RF-connections are tightened to the right moment with a moment wrench; loose connections affect signal quality surprisingly much.

Remote controlled attenuator
This is a programmable attenuator with an Ethernet port. The attenuator can be connected to a network and configured via a Telnet connection. This is needed for handover tests. One can choose how many dBms to attenuate the signal and in what time. An example would be to attenuate signal from 0 to 50, 1 dBm in every 1,000 milliseconds. The attenuator has four inputs and outputs for a radio signal.

Remote controlled power switch socket

A 8-port programmable power switch socket is useful when anything in the rack needs to be powered off. All devices are connected to this. The remote connection to a power switch happens with an Internet browser or a telnet. The IP-address can be changed.

3.3 Tools used

There are lots of free software for Linux to choose from. This can also be damaging because it can be hard to pick the tool that serves the user’s needs the best.

3.3.1 Wvdial

Wvdial is a command line program for connecting UEs with a dial-up. Automatic configuration is done with a command “wvdialconf”. This writes a default configuration to the configuration file. The user still needs to add phone number and an APN and when needed a login username and a password. Wvdial is executed with a command “wvdial”. Wvdial is run from Raspberry which is connected to UEs with a USB.

An example of a wvdial configuration file:

```
[Dialer Defaults]

Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Init3 = AT+CGDCONT=1,"IP","apn_name"
Username = ''
Password = ''
Phone = *99#
Modem Type = USB Modem
Modem = /dev/ttyUSB0
Baud = 460800
New PPPD = yes
ISDN = 0
```

More information of AT-commands can be found from computerhope.com (9) and m2msupport.net (10) websites.

Wvdial allows the user to create a custom configuration inside the configuration file. This is handy when there are multiple devices that need to be connected. This custom configuration is done by editing and adding lines to the file. Like in this example:

```
[Dialer USB1]
Modem = /dev/tty/USB1

[Dialer HEP0]
Modem = /dev/tty/ACM0
Phone = *666#
```

Now when a device in the USB1 port needs to be connected, the command “wvdial USB1” is used and when connecting a device in ACM0, the command is “wvdial HEP0”. In USB1 configuration these custom lines overwrite the “Modem” line in a default configuration. In HEP0 configuration lines overwritten lines are “Modem” and “Phone”. Other values used are from the default configuration. When the command “wvdial” is given, all values are from the default configuration.

3.3.2 USB-modeswitch

Usually when a wireless USB-modem is plugged to a Windows computer the computer recognizes the device as flash drive. This is an installation partition of the device from where you can install required drivers. After installation the device boots automatically from the functional partition that enables connectivity.

In Linux the switch from an installation to a functional partition is done with this tool. All USB-modems used in this automation were recognized by a USB-modeswitch and switched the partition automatically.

3.3.3 ADB

Android Debug Bridge is a command line tool used to control Android devices remotely. In this environment the control is done with a USB connection. And this USB connection is used for speech calls and device reboots. If multiple devices are connected, the difference between devices is done with a serial number.

In Raspberry Pi a normal ADB does not work, instead Raspberry uses ADB_RPI which is a ported ADB so it uses the same commands and does eve-rything that a regular ADB does.

4 ASSEMBLY

After choosing proper equipment, the next step was to assemble the test environment. Prior to a final assembly a proof of concept that Raspberry is capable of UE controlling was needed.

4.1 Development

It was decided that the development should happen only with one Raspberry and a couple of UEs. Since there was an environment for over-the-air testing and only change in this environment would be just adding Raspberry for the UEs controlling, it was a perfect match. Figure 3 presents the development setup where the RNC, base station and PC are connected to the Ericsson lab network. The PC is used only for configuring the RNC and base station. Raspberry is configured locally with a keyboard and mouse. There is a network route to the FTP-server from a mobile network so that data transfer happens from and to this server.

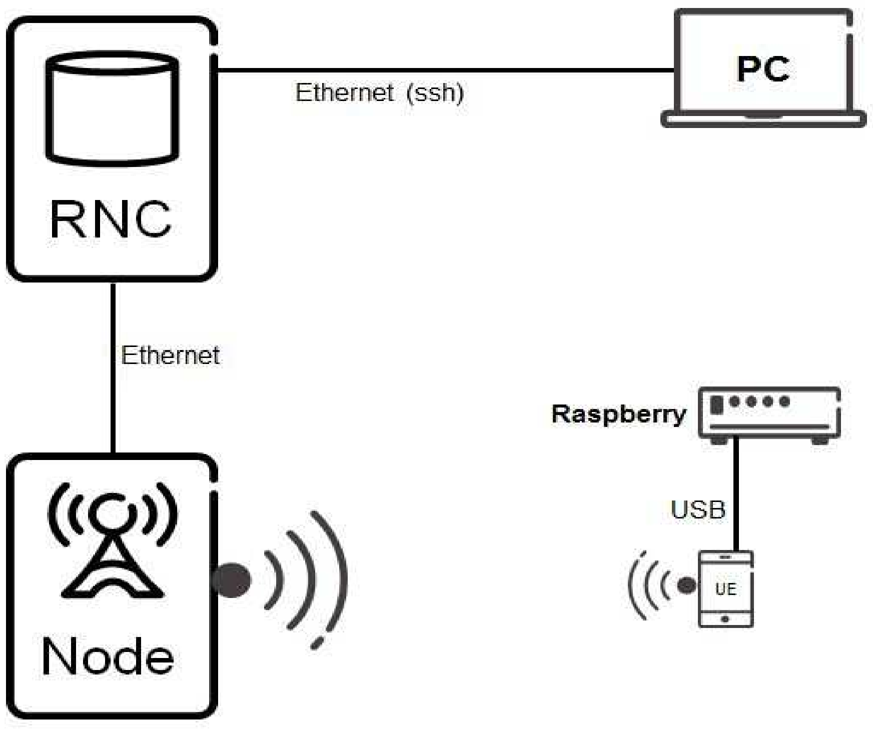


FIGURE 3. Development setup

The first step was to get Raspberry Pi running. Using a NOOBS operating system install manager made this OS installing very easy. A guide for NOOBS can be found from Raspberry Pi website (11). The NOOBS includes many operating systems and as recommended on the website, Raspbian was selected.

After installing Raspbian, the first problems occurred when UEs needed to be connected to Raspberry. There were already a keyboard and a WLAN-modem connected to only two USB-ports of Raspberry. This problem was solved with a multiport USB-hub. For a sufficient amount of power, it needed to have an external power. The exact required milliamperes were not solved, but a 4-port USB-hub that takes its power just from a single USB-port (500 milliamperes) was not enough.

After power and lack of port problems were solved, the next problem appeared. Raspberry recognized that the USB-modem was connected but it did not recognize it as a modem but as a flash drive. A research revealed that Linux has problems with wireless USB-modems that have a separate install partition with Windows drivers in it. Linux has a tool for this problem called a usb-modeswitch which was introduced in chapter 3.3.2. Downloading and installing all tools and programs was done with a Linux packet manager AptGet.

The next phase was to establish a dial up connection with the USB-modem and Android phone. The investigation before assembly revealed that the right tool for the job would be wvdial. Wvdials usage is simple and it is explained in chapter 3.3.1.

After the dial up connection, a data transfer test was next. The first connection attempt to an ftp-server immediately brought up one problem - “no route to host”. This was fixed by adding an ip route from device to the ftp-server. After this route transfers ran smoothly.

The last thing to test was cs calls. The first technique that was tested included Ericsson’s internal tool for Android phone controlling, but this proved to be too complicated and unstable and it lacked in some features that were needed in other tests. The second technique to test was AT commands. AT commands are sent straight to a modem and this was working great when just calling a phone call. But after the dial up was connected the modem did not accept AT commands any more. This is because the dial up reserves devices modem for its use. The third technique was the ADB which is explained in chapter 3.3.3. The ADB proved to be very reliable for phone calls and the multi-RAB was not a problem.

After these fundamental functions were confirmed to work, it was time to start putting the test cases on script. More of scripts and test cases can be found in chapter 5.

An attenuator, or attenuator control scripts, did not need much development because same attenuators were already in use in other environments. Adding these control commands was just a matter of adding a few lines to scripts. The attenuator was added to the final setup, not to this development setup

Another device that was added later to the automation system was a server. These servers are also in use in other environments and adding one to the automation system was just a matter of configuration, moving scripts from Raspberry to server, routing needed to be done and an ssh connection had to be established.

This development setup was just used to test Raspberry's tools and performance and same time in this process UEs performances were but to test.

Problems after assembly

There were a few problems that emerged from time to time. One was the connection between Android phones and Raspberry. Sometimes phones just lock up its modem and wvdial does not recognize the phone's modem. The source of this problem has not been found yet but there are some educated guesses. The most used explanation is that the previous dial up connection stays enabled after the dial up is disconnected. A fix for this is a reboot for Android and the ADB provides remote reboot.

Another problem occurred when UEs and Raspberry have been idling for a long time. Sometimes Raspberry seems to shut down itself and sometimes USB-devices are not recognized. The fix for this is also rebooting the Raspberry or if this is not possible, powering off and powering on fixes the problem. In the final setup this problem was solved by using an ssh-connection for shutting down and a remote controlled power socket for powering off the Raspberry.

Raspberry’s limited performance came up when the number of UEs was increased. Two to three UEs are fine but when six UEs were downloading with full speed in the HSPA it was too much to handle. There are also differences in UE’s performances, generally phones do not have the same kind of stability and speed as USB-modems.

4.2 Final setup

There are some differences in the final and development setup. UEs connect to a base stations cell with RF-cables and there are multiple Raspberries connected to the server. An attenuator was added between two base stations and UEs and a remotely controlled power socket switch was attached. Anything that might need booting up was connected to this power socket switch.

The same kind of attenuator was already in use in other environments so adding the attenuator to the system was not a problem. Adding lines to scripts from the already existing attenuator control scripts and plugin in the cables and it was ready to go.

Before putting this system online, it needed to be decided how the Raspberries were controlled and where the scripts were executed. There were two options for Raspberry controlling. The first was to continue like in the development setup and keep the scripts in Raspberry and to connect Raspberries straight to the lab network. Another method was the one that Ericsson had already used in another project and it was proved to be a good. In this method the user connects to a server with different usernames that had configuration file in their /home/user folder that has information about one test place. To be more exact, the configuration file has IP-addresses of everything; Raspberry, a power socket switch, a base station, the ftp-server and so on, as well as serials of Android phones that are connected to Raspberry. So in this "user connect" way each user can only control one test place. Scripts running from the server take this information and use it to control correct devices. All the UE controlling is still done from Raspberry but commands come from the server. In this method the logs are saved to the server and this saves some resources from Raspberry, so it was a logical choice to use in the final setup. The scripts and method are Ericsson internal information and more detailed description is not possible.

5 TEST CASES

The tests in automation are executed with bash scripts from Raspberry but they will work on any device that understands Unix-based commands. In some test cases features need to be locked or unlocked from the RNC. The RNC configuration is done with the internal Ericsson tool and Ericsson RNC-commands. All scripts in this chapter are from the development setup.

5.1 Traffic tests and scripts

These tests measure data speeds of the base station. The speed varies according to features are enabled from the RNC. From consumers point of view these speeds are important.

Dial-up and add route

Before any data tests a device need to be connected with a dial up connection and after establishing the connection, a route to the FTP-server is added. Old connections need to be closed and processes are killed. For a device recognition a "wvdialconf" command is used. In the development setup Android phones were always recognized as ACM (/dev/ttyACMx) devices and USB-modems as USB devices. So it was easy to determine what device to use. More of usage wvdial can be found in chapter 3.3.1.

After connection, adding an ip route to the ftp-server is needed. The command “ip route add *ip.ftp.server/mask dev ppp0*” does this. Scripts and more information of this can be found in appendix 2.

Download

This script is henceforth called “ftp_dl.sh”. The first thing to do is to login to the FTP-server. Then the user needs to choose the folder from where to download the remote file. The download is triggered with a command "get *fil*e". This test tests the base station download speed; the results are displayed in kilobytes per second (kB/s). The script and information can be found in appendix 3

Upload

In upload the user has to determine what local file to upload and to what remote folder. Parameters and arguments are determined with the same logic as in “ftp_dl.sh”. This script is henceforth called "ftp_ul.sh". This test tests the base station upload speed; the results are displayed in kilobytes per second (kB/s). The script can be found in appendix 4.

Download and upload simultaneously

This test tests the base station download and upload simultaneous transfer speeds, the results are displayed in kilobytes per second (kB/s). A simultaneous transfer can be done with the help of Linux command line feature that can send processes to run in background. This is done by adding an &-sign after the command. In this case the command “./ftp_dl.sh & ./ftp_ul.sh” runs both scripts at the same time. In script it looks like this:

```
#!/bin/bash
./ftp_0_dl.sh loops host.ip.address download_file &
./ftp_0_ul.sh loops host.ip.address upload_file
```

5.2 Multi-RAB tests and scripts

The same tests as in chapter 5.1 but with an active conversational RAB (phone call). In an Android phone, a call can be triggered with ADB command keyevents (12). The parameter and argument logic are the same as in “ftp_dl.sh". This script is henceforth called “call.sh”. The script can be found in appendix 5.

The same method for simultaneous running can be used as in “Download and upload simultaneously”. In this example a call and download are executed sim-ultanoeusly:

```
#!/bin/bash
./call.sh loops sleep_seconds phone_number &
./ftp_dl.sh loops host.ip.address download_file
```

5.3 Soft handover tests

The same tests and scripts as in chapter 5.1 and 5.2 but this time a signal comes from two base stations through a programmable attenuator and are then combined to one RF-cable. The attenuator can be controlled via a telnet connection with commands defined by the manufacturer. For example one attenuator could fade from 10 dBm to 30 dBm and at the same time another on could gain from 30 dBm to 10 dBm. In this automation all handovers are soft handovers.

From scripting point of view a few additional lines are needed when fade and gain commands are given to the attenuator. These commands determine the total amount of fade or gain, how many dBms to change in one step and how long does the attenuator wait between steps. If roughly simulating a person's walking speed, this could for example be a change from 10 dBm to 40 dBm in 1 dBm step in every 1,000 milliseconds. The change of dBms needs to be from 40 dBm to 10 dBm in another attenuator. These commands are displayed in appendix 6 which is only viewable by the subscriber of the work.

5.4 Result logging

Linux FTP prints out a good log when verbose is enabled with a "-v" parameter):

```
Connected to x.x.x.x
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: ari_auto_x.bin remote: ari_auto_x.bin
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for ari_auto_x.bin (xxxxxxx bytes).
226 File send OK.
```

```
xxxxxxx bytes received in xx.xx secs (xxxx.x kB/s)
```

The user can print this to a file by adding ">> log_file.log" after the initial ftp command. And with the tool "grep" from this file it is easy to print only lines that, for example, include the string "kB/s". This happens with the command "grep "kB/s" log_file.log" and if wanted, the print can be saved to file with ">> grep_log_file.log". And using preferred Linux editors user could get only numbers from this "grep_log_file.log" file.

In test automation these average speeds are the basis of determining if the test passes of fails by comparing these average speed results to the average speed defined before. If test average speeds are lower than this number the test fails. If it is higher the test passes. Criteria vary from case to case. Also in cases where there are hundreds of transfers the user could calculate the average of every "*n"* transfer. It depends on the case how precise results are needed.

In multi-RAB and conversational call cases the verification of conversational call is done by taking a print out from the RNC to log. From this print the user can see that the UE is connected to the right base station, IMSI number for identification and that the UE is in correct RAB. In handover cases the same print is used to verify that UE's context number is the same before and after changing cells. The context number changes if the UE needs to establish new connection.

Failed tests are performed again and if tests fail again, then it is up to test to check all the logs for errors. If there are no errors then tests need to be performed manually to verify if the problem is in the software, environment or UE. If an error is found from the software, the tester creates error ticket and the developer fixes the problem.

6 SUMMARY AND CONCLUSIONS

The main aim of this thesis was to automate base station test cases in a real RNC-environment. Choosing cases to automate was determined with the subscriber, OY LM Ericsson Ab, and this selection process was also one of the aims in this thesis. The third goal was to document used equipment's and tools.

The planned cases were traffic, multi-RAB and soft handover tests and all of these were automated. The used equipment and tools were documented and automation is in daily use in the software release testing.

There are still same problems with UEs that there were in the development. Sometimes Android phones lock their modem and a reboot is needed. And sometimes USB-devices or a port seems to go to "sleep", especially after a long period of idling. In both of these problems some relief was offered by a remote controlled power switch socket. By powering off and powering on devices before test set devices work more stable. Some human error related problems have occurred too. Sometimes devices are connected to a wrong port, sometimes the RF-cable is unplugged, a wrong base station is connected to shielded box etc.

Test automation system can be enhanced by adding a signal analyzer that could be remotely controlled and monitored to a base station and either to a lab network or to an automation server. This would enable a signal output power testing.

Another thing could be attaching a third base station to the attenuator with a different channel number and frequency. This would enable hard handovers (also known as an inter-frequency handover). The attenuator has four inputs and outputs so this could be easily done.

REFERENCES

1. LM Ericsson. *Facts & Figures*. Date of retrieval 11.4.2014. http://www.ericsson.com/thecompany/company_facts/facts_figures

2. JPL's Wireless Communication Reference Website. *Cellular Telephone Networks: Cell sizes*. Date of retrieval 14.5.2014. http://www.wirelesscommunication.nl/reference/chaptr04/cellplan/cellsize.htm

3. Agilent. *3G Networking Protocols: The Bridge Between the Air Interface and the UTRAN*. Date of retrieval 19.5.2014. http://www.agilent.com/cm/wireless/pdf/3GUTRAN.pdf

4. TelecommHall. Posted by leopedrini. May 20, 2013. *What is RRC and RAB?* Date of retrieval: 19.5.2014. http://www.telecomhall.com/what-is-rrc-and-rab.aspx

5. The 3rd Generation Partnership Project (3GPP). *3GPP specifications*. Date of retrieval 14.5.2014. http://www.3gpp.org/specifications

6. The 3rd Generation Partnership Project (3GPP). *Keywords & Acronyms, HSPA*. Date of retrieval 15.5.2014. http://www.3gpp.org/technologies/keywords-acronyms/99-hspa

7. Umtsworld.com. *UMTS Handover*. Date of retrieval 15.5.2014. http://www.umtsworld.com/technology/handover.htm

8. xda-developers. Date of retrieval 13.5.2014. http://forum.xda-developers.com/

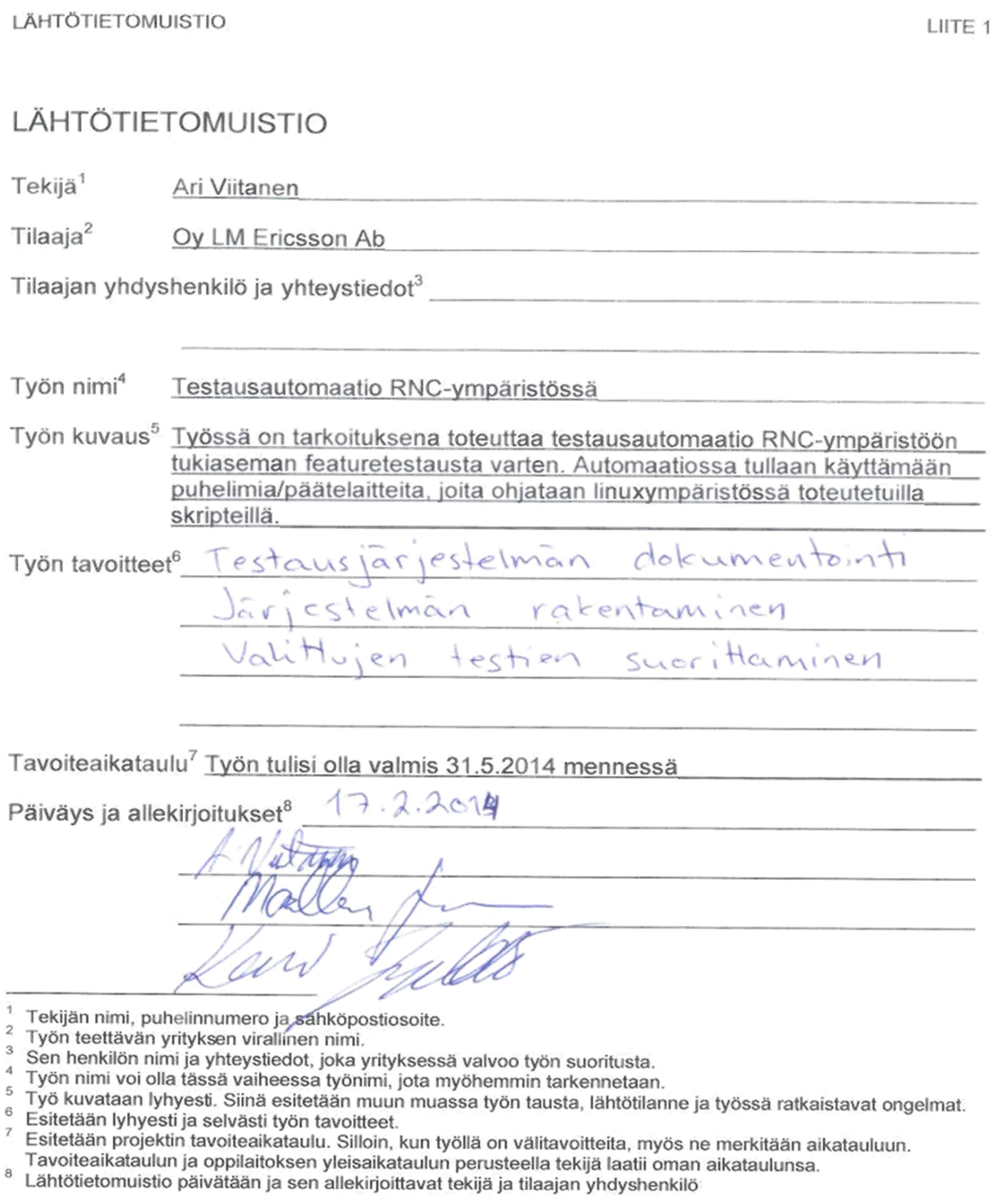
9. Computer Hope. *Basic Hayes Modem AT string*. Date of retrieval 12.5.2014. Available at: http://www.computerhope.com/atcom.htm

10. M2MSupport.net. Date of retrieval 12.5.2014. http://m2msupport.net/m2msupport/atcsq-signal-quality/

11. Raspberry Pi foundation. *NOOBS Setup*. Date of retrieval 16.5.2014. http://www.raspberrypi.org/help/noobs-setup/

12. Simulating keypress events on Android. Date of retrieval 15.5.2014. http://thecodeartist.blogspot.fi/2011/03/simulating-keyevents-on-android-device.html

APPENDICES



LÄHTÖTIETOMUISTIO LIITE 1

LÄHTÖTIETOMUISTIO

Tekijä[1] Ari Viitanen

Tilaaja[2] Oy LM Ericsson Ab

Tilaajan yhdyshenkilö ja yhteystiedot[3]

Työn nimi[4] Testausautomaatio RNC-ympäristössä

Työn kuvaus[5] Työssä on tarkoituksena toteuttaa testausautomaatio RNC-ympäristöön tukiaseman featuretestausta varten. Automaatiossa tullaan käyttämään puhelimia/päätelaitteita, joita ohjataan linuxympäristössä toteutetuilla skripteillä.

Työn tavoitteet[6] Testausjärjestelmän dokumentointi
Järjestelmän rakentaminen
Valittujen testien suorittaminen

Tavoiteaikataulu[7] Työn tulisi olla valmis 31.5.2014 mennessä

Päiväys ja allekirjoitukset[8] 17.2.2014

[illegible]
[illegible]
[illegible]

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

“killall wvdial” ends any possible earlier wvdial connections, “wvdialconf” detects connected UE’s and “wvdial wanted_dialer” connects to the wanted device.

“ip route add *ip.ftp.server/32 dev ppp0*” adds a route to a FTP-server for device ppp0 where 0 means the first connected device - number 1 would be the second device connected. “/32” means a network mask. Arguments “$1”, “$2” etc. could be used with this script to define any of these parameters. An example of a script:

```
#!/bin/bash
killall wvdial
sleep 10
wvdialconf
sleep 5
wvdial wanted_dialer & disown
sleep 60
ip route add ip.ftp.server/32 dev ppp0
```

An FTP login is done with a command “ftp –inv host_ip” in which ‘-i’ turns off interactive prompting, “-n” restrains the FTP from attempting an auto login and “–v” enables a verbose and progress. The username is determined in a parameter “USER” and a password in parameter “PASS”. Number of repetitions is determined with a parameter “END=$1”. “$1” means the first argument after a bash script command and “$2” is the second argument and “$3” is the third argument.

After login the user determines what remote file to download and from what folder. The folder is changed with a command “cd download_folder_name” and a download is triggered with command “get file_name”. In this particular script the user could download a file “download_file” hundred times from the host “ftp.ip.address” with a command “./ftp_dl.sh 100 ftp.ip.address download_file”. A script looks like this:

```
#!/bin/bash
START=1
END=$1
HOST=$2
USER=username
PASS=password
for (( c=$START; c<=$END; c++))
do
timeout 1200 ftp -inv $HOST << EOF
user $USER $PASS
cd download_folder_name
get $3
bye
EOF
Done
```

Almost the same as downloading. Command “cd upload_folder_name” changes the folder and “put file_name” uploads the local file.

```
#!/bin/bash
START=1
END=$1
HOST=$2
USER=username
PASS=password
cd local_folder
for (( c=$START; c<=$END; c++))
do
timeout 1200 ftp -inv $HOST << EOF
user $USER $PASS
cd upload_folder_name
put $3
bye
EOF
Done
```

A conversational call is started with a command “adb_rpi shell service call phone 2 s16 "$3"” and you hang up with “adb_rpi shell input keyevent 6”. The rgument “$2” determines how many seconds a script waits between a call start and hang up. The phone number is inputted in argument “$3”.

```
#!/bin/bash
START=1
END=$1
for (( c=$START; c<=$END; c++))
do
/android/adb_rpi shell service call phone 2 s16 "$3"
sleep $2
/android/adb_rpi shell input keyevent 6
done
```

CONFIDENTAL