Android Mesh Networks

A Degree's Thesis
Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by
Aleix Sala Bach

In partial fulfilment
of the requirements for the degree in
Telematic ENGINEERING

Advisor: Josep Paradells

Barcelona, June 2014
Abstract

The main idea of this project is to figure out how we can create a mesh network in Android smartphone in order to take advantage of the benefits of this kind of networks. Mesh networks has some special characteristics that could improve the user experience or even save lives in some extreme situations. For example in some situation where the density of people collapse the regular network, football stadiums or big demonstrations, or when some big catastrophe happens, tornados, tsunamis, earthquakes that take down the network, mesh networks would allow us to make some applications that make the people able to communicate with the people that is near to them.

In order to develop the entire project the Android Stick MK808B had been chosen as it fulfilled all the requirements for being able to create a mesh network. Apart from the technical specifications the cost of the product had also been taken in account. Cheap devices were bough in order to make possible to buy many of them and check the network working. To create the mesh network SPAN framework was used. SPAN framework allowed us to create a mesh network using OLSR in a few android devices, so after some modifications SPAN framework was able to create the mesh network in mk808b devices too. All modifications were made at Application Level as no suitable device had the Kernel Source published. Because of this, it is no possible at the moment to build any application that uses Mesh Networks in all android devices. The main problem is that Android doesn’t allow Ad-hoc mode itself and this makes impossible to create an application that creates a mesh network for any device without modifying and flashing a kernel for each one.

Mesh networks could be really useful in everyday or extreme situation with smartphone. But as long as Android doesn’t support it and make it possible to work with ad-hoc mode it is no possible to create an application that uses it without modifying the kernel of every device, which is impossible to be done for a regular user.
Resum

La idea principal d'aquest projecte és esbrinar com podem crear una xarxa mesh en un Smartphone Android i així poder aprofitar els avantatges d'aquest tipus de xarxes. Les xarxes mesh tenen unes característiques que podrien millorar dràsticament l'experiència de l'usuari i fins i tot salvar vides en situacions extremes. Per exemple, en una situació on la densitat d'usuaris faci col·lapsar la xarxa telefònica convencional, estadis de futbol, grans manifestacions, o quan passa alguna gran catàstrofe, tornados, tsunamis, terratrèmols, les xarxes mesh ens permetran fer aplicacions que permetin a les persones comunicar-se.

Per al desenvolupament de tot el projecte he triat l'Android Stick MK808B perquè compleix amb tots els requisits necessaris per ser capaç de crear una xarxa mesh. A més de les especificacions tècniques el cost del producte també s'ha tingut en compte. S'han buscat dispositius barats per que fos possible comprar varis d'ells i així comprovar la xarxa en un gran nombre de dispositius. Per crear la xarxa mesh vaig utilitzar el SPAN framework. El SPAN framework ens permet crear una xarxa mesh utilitzant OLSR en uns pocs dispositius Android, així que després de realitzar algunes modificacions al framework, aquest va ser capaç de crear la xarxa mesh en els dispositius mk808b. Totes les modificacions es van fer a nivell d'aplicació ja que no vaig trobar cap dispositiu que tingués el codi font del kernel obert. A causa d'això, no és possible crear una aplicació que utilitzi xarxes Mesh en qualsevol dispositiu Android. El principal problema és que Android no permet el mode Ad-hoc i això fa impossible crear una aplicació que funcioni per a qualsevol dispositiu sense necessitat de modificar i gravar un kernel per a cadascun.

Les xarxes mesh poden ser molt útils tant en situacions quotidianes com en situacions extremes, però mentre Android no suporta el mode ad-hoc no serà possible crear una aplicació que aprofiti els avantatges de les xarxes mesh sense modificar el kernel de cada dispositiu, cosa impossible de fer per a un usuari normal.
Resumen

La idea principal de este proyecto es averiguar como podemos crear redes mesh en un smartphone Android y así poder aprovechar los beneficios de este tipo de redes. Las redes mesh tienen unas características que podrían mejorar la experiencia de usuario y incluso llegar a salvar vidas en situaciones extremas. Por ejemplo, en una situación donde la densidad de usuarios haga colapsar la red telefónica convencional, como estadios de futbol, manifestaciones o cuando ocurren catástrofes como tornados, tsunamis, terremotos, las redes mesh nos permiten crear aplicaciones que permitan a las personas comunicarse con la gente cercana a ellos.

Para el desarrollo del proyecto he elegido el Android Stick MK808B porque cumple con todos los requisitos para ser capaz de crear una red mesh. Además de las especificaciones técnicas, el coste del producto también se tomó en cuenta. Se han buscado dispositivos económicos para que fuera posible comprar un número elevado de ellos y así poder hacer experimentos. Para crear la red mesh se utilizó el SPAN framework. Span Framework nos permite crear una red mesh utilizando OLSR en unos pocos dispositivos Android, así que después de algunas modificaciones en el framework también fue capaz de crear una red mesh en los mk808b. Todas las modificaciones se hicieron a nivel de aplicación ya que ningún dispositivo que cumpliera los requisitos para el proyecto tenía el código del kernel abierto. A causa de esto, no es posible crear una aplicación que aproveche las redes mesh en cualquier dispositivo Android. El principal problema es que Android no permite el modo Ad-hoc y eso hace imposible crear una aplicación que funcione en cualquier dispositivo sin necesidad de modificar ni grabar un kernel para cada uno.

Las redes mesh pueden ser muy útiles en situaciones cuotidianas o extremas, pero mientras Android no de soporte el modo Ad-hoc no será posible crear una aplicación que utilice este tipo de redes sin modificar el kernel para cada dispositivo, lo que es imposible de hacer para un usuario normal.
Acknowledgements

I have received assistance from Josep Paradells as he was my project director. He has given me advised during the project development, all the materials I needed and a workplace to develop the project.

Apart from Josep I have also received advise from the fellows of Bottom up Broadband as my TFG was one of the BuB projects. Bub is part of the Common4Europe project that is financed by the European Union. I want to thank all the support that I have received from UPF and the fellows from the other TFG during the weekly meetings. Also I want to thank the opportunity that I had to travel to Leipzing, Germany to assist to the Battelmesh, where the main mesh protocol developers gathered for a week to discus and test different mesh protocols. Whit my assistance I could get answered a few questions and get some advice on the project and also talk to people that worked in similar projects and share our experience.
# Revision history and approval record

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20/01/2014</td>
<td>Document creation</td>
</tr>
<tr>
<td>1</td>
<td>15/03/2014</td>
<td>Document revision</td>
</tr>
<tr>
<td>2</td>
<td>20/06/2014</td>
<td>Document revision</td>
</tr>
</tbody>
</table>

## DOCUMENT DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>Name</th>
<th>e-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aleix Sala Bach</td>
<td><a href="mailto:aleix@salabach.cat">aleix@salabach.cat</a></td>
</tr>
<tr>
<td>Josep Paradells</td>
<td><a href="mailto:josep.paradells@entel.upc.edu">josep.paradells@entel.upc.edu</a></td>
</tr>
</tbody>
</table>

Written by:  
Date: 20/06/2014  
Name: Aleix Sala Bach  
Position: Project Author

Reviewed and approved by:  
Date: 14/10/2014  
Name: Josep Paradells  
Position: Project Supervisor
# Table of Contents

**ABSTRACT** .................................................................................................................. 1  
**RESUM** .......................................................................................................................... 2  
**RESUMEN** ....................................................................................................................... 3  
**ACKNOWLEDGEMENTS** ................................................................................................. 4  
**REVISION HISTORY AND APPROVAL RECORD** ......................................................... 5  
**LIST OF FIGURES** ......................................................................................................... 7  

1. **INTRODUCTION** ........................................................................................................... 8  
2. **STATE OF THE ART OF THE TECHNOLOGY USED OR APPLIED IN THIS THESIS:** ..... 9  
   2.1. Mesh Networks ........................................................................................................... 9  
   2.2. Routing in MANET ..................................................................................................... 10  
   2.3. Applications of MANET ........................................................................................... 12  
   2.4. Span Framework ........................................................................................................ 13  
3. **METHODOLOGY / PROJECT DEVELOPMENT:** ......................................................... 15  
   3.1. Finding the Proper Device ........................................................................................ 15  
   3.2. Preparing the Workspace .......................................................................................... 16  
   3.3. Building the Kernel ................................................................................................... 16  
   3.4. Span Framework ........................................................................................................ 17  
   3.5. Rooting MK808B ....................................................................................................... 19  
   3.6. Testing Span Framework ............................................................................................ 22  
   3.7. Building Chat Application ......................................................................................... 23  
   3.8. Testing Mesh Chat ..................................................................................................... 25  
4. **RESULTS** ...................................................................................................................... 26  
5. **SOCIAL IMPACT** ......................................................................................................... 29  
6. **BUDGET** ...................................................................................................................... 30  
7. **CONCLUSIONS AND FUTURE DEVELOPMENT:** .................................................... 31  
**BIBLIOGRAPHY** ............................................................................................................. 32  
**GLOSSARY** ..................................................................................................................... 33
List of Figures

Each figure in the thesis must be listed in the “List of Figures” and each must be given a page number for its easy location.

Figure 1 – MANET Network
Figure 2 – MANET protocol behaviour
Figure 3 – Android network stack after SPAN Framework injection
Figure 4 – DeviceConfig file
Figure 5 – Function to detect on which device is working the framework
Figure 6 – Update.zip structure
Figure 7 – CWM files in to the Update.zip
Figure 8 – Updater script
Figure 9 – Android recovery
Figure 10 – CWM recovery
Figure 11 – OLSR information table
Figure 12 – Eclipse IDE
Figure 13 – Eclipse GUI design
Figure 14 – OLSR Hello message
Figure 15 – App splash screen
Figure 16 – Available client list activity
Figure 17 – Message creation activity
Figure 18 – Sliding menu
Figure 19 – Application options
1. **Introduction**

The main goal of this project is to study if it’s possible to create a mesh network with Android smartphones as android doesn’t support ad-hoc mode by itself and see if we can take advantage of this network topology called mesh. Once how to enable ad-hoc mode in android is figured out an android applications will be programmed in order to create a network and enable users to send plain text messages between them using a mesh network.

As the project is started from scratch, first of all I need to study what is a mesh network, and how the kernel can be modified to enable this non-supported feature. Later on due to the impossibility to have an Android Kernel Source with a working driver for the network chipset I will change how the ad-hoc mode is enabled. To solve this problem SPAN framework will be use, as it enables ad-hoc mode in Application level and doesn’t requires modifying the kernel at all. Due to the lack of a working Android kernel source some changes had been done to the initial work plan of the project.

SPAN framework is an open source framework. It was started at MITRE organization but then the main developer got out and brought the framework to the community as an open source project. Today we can find some developers working on SPAN framework on github, and many other developers working on android applications using mesh through SPAN project.

A part form the kernel problem, I had some troubles with the first devices I intended to work with as were totally close a no modification could be done. Because of that I had to search for another suitable devices and it delayed the initial work plan.
2. **State of the art of the technology used or applied in this thesis:**

2.1. **Mesh networks**

A Mesh network is a kind of network topology in which every node cooperates in the distribution of the data. A mesh network is one type of Ad-hoc network and using this technology with mobile nodes, smartphones or other android devices, make this network to become a MANET (Mobile Ad-Hoc Network). MANET networks have the typical problems of ad-hoc network but also have to deal with the problems introduced by the mobility of the nodes. Two different techniques can be used in a mesh network, flooding or routing.

In this project we are using the routing technique as is much more efficient than a typical flooding technique and in mobile devices the power consumption has to be taken into account.

Mesh networks that use the routing technique work by defining paths along the net and sending the message hope by hope until the destination is reached. As we said before MANET networks have to deal with some problems because of the wireless and because of the mobility of the nodes. To solve some of this problems there are self-healing algorithms, that differ depending which protocol is used, that allow continuous connections and reconfigurations of broken paths in order to ensure availability to all the paths in the network. So a MANET network is a continuously self-configuring, infrastructure-less network of mobile devices connected without wires.

As all the devices in a MANET network are mobile, they can move to any direction and will therefore change its links to other devices frequently.

![MANET Network](image)

**Figure 1 – MANET Network**

In the Figure 1 we can see a typical MANET network, and how the node MH2 moves from one position to another. As we can see when MH2 reaches its final position it has no sense to maintain the previous path through MH3, maybe it is not even possible because of the distance. Here is when we see the need of the update an reconfigure links frequently, if it is not done very frequently it can take a long time to notice MH2 has changed its position and it broke the link through MH3 and a lots of packets can be lost.
In this particular case the risk of losing packets is not that much as MH2 is at one extreme of the network, but if for example was MH5 who moves to another location and the network won’t detects and reconfigures in a short time we can have 2 separated networks and much more traffic I going to be lost.

Security is an issue to this kind of networks at the moment. None of the existing protocols analysed by PGP have made a decent trade off between security and performance.

### 2.2. Routing in MANET

Mobile ad hoc networks are characterized by a multi-hop network topology that may change frequently due to mobility. Efficient routing protocols are needed to establish communication paths between nodes, without causing excessive control traffic overhead or computational burden on the power constrained devices.

One of the major challenges in designing protocols for MANET networks is that nodes needs to know at least the reachability to its neighbours in order to calculate the packet route, but as we have seen, the network topology change quite frequently. In addition as the number of devices can be large, finding a route to the destination requires a large and frequent amount of traffic, furthermore as the nodes are mobile it increases even more the control traffic and this can lead to a non-bandwidth left for data packets.

Mainly three types of routing protocols can be found:

**Proactive or Table-driven** protocols mandates that all nodes should keep track of all possible destinations, so when the nodes needs to send a packet it already knows the route where the packed needs to be sent. In order to accomplish this, all nodes send periodically its routing tables through the network.

In one hand it has some advantages as it has practically no delay when it needs to forward a packet but it also has some drawbacks as it consumes a lot of bandwidth to keep all routes updated and also maintain routes that are never used.

**Reactive** protocols unlike the proactive protocols employ a lazy approach as nodes only discover the routes when these are needed and also only maintain them for the time the route is used. It has much lower overhead than proactive, but it has also drawback as it has a significant delay time as it has to discover the new route. Because if this it floods the network in order to find the proper route and control traffic can be bursty.

**Hybrid** protocols combine characteristics of both Reactive and Proactive in order to obtain a better performance. For example they use the table-driven mode for two hopes nodes and use the reactive mode when they need to search for a long distance device.

Depending on the traffic and the mobility patterns Proactive or Reactive protocols would have better a trade-off.
Let’s suppose we have a network like the one in Figure 2, and we can work with the different protocols described before. Node A communicates with nodes B, F and J.

The proactive protocols will have the routes to all nodes. So for the communication protocol doesn’t need to flood the network with any packet as already know the routes. Some proactive protocols could be OLSR or BABEL.

The reactive protocols will not have any route as we suppose it’s the first communication. The first thing before sending the packets it will have to flood the network in order to find the routes. Some proactive protocols could be AODV or DSR.

And for the hybrid protocols will have routes to B, C, D, E, F, I and L proactively and J reactively. For example some proactive protocols could be ZRF.
2.3. **Applications of MANET**

MANET networks can be really interesting in the smartphone scenario. Nowadays we can find a wide range of situations where the regular network is not able to give service to all users. Among this situation mainly 2 can be found:

- High people density events
- Natural disasters

In one hand Internet has had a big impact on people lives and even more with social networks. Everyday is more common for people to gather in large groups due to social events. If you think in all possible cases where larges groups of people meet together on a regular day you can find many cases. Sports stadiums, music festivals, nightclubs can concentrate a huge amount of people in a really small area. This makes impossible for the regular network to offer service to all the users and this became to the impossibility to communicate for these users. Furthermore at this moment that we are suffering a huge economic crisis, the number of demonstrations has increased markedly and demonstrations are also a huge challenge for the regular network. Besides there are some countries where the network is manipulated for the government, making impossible for the people to talk about certain topics or in certain days, elections days and so.

In the other hand in some regions they suffer from extreme natural disasters that take down the regular network, not momentarily but for a long time, making impossible to communicate in the moment that they need most. One of the more challenging problems that rescue teams have to deal with is to find possible survivors that may be buried down a building or just can be seen with naked eye. With a MANET network that activates automatically when there is no regular service lots of lives can be saved, as maybe they can talk with someone buried and rescue him, or just scan for devices and see if there is someone there.

In all these situations described much of the times people doesn't want to communicate with other people who is far away, actually much of the times the just want to communicate with the other who are around them. Because of that having MANET networks through smartphones could be a perfect solution for these situations. Maybe it couldn't be possible either to make a regular call, but just to have the possibility to communicate with the people near to you, which are in the same place, will solve most of the problems, and also this will help the regular network, as many users wouldn’t be using it and it could serve to users who really need it. In addition taking control of a MANET network is much more difficult than taking control of the regular network, and these would make extremely difficult for governments or other organizations to manipulate or ban peoples communications on certain scenarios.

Apart from these situations that can be considered sensible situations, this networks opens the smartphones to a whole new range of applications that cannot be done with the regular network. Shopping centres, museums, amusement parks, and music festivals can take advantage of these for example sending personalized information to the attendees when a certain event occurs in the area, sending the schedules of actuations. Even more people can use it to share photos, videos and music between them without having to use the Internet.
2.4. **SPAN Framework**

SPAN Framework is an open source implementation of a generalized Mobile Ad-Hoc Network framework.

SPAN Framework is created because recent worldwide events have showcased that our current communications infrastructure is not as reliable as we would like to believe. Natural catastrophes and high-density events can overload the regular network, so SPAN Framework is created as an alternative network to be used in these situations. This framework is based on Wireless Tether for Root Users application written by Harald Mueller and it started out as an open source project licensed under GPLv3. Its architecture is designed to allow arbitrary routing protocol during the runtime.

Span Framework is injected into the existing Android network stack between layers 2 and 3 and this allows the framework to control all the network traffic. Due to this its existence is completely hidden to the OS and this allow to all current apps like Twitter, Facebook, Whatsapp… or other regular apps to simply work.

![Android network stack after SPAN Framework injection](image)

Figure 3 – Android network stack after SPAN Framework injection
Android is primarily designed to configure the built-in wireless chip to operate in managed mode and to authenticate to an existing wireless network. In contrary, an ad-hoc network does not consist of static access points and each device should be capable of intelligently route packets to other peers in the network.

In order to configure the wireless chip in ad-hoc mode SPAN Framework dives deeper than the Android framework and work with the wireless chip drivers directly using the iwconfig Linux command utility to set the parameters of the wireless interface. In order to use iwconfig the Linux kernel must have support for the Wireless Extensions API. The devices that were possible to configure at this moment in Ad-hoc mode were the ones with Broadcom BCM4329 and BCM4330 wireless chipsets as are the only that have Wireless Extensions enabled and the other ones didn’t had the driver source released to the community.
3. **Methodology / project development:**

3.1. **Finding the proper device**

In this study case we are focusing in Android devices, and as Android doesn't support Ad-hoc mode not every device is suitable to work with. Furthermore most of the smartphone have a high cost and as two or three devices may be needed the price has also had to take into account.

The first device that I have tried because of the availability was an Android Stick based on chipset rk3066. After some days of study and some testing I had to discard this device. Although I managed to gain root access there wasn't any kernel source available and that made impossible to continue the work with this device.

Before trying a new device I did some research about Android kernel and as Android kernel is based in Linux kernel each manufacturer should release the code because of the license, but it seems it's not like that. Because of that apart from having some kernel source available I started looking for alternatives as it seems quite difficult to find a device with the proper hardware and that also have released the kernel source.

The next devices I worked with were the MK808b, which didn't have the kernel source, released by its manufacturer but it seems that some developers could get its source form a similar tablet. It also fulfils all hardware requirements of SPAN Framework just in case the kernel didn't work as expected. After a lot of testing, modifying, compiling and flashing the kernel from Omegamoon developer it didn’t work on Android because this kernel was for a Linux distribution for Android devices and furthermore the WI-FI driver was not released by the manufacturer so with that kernel WI-FI didn’t work. As I also picked MK808B because of its compatibility with SPAN Framework I had to change the time plan but I can continue working with this device.
3.2. Preparing the workspace

To be able to modify the kernel recompile it and flash to the device and to program an Android application I had to prepare the workspace, I worked from an UBUNTU PC.

In order to work I needed to set up the following:

- ADB
- Android tool chain
- Android SDK with Eclipse

**ADB (Android Debug Bridge)** is a command line tool, which lets you connect with, and Android device. ADB doesn’t detect mk808b pen so I had to make some modification in order to adb detects the mk808b.

First of all I had to discover the ID Vendor of mk808b which adb uses to identify each device. Using dmesg command it’s easy to see that the ID Vendor of mk808b is 2207. Knowing that a new line must be added in `/etc/udev/rules.d/51-android.rules` and then modify `/android/adb_usb.ini` which lists all ID Vendors detected by ADB. Once this is done ADB detects correctly the mk808b.

**Android tool chain** is the setup, which allows GCC to compile the C files not for Linux but for android.

**Android SDK and Eclipse** are all files needed in order to program with Eclipse the android application. In addition Android SDK includes the ADB I talked earlier and flashboot that is the tool I used to flash the custom kernel and recovery.

3.3. Building the Kernel

The kernel I tried to use was on from Omegamoon Github. As mk808b doesn’t have a kernel source release some developers tried to get a working kernel by adapting a kernel from another similar device.

The main modification that had to be done in the kernel was in the WI-FI driver. All WI-FI chipsets based on BCM4329 and BCM4330 have already support Ad-hoc mode, but they have erased the lines that allow us to activate it.

Basically the file `/drivers/net/wireless/bcmdhd/wl_cfg80211.c` has to be modified. We just need to add the mode NL80211_IFTYPE_ADHOC on the list of supported interfaces.
static s32 wl_setup_wiphy(struct wireless_dev *wdev, struct device *sdiofunc_dev)
{
    // ...
    wdev->wiphy->interface_modes =
    BIT(NL80211_IFTYPE_STATION) | BIT(NL80211_IFTYPE_ADHOC)
#if !defined(WLP2P) && defined(WL_ENABLE_P2P_IF)
    | BIT(NL80211_IFTYPE_MONITOR)
#endif
    | BIT(NL80211_IFTYPE_AP);
    // ...
}

With this simple modification as the driver already had the ad-hoc mode and was just
deactivated, using the WEXT we can set-up the Wi-Fi to work in Ad-hoc mode.

Unfortunately the Wi-Fi driver used was not stable and this lead to some instability as
disconnecting from the network. Furthermore this kernel was not suitable to work with
Android as it only worked in a Linux distribution programed for android devices. All this
problems make this option not viable so I changed the original work plan and started to
work with an alternative.

3.4. SPAN Framework

After some research of how can I setup a mesh network on Android without having to
modify the kernel I found the SPAN Framework. As I explained before it injects in the
layer 2 and 3 and its invisible to the OS. This is a great advantage as all apps are going
to work in the MANET network and no modifications had to be done.

For SPAN Framework to work 3 main conditions must be fulfilled:

- Device has to be rooted
- Has to have WEXT support
- WI-FI chipset must be BCM 4329/4330 or may work with chipsets based on these
  ones.

These characteristics were what I based on to find the MK808B that fulfils all of them but
the SPAN Framework only was prepared to work with:

- Nexus 7 (2012)
- ASUS Transformer Prime
- Galaxy Nexus
- Galaxy SIII

SPAN Framework had to be modified to make possible to enable the ad-hoc mode in
MK808B too. For this I opened the apk with the eclipse and just had to modify the file
deviceconfig.java.
As the Wi-Fi interface it’s called wlan0 I added the device with the other devices like him as all of them needs the same commands to enable the Ad-hoc.
Then the file ManetConfig.java is the one who really enables the Ad-hoc, but no modifications has to be done as all wlan0 type enables the same way.

Once all this modifications are done I just need to recompile the apk and install it to the MK808B.

3.5. Rooting MK808b

In order SPAN Framework to work, it needs root privileges, so first of all we need to root the MK808b. As it comes with one of the new versions of android 4.4 it’s easy to do so through flashing a custom recovery and using the SuperSu application to gain root access.

All Android devices come with a recovery console that is basically a partition on the device’s internal memory and can be booted into. The stock recovery of almost all Android devices provides a few basic yet handy options that allow you to factory reset your device, clear its cache partition, and recover its operating system using an official ROM in zip format. For our goal we need more options that the ones in the original recovery so I’m going to flash the custom recovery ClockworkMod Recovery.

Clockworkmod recovery basically replaces the stock recovery with one that lets you do all you can do with the stock recovery, plus some additional options to give you a lot more control on your device. With a custom recovery, you can install official and unofficial ROMs as well as other updates including hacks, themes, kernels using zip files, wipe not just user data but pretty much every standard partition on your device, mount the storage card for USB mass storage access without leaving recovery, partition your SD card, wipe dalvik cache and battery stats, fix permissions, perform, manage and restore backups, and so on.

I’m going to use the option of installing a Zip to install the SuperSu app that will grant me the root privileges.

To install the CWM I’m going to create an update.zip like it was an official update from Google so with the custom recovery will recognize it as official and will allow us to flash the custom recovery. First of all I’m going to create the basic structure of an update.zip, to do so I’m going to copy the META-INF folder and files from mackief GITHUB as it contains the basic files.

![Update.zip structure](image)

Figure 6 – Update.zip structure
Then I’m going to add the CWM files in the root of the folder, which are the files that I want to flash to the smartphone.

<table>
<thead>
<tr>
<th>flash_image</th>
<th>29/02/2008 10.33</th>
<th>885 KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>META-INF</td>
<td>avui 11.21</td>
<td>--</td>
</tr>
<tr>
<td>misc-boot-recovery.img</td>
<td>29/02/2008 10.33</td>
<td>49 KB</td>
</tr>
<tr>
<td>recovery.img</td>
<td>29/02/2008 10.33</td>
<td>9.8 MB</td>
</tr>
</tbody>
</table>

Figure 7 – CWM files in to the Update.zip

After this the file updater-script in META-INF/com/google/android is the one which says what the update should install to the smartphone, so I need to modify it to make the script install the new recovery.

```bash
assert(getprop("ro.product.device") == "rk3066" || getprop("ro.product.device") == "rk3066c");
ul_print("\n");
ul_print("private:");
ul_print("* CWM-based Recovery for RK3066");
ul_print("* http://androtab.info/clockworkmod/rockchip/ ");
ul_print("* \n");
ul_print("\n");
package_extract_file("flash_image", "/tmp/dump_image");
set_perm(0, 0, 6755, "/tmp/dump_image");
if is_mounted("/mnt/sdcard") then
  run_program("/tmp/dump_image", "recovery", "/mnt/sdcard/dumped_recovery.img");
package_extract_file("misc-boot-recovery.img", "/tmp/misc.img");
write_raw_image("/tmp/misc.img", "misc");
delete("/tmp/misc.img");
else
  if is_mounted("/mnt/external_sd") then
    run_program("/tmp/dump_image", "recovery", "/mnt/external_sd/dumped_recovery.img");
package_extract_file("misc-boot-recovery.img", "/tmp/misc.img");
write_raw_image("/tmp/misc.img", "misc");
delete("/tmp/misc.img");
  endif;
endif;
delete("/tmp/dump_image");
if is_mounted("/system") then
  unmount("/system");
endif;
package_extract_file("flash_image", "/tmp/test");
set_perm(0, 0, 6755, "/tmp/test");
mount("extra", "MTD", "system", "/system");
if run_program("/tmp/test", "-e", "/system/etc/install-recovery.sh") == 0 then
  set_perm(0, 0, 644, "/system/etc/install-recovery.sh");
  ul_print("/system/etc/install-recovery.sh is disabled");
endif;
unmount("/system");
delete("/tmp/test");
package_extract_file("recovery.img", "/tmp/recovery.img");
if write_raw_image("/tmp/recovery.img", "recovery") then
  ul_print("CWM-based Recovery for RK3066 is installed");
endif;
delete("/tmp/recovery.img");
```

Figure 8 – Updater script
As can be seen in the first line the device is checked, because every device has its own recovery image. If we use a recovery image from another device it could be bigger than the original one and it will re-write memory sectors from boot partition and this will cause the device to brake as it couldn’t boot the Android anymore. Once we are sure the device is the same as the recovery was made for it installs the recovery.

Now I have the update.zip prepare to be flashed into the mk808b. I start the recovery mode by pushing with a clip a hidden button and starting the device. Then the option “Apply update from external storage” is selected and it installs the CWM.

Figure 9 – Android recovery
Then with the new Recovery installed and with the option “Install Zip from sd card” I installed the SuperSu apk to gain root access.

![Figure 10 – CWM recovery](image)

3.6. Testing SPAN Framework

In order to test if SPAN framework works I used their own applications to see if MK808b once rooted was able to create a mesh network. I set up 2 MK808b with the SPAN Framework running and watched at OLSR table to see if the devices see themselves.

![Figure 11 – OLSR information table](image)
As it can be seen the device with the ip 192.168.1.101 sees the 192.168.1.100 in the neighbours table. This shows us that the mesh is actually working between the 2 android sticks.

3.7. Building Chat Application

To prove that the MANET network is working between the Android devices I have modified the SPAN Framework to create an application that allows the user to send messages to near devices that are running the same application and also offers the possibility to send an emergency message in case you need it.

To program the android application I have used Eclipse. There I created new classes and modified some existing classes from the SPAN Framework.

Once the application was ready, I used Eclipse to build and sign an APK file with a valid key. This way Android OS recognizes as a valid Android application and allows it to install on the devices.

The APK can be found in the bin folder of the used workspace. Then we only need to put the APK in the SD card and install the application a run it.

Figure 12 – Eclipse IDE
I have also designed the visual interface of the application using Eclipse. Using the menus from Eclipse editing manually the XML file that contain all the visual details, I have designed a user interface.

![Eclipse GUI design](image)

*Figure 13 – Eclipse GUI design*
3.8. **Testing Mesh Chat**

Once I had the 3 MK808b with Mesh Chat installed and running I could do some testing and observe how OLSR works.

First of all I needed to configure the WIFI adapter from the computer in monitor mode in order to capture all the traffic with the Wireshark. Once I was able to read the packets that all the devices were sending I could notice the hello packets that OLSR use for discovers the neighbour and how once the neighbour was discovered it was added to the OLSR table info.

![Figure 14 – OLSR Hello message](image)

All this process was automatically done by the protocol itself just to obtain the routes to its possible neighbours. After this we can see on the Mesh Chat application how the clients start showing available to send messages to them. Once clients start showing up in the client list the user can start using the application and chatting with other people.

OLSR sends Hello messages periodically so the client list is going to be updated periodically too. Each time it sends the Hello message 3 situations can happen. Sometimes we will lose the possibility to communicate with some node as there will be no possible route to it, sometimes new devices will appear available as a possible route will appear and some times anything is going to change.

With this test we can check the theoretical behaviour of OLSR and also check that our application is really working.
4. Results

The main result of this thesis is an Android application that is able to create a mesh network in Android smartphones and uses this network to send messages to all the users. So I can happily say that the main goal of the thesis is fulfilled. With this it is also proved that it is really possible to create and communicate through mesh networks with android devices, although all its difficulties and inconvenience. Now let’s see how Mesh Chat looks:

First of all the application starts with this screen while it loads all the necessary code

![Figure 15 – App splash screen](image15)

When finishes loading it shows a screen where you can see all available devices near to you. Even if any device is showed up an Emergency option is always visible, as it will broadcast a message to all devices.

![Figure 16 – Available client list activity](image16)
If it has some available device, once selected it brings you to the screen where you can send messages to the selected user. The first Empty box shows previous message and you can write a new one in the second box.

Figure 17 – Message creation activity

On the left side there is a sliding menu where you can choose all the application options

Figure 18 – Sliding menu
• **Xat:** Is the main part of the application where you can choose the device you wanted to communicate with and send the message.

• **Info routing:** Shows the information of the routing table, where you can see all visible neighbours along with other OLSR information of the network.

• **Create Mesh Network:** As the name indicates starts the Mesh Network

• **Settings:** Shows you a menu with different configuration options.

In the Setting you can change for example the name that your device will advertise and will be listed in the others devices list. Apart from that you can change some network parameters needed to successfully connect all the devices.
5. **Social impact**

Not only this project but also the whole mesh technology applied to the smartphone could have a huge impact in the society. When Internet came out it revolutionizes the way people communicate. After that Internet arrived to the mobile phones, and with chat application like Whatsapp revolutionized again how people communicate.

Mesh technology give the possibility to communicate with people that is near to you even when today it is no possible. Maybe because there is a huge event and there is too much people for the regular network to give service to everyone or maybe because of some natural disaster that took down the network. In all this cases people want or need to communicate with other people and mesh network could be the perfect solution in this situation.

Nowadays there are lots of countries that manipulate Internet, banning some services in the whole country and banning site where people talk about certain topics. For example China is one of the known countries that have a high manipulation from the government. During the pasts weeks there have been some demonstrations against the government and of course it is no possible to talk about this on the regular Internet. So how people organize themselves? They use mesh technology through the Bluetooth to accomplish a mesh network that the government cannot control.

This technology could also be used for saving life when natural disaster occurs. When some people get trapped down a building if their mobile phone could call or send messages through a mesh network it would be much more easily to save them.

Those are just some examples of how the mesh technology applied on smartphones could be used. Not only for having fun, but also for creating a network that no one could control, as every user is part of the network and a network that doesn't relay on a physical structure that can be destroyed.
6. **Budget**

This thesis objective was not to build a prototype but I have used and bought some components to complete my objectives.

Regarding to the materials:

- 3 Android Sticks MK808B which have a cost of 40 euros each one
- 1 computer to develop all the thesis 500 euros
- 1 Wi-Fi network adapter 20 euros

So all material used cost about 560 euros.

Now supposing that Junior Engineer salary in Barcelona is about 33000 euros per year, working 8 hours per day:

- 1 Junior Engineer worked 20h per week for 8 months

The engineer travelled to Leipzig for one week to assist to the Battlemesh in order to get advise about mesh networks and mesh protocols. The cost of the trip is 400 euros.

The cost of the engineer working on the thesis is about 11000 euros plus the Battlemesh assistance of 400 euros makes a total of 11400 euros.

So having the cost of all the materials used and the salary and expenses of the engineer the total cost of this thesis is 11960 euros.

If I was a company I should add a 15% (1794€) to the cost for the use of the facilities and also a 20% (2392€) as the product benefit to sale to another company. That makes a total of 16146€.
7. **Conclusions and future development:**

After researching a lot of how MANET networks work and which are its advantages and disadvantages, I get to the conclusion that this kind of networks could help people in many situations, some of them in situations of death or life. But as far as android doesn’t support by itself the Ad-hoc mode it’s not likely to think that some application could use this kind of networks for the general public.

All the modifications I had to make to enable ad-hoc mode just for one device shows us that with most of the current market devices would be totally impossible to do so. The main problem is that although Android is open source, no every single line of the code is really open to the developers. Most of the time the only part that is opened is just the main android code, but all code related to a specific device that is not part of android itself, as drivers and specific manufacturers modules it’s never released to the public and this make totally impossible for the developers to build some solution to enable Ad-hoc mode in most of the devices. In addition, although all the manufacturers release all needed code, it’s not likely for regular people to make all this process of rooting and flashing a custom kernel and a custom recovery just to install some app. Because of this the only way of really implementing MANET networks in smartphones would be if Google add support from default android and final user don’t need to modify anything of their smartphones, just download some applications form the market and run it.

Although its true that MANET networks had high power consumptions and it could be a reasonable point from Google for not supporting Ad-hoc mode, I’m sure that a better alternative than disabling it could be found. For example it could just active Ad-hoc mode when there is no connectivity from the regular network or let the user decide, knowing the high consume it would have, if they want to have Ad-hoc enabled or not.

To sum up, MANET networks on smartphones could be a really useful utility in many day life situations, but as long as Google doesn’t implement native support to them it’s not likely to think that they are going to expand to the final user and have a really utilization among them.
Bibliography:


Glossary

MANET: Mobile Ad-hoc Network

Mesh network: is a kind of network topology

Ad-hoc: decentralized type of wireless network

Root: Gain root permissions in an Android device

SPAN Framework: Framework developed for the MITRE organization to create mesh networks on android.

Eclipse: programing workplace for android and other programing languages.