FLINDERS UNIVERSITY INSA LYON

Interconnecting a Serval Mesh with a SMS mobile operator backend

By

Vincent HOARAU

Supervisor :

Dr. Paul Gardner-Stephen

A master thesis submitted to the School of Computer Science, Engineering, and Mathematics in the Faculty of Science and Engineering during my exchange semester in partial fulfillment of the requirements for the degree of Master of Telecommunications

In the

Flinders University

June 2018

Declaration of Personal Work

"I, Vincent HOARAU, certify that this work does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university: and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text."

Signed : Vincent HOARAU

Date : 30 June 2018

Abstract

Nowadays, telecommunications and especially mobile and wireless telecommunications forms an integral part of our daily life. We use these technologies in any business sectors : education, security, food production and factories etc. And for this, they are today considered as an economic development vector. Any business or company which wants to develop itself must have access to these technologies. Also, these technologies are massively used by population to socially integrate, by emergency service to help people or simply to communicate.

In some conditions, these technologies are not available because these hard conditions prevent these technologies to work or to be deployed. These conditions can be geographical, topography-related or demographic because areas are not profitable for the different technologies providers.

The Serval Project aims to bring a solution to people who lives in these areas. The Serval Project can build a mesh network which offers voice and text messaging services. With simplicity and low-cost as main concerns, this build can be easily and quickly deployed with low-cost wireless devices.

However, this mesh operates independently from existing network and so this mesh can't connect to them. This project aims to bring a first solution to connect the mesh to live mobile networks and offers text messaging services between mesh and customers of the mobile network operators. The mesh could be seen as an extension of existing technologies when they are not available.

Keywords : Serval Project, MeshMS gateway, SMPP, SMS, Mobile Network

Acknowledgments

As an exchange student, I would like to thank the people who works at the administration office of my home school, INSA de Lyon, who allow me to do this exchange at Flinders University and so Flinders University's offices as well. I would like to thank the other students with who I have spent a lot of time in the Telecommunications laboratory for their help and sympathy, and also to the local members of this laboratory : Ghassan, Benjamin, Jeremy.

A special mention must go to Dr. Paul Gardner-Stephen, my supervisor on this project, who offered me the opportunity to work on this project which was really interesting. Paul also gave me advices, guidelines and general support.

Abbreviations

xG	**x**-**G**eneration (second : 2G ; Third : 3G ; Fourth : 4G)
ACK	**ACK**nowledgement
AUC	**AU**thentication **C**enter
BSC	**B**ase **S**tation **C**ontroller
BTS	**B**ase **T**ransceiver **S**tation
CSMA/CA	**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance
ESME	**E**xternal **S**hort **M**essaging **E**ntity
GPS	**G**lobal **P**ositioning **S**ystem
GSM	**G**lobal **S**ystem for **M**obile communications
HLR	**H**ome **L**ocation **R**egister
IOT	**I**nternet **O**f **T**hings
IP	**I**nternet **P**rotocol
MDP	**M**esh **D**atagram **P**rotocol
MNO	**M**obile **N**etwork **O**perator
MS	**M**obile **S**tation
MSC	**M**obile **S**witching **C**enter
MSISDN	**M**obile **S**tation **I**ntegrated **S**ervices **D**igital Network **N**umber
OSI	**O**pen **S**ystem **I**nterconnection
OSMOCOM	**O**pen **S**ource **MO**bile **COM**munications
PDU	**P**acket **D**ata **U**nit
SID	**S**erval **Id**entification
SIP	**S**ession **I**nitiation **P**rotocol
SME	**S**hort **M**essaging **E**ntity

SM-MO	**S**hort **M**essage – **M**obile **O**riginated
SM-MT	**S**hort **M**essage – **M**obile **T**erminated
SMPP	**S**hort **M**essage **P**eer-to-**P**eer
SMS	**S**hort **M**essage **S**ervice
SMS-C	**S**hort **M**essage **S**ervice - **C**enter
SMS-SC	**S**hort **M**essage **S**ervice – **S**ervice **C**enter
SDR	**S**oftware **D**efined **R**adio
TCP	**T**ransmission **C**ontrol **P**rotocol
VLR	**V**isitor **L**ocation **R**egister
VoIP	**V**oice **o**ver **I**nternet **P**rotocol

Table of contents

Table of figures

Introduction

In the last two decades, mobile and wireless technologies have been developed more and more. In firms and private individuals, Wi-fi can be found in any firms and private individuals properties for local networks or to access Internet. GPS has been implemented on our mobile phones and most of population is using GPS and satellites technologies daily to receive television stations or Internet again, in the outback in Australia for example. Mobile networks are becoming more and more common in the late 1990s. Since 1993 and the first GSM mobile network launched by Telstra to the next generation of mobile network, the 5G, which is expected for 2020, these wireless networks has become essential for the economy and everyday life. These networks can be used for voice communication, text messages services but nowadays to access Internet and to serve the Internet of things (IOT) : sensors, drones etc. Nevertheless, these mobile networks cannot be available in any regions of the world as the GPS can be because they need wide infrastructures to spread the wireless network and most of MNO don't do this. Indeed, some uninhabited or very few populated areas are unprofitable to develop infrastructures there or the cost are too high to connect these infrastructures to the other networks. Therefore, people who live in these kinds of regions have no way to communicate easily and cost-efficiently. As an example, and closer to us, around 75% of the Australian territory is still not covered by mobile networks.

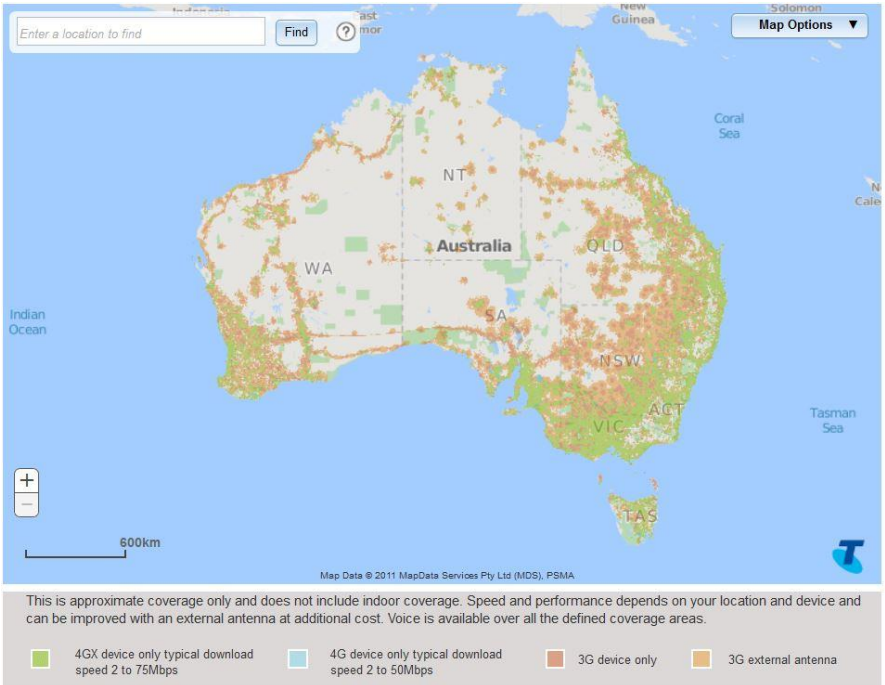

Figure 1 - Telstra (Australian MNO) mobile network coverage in June 2018 (credit : telstra.com.au)

Therefore, some sections of roads are not covered and so people cannot join emergency services if they have a problem (excepted satellite communications which can be expensive).

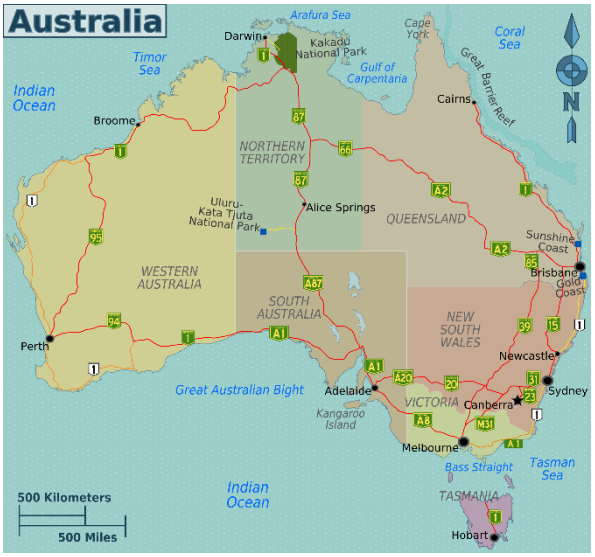

Figure 2 - Main roads on Australian territory (credit : common.wikimedia.org)

Also, these infrastructures are highly vulnerable to natural catastrophes as hurricanes or earthquake. The few hours which are following one of these catastrophes are crucial for emergency services, because they have a lot of opportunities to save lives, but they could not have the ability to communicate with each other or even with the population because mobile services as voice are down. To answer to these two issues, Serval Project has been launched and its architecture is adapted to bring mobile communications in these kinds of situations.

This document will have several chapters : the chapters 1, 2 and 3 will be a literature review and will talk about existing technologies, protocols and their working principles as the following 4 5 and 6 will describe how we can use all these protocols to implement this project.

Chapter 1. The Serval Project

1.1 Motivation and goal

The Serval Project has been designed to solve these issues with two main priorities: simplicity and cost-efficiency. The first priority allows a Serval mesh to be built anywhere in any conditions in a short duration (several hours) and by anyone. The second is, no matter the size of the network, the number of people involved, it might stay low-cost. The Project is a non-profit organisation, is open-source and so everyone is encouraged to test or to contribute to the Serval Project by coding or doing some reports about bugs or functionalities.

As support, the team behind the Serval Project can count on some Oceania MNO, e.g. Telecom Vanuatu and Digicel Vanuatu and also on the New-Zealand Red Cross to facilitate logistics organisation. These supports allow the Serval Project to be experimented in real conditions in Vanuatuan islands.

1.2 The Mesh Architecture

There is not only one example of architecture for the Serval Mesh as it is completely depending on the devices involved. However, at this stage of the project, the serval Mesh can contains two types of device :

Figure 3 - Serval App on an Android device (credit : developer.servalproject.org)

A smartphone with the Serval Mesh Android Application. This type of phone can build a network with some other same phones using Wi-fi or Bluetooth connection and so will be able to communicate with these phones by voice or text messaging service. A version of this app is available for iOS (Apple Phone operating system) and permits only text messaging. Any smartphone with the Android Operating System should be able to run this application because its conception has planned low-cost devices, which means low performance.

The second device is a customized device, the Mesh Extender. Built from the openWRT firmware, its main goal is to extend the mesh network created by Serval Smartphones. Indeed, this device embed wireless connections as Wifi but also UHF radio to connect to other Mesh Extenders. This radio interface allows a range of hundreds of meters for the mesh, compared to tens of meters with Wifi only. The frequency range for this radio interface is 900 MHz or 868 MHz, according to the legislation where the Mesh Extender will operate.

Figure 4 - Mesh Extender Device (credit: developer.servalproject.org)

Basically, to build a mesh network, the Serval Project can rely on these devices and a typical example of mesh could be the following one :

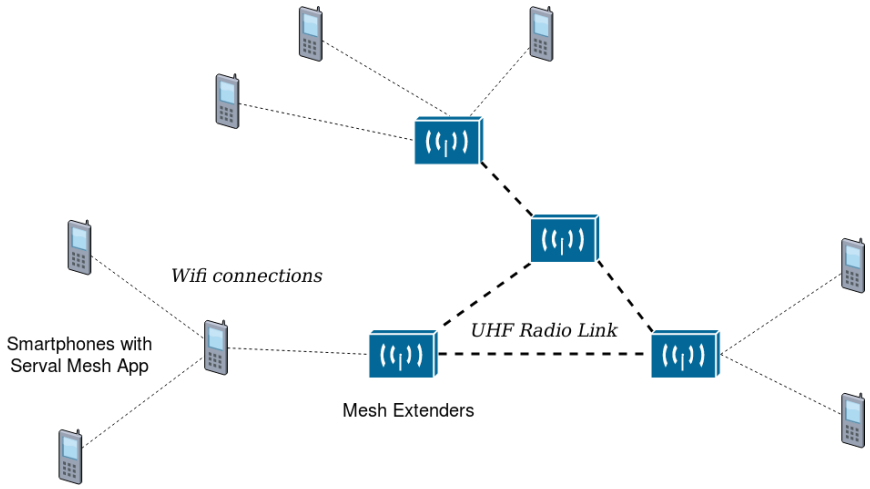

Figure 5 - An example of Serval Project Mesh

All the smartphones involved in this mesh can contact each other with voice and text messaging services (via MeshMS service) and there is no concrete limitation size. A smartphone can join or leave the mash at any time, just as a Mesh Extender can do. The mesh will be built again with new established or removed radio links. The limitations will be instead related on the performances and the operating life of the devices. We can easily guess that wireless links could be congested if the number of smartphones is around hundreds or thousands.

1.3 Serval Protocols

For a better understanding about the project, it seems mandatory to introduce the protocols which allows communication between serval devices. First, these protocols are adapted to a mesh architecture and so take in count the issues of such an architecture: Mobility, device number, packet loss risks and changeable architecture. These customized protocols start from the layer 3 (Network layer) from the OSI model.

1.3.1 The Serval DNA core program

Before explaining the details of the protocols, we will first talk about the Serval DNA core program. This core program, most written in C programming language, is the core component of any device in the mesh network : smartphone or Mesh Extender. Also this core program can be implemented in a computer or any device which can run a Linux system and has network interfaces. In that way, a computer can be considered as a Serval device because the Serval DNA allows this computer to send and receive messages as a smartphone would do, but also to forward them as a Mesh Extender would do.

1.3.2 The Mesh Datagram Protocol (MDP)

The MDP protocol is a layer 3 protocol developed for the serval Project and is entirely suitable for ad-hoc wireless link and mesh networks (Serval Project Team, 2017). The links used at the layer 2 can be either wired or wireless links and so MDP can be carried over any layer 2 protocols (e.g. CSMA/CA in Wi-Fi links).

It can guarantee that the message has been properly delivered but duplicated messages can also occur. Indeed, MDP uses per-link retransmission and adaptative link-state routing to improve packet delivery rates (Serval Project Team, 2017), so the risk of duplicated packets exists especially when the network is experimenting some architectural changes, new devices added as example.

We can find these elements in the MDP protocol basic concepts :

- **Node** : a device with one or more interfaces with one and only one Serval DNA instance.

- **Link** : a direct connection between two Serval mesh devices.

- **Serval ID (SID)** : A SID is a unique 256-bit public key generated randomly. This SID is used as a MDP address to identify any device in the network : identify senders, recipients and to encrypt MDP messages.

- **"Network Discovery" Packets** : These packets are sent regularly to discover the network and to share the SID of the neighbors of a node to the other nodes. Each node has entire knowledge about the whole network (SID, nodes etc.)

1.3.3 Rhizome Distribution System

Briefly, the Rhizome Distribution System is a store and forward service for sending and receiving messages in the mesh. The mesh and its protocols are built on a way that any message sent on the mesh is broadcasted to any device but only the recipient can decrypt it from a key derived from the SID key. Rhizome can store the messages in any nodes. MeshMS service is based on this system and on the protocols above.

1.4 Limitation and proposition

At this stage, the Serval Project has one advantage and one limitation based on the same fact. The mesh operates entirely independently of existing networks, so it is very resilient, but the network cannot communicate with existing networks, that is a real limitation to the project.

The proposition is to build a bridge between mesh networks and existing mobile networks. This bridge would allow text messaging services and, in that way, the mesh could be seen as an entire extension of mobile networks when this mobile network is not available due to hard conditions (topography and/or unprofitable conditions). This project makes sense in islands for example. Sometimes, we could have the main island which is covered when secondary islands (which are less inhabited) are not covered. These islands could profit of the mesh to communicate with the main island. Closer to the technical, a device mesh could be able to join a MNO customer with this bridge, even if

the MNO customer is not taking part of the mesh and does not have the Serval DNA on his/her phone.

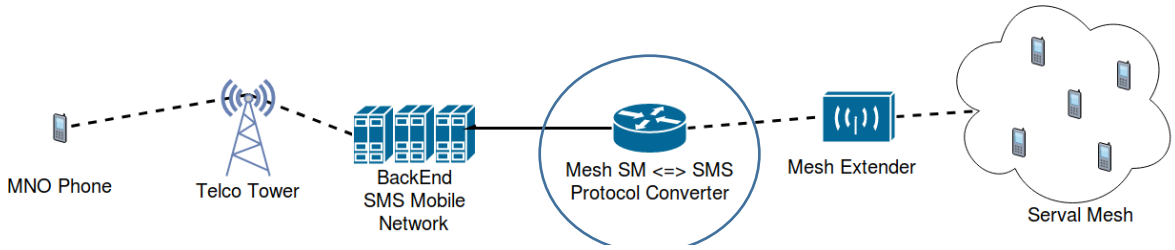

Figure 6 - General principle diagram of the final architecture

To implement this MeshMS to SMS gateway, it is mandatory to acquire knowledges about how a mobile network and the SMS works, the main protocol of SMS, SMPP, and its equipment etc. We will develop the SMPP protocol in the next part.

Chapter 2. The SMPP Protocol

To create our own SMPP entity, we must know more about the SMPP protocol. SMPP v3.4 is the most used protocol to send and forward messages in mobile networks back-end and stays compatible with the 3.3 and 5.0 version.

2.1 The different elements in a SMPP transaction

To send and receive SMS, the protocol SMPP (which can be used stand-alone, e.g. not in a mobile network) needs only 2 elements :

- **The SMS-SC or SMS-C**, which is the server in a SMPP connection. SMPP works like any server-clients protocol. Its main purpose is to store and forward, convert, and deliver the SMS. Also, it maintains connection with the following elements.
- **The (E)SME** which is basically the client. Any device or software which can send/receive SMS with the SMPP protocol can be considered as an ESME, except phones (which are called SME). Its role is to translate, and forward SMS received from a phone to a SMSC in the right protocol, or to receive SMS, or both. ESME are also any massive SMS generator. This kind of ESME can be found in companies which send massively SMS to their customer for advertisements, information etc.

2.2 SMPP Connection Management

To connect an ESME to a SMSC, SMPP is mostly using the following protocols :

- IP, at the layer 3 (Network) at the OSI model. The ESME has an IP address and must know the IP address of the server.
- TCP, at the layer 4 (Transport). It means that SMPP is working according a “connection and acknowledgement mode”, as TCP requires it.
- And so SMPP, at the layer 5 (Session), 6 (Presentation) and 7 (Application).

TCP means that each message sent between these two entities must be acknowledged. To accept connection, any SMSC must have an open TCP socket to listen potential connections from an ESME. The well-known port for SMPP is TCP/2775.

Once this socket is open, the ESME must connect with a TCP connection, the three-way handshake. Then the ESME will send proper SMPP packets to be fully SMPP-connected to the SMSC. The ESME can connect according to 3 modes :

- Transmitter : The ESME can only send/translate SMS
- Receiver : The ESME can only receive SMS and forward them to a phone.
- Transceiver : Can do both roles.

The packet for the connection is called BIND following by the wanted role (e.g. BIND_Transceiver). Once this packet is sent to the SMSC, the ESME is properly connected with the SMPP protocol. On the same way, the packet for disconnection is UNBIND_Transceiver.

One last particularity in SMPP, the connection must be maintained according to the frequency rule required by the SMSC. Indeed, the ESME regularly must send a packet to keep the connection alive. These SMPP packets are called ENQUIRY_LINK and a common time frequency is 30 seconds. Also, we must keep in mind that for each packet sent in this connection, an acknowledgement must be sent by the recipient as well.

2.3 SMPP packets structure

Any SMPP packets has the following fields :

- A **header** with :
  - The **command length** : The length of the packet
  - The **command ID** : a number known by SMPP to recognize the type of command (as example BIND, UNBIND)
  - The **command status** : field to indicate if the command is correct or not (in the ACK, Acknowledgment)
  - The **Sequence Number** : correspondence between a packet and its ACK.
- A body : optional. Briefly, we will find the content of the message

The most important here is not to remind all the fields. Instead, having the basics about the structure will help us to build our own SMPP packets.

Chapter 3. How to send a SMS in a mobile network

3.1 A brief description of the elements in a mobile network

First, an introduction to the different elements to understand their roles. We will focus on 2G (GSM) networks but the mechanisms will not fundamentally change in 3G or 4G. This focus on 2G is because the most widespread mobile network in the world is the 2G and the following generations (especially the 3G) is based on the 2G and share with this generation most of equipment.

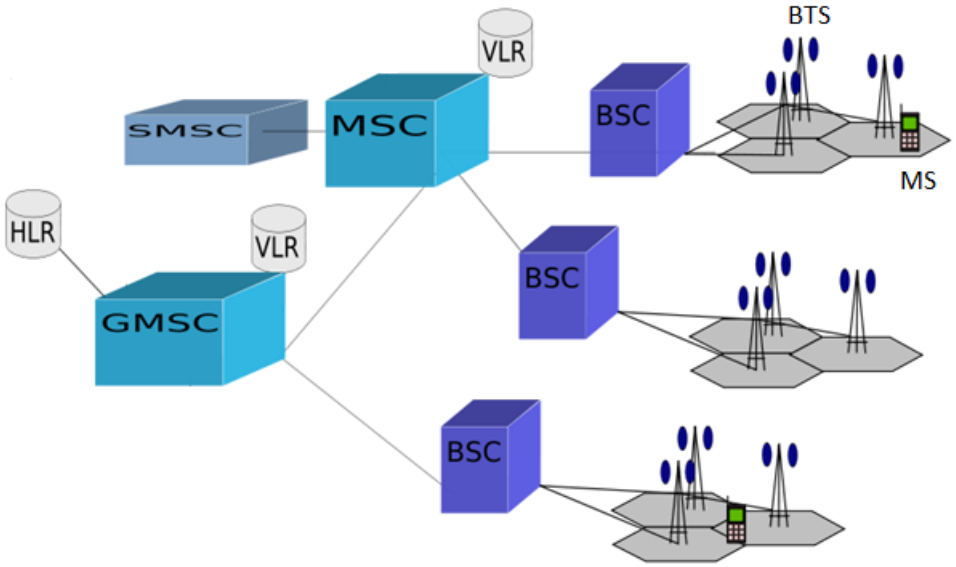

Figure 7 - A simplified 2G Architecture in a SMS context (credit : commons.wikimedia.org)

- The **MS**, Mobile Station : A phone or any device with a SIM card.
- The **BTS**, Base Transceiver Station : The set of devices which allows to spread the mobile network cell. The Cell is the area covered by this BTS.
- The **BSC**, Base Station Controller, which gather a set of BTS and briefly manages: channel frequencies, power emission output for BTS.
- The **MSC**, Mobile Switching Center : gateway in mobile networks. They manage commutation, the VLR.

- The **VLR**, Visitor Location Register : A temporary database, which contains some elements (Number, Localisation) about the MS which are associated to its BTSs. There is usually one VLR per MSC.

- The **GMSC**, the Gateway MSC : Gateway to other networks (Internet or PSTN, Public Switched Telephone Network). In the SMS context, it can be also known as the VMSC, Visited MSC and is important, we will explain it later.
- The **HLR**, Home Location Register. It contains information about the plans of the MS : their authorized services (e.g. quota of datas, access to international calls) and so the HLR knows if the MS has the right to send SMS. This info can also be temporary found in the VLR. The HLR provides its own AUC, Authentication Center, which manages encryption functions.
- The **SMSC**, explained above in Chapter 2, 2.1.

3.2 Send a SMS from a phone to another phone

Let's explain a basic case study when a MS want to send a SMS. There are two different stages : SM-MO (Short Message Mobile Originated) and SM-MT (Short Message Mobile Terminated). The first one refers to the SMS coming from the phone to the SMS Center, where it will be stored. The second refers to the SMS coming from the SMSC going to the phone.

http://www.efort.com/r_tutoriels/SMS_EFORT.pdf

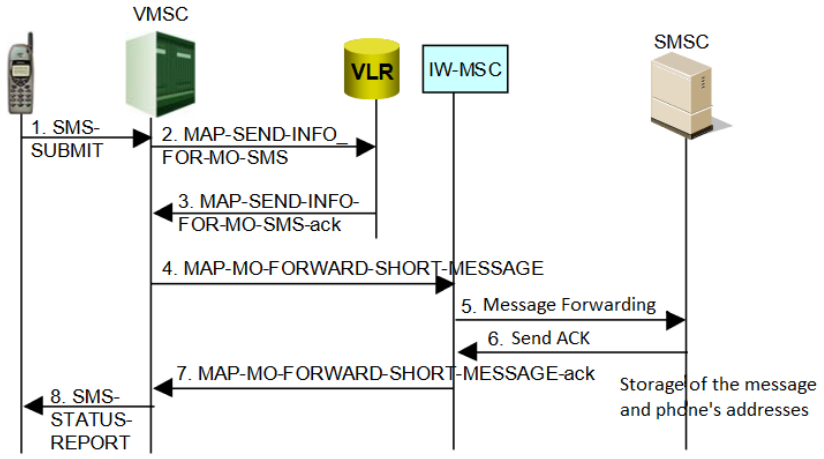

Figure 8 - Steps on a SM-MO Transaction (credit : efort.com)

1. The phone send a SMS, which is forwarding thoughout the network until the (V)MSC. The MSC can have the role of VMSC.
2. The (V)MSC will get at the VLR the real phone numbers (MSISDN) of the sender and recipients and will check if the sender has any restrictions.
3. The VLR then send an ACK packet with this information.
4. If the answer is correct, the (V)MSC send the message to the **IW-MSC function. This function is mostly located within the V(MSC) itself.** The message here contains the SMSC address, the phone numbers (MSISDN) and obviously the content of the message.
5. The SMSC stores this message.

The following steps are only the ACK forwarding to the MS. Then, the SMSC must forward this message to the recipient : this is the SM-MT.

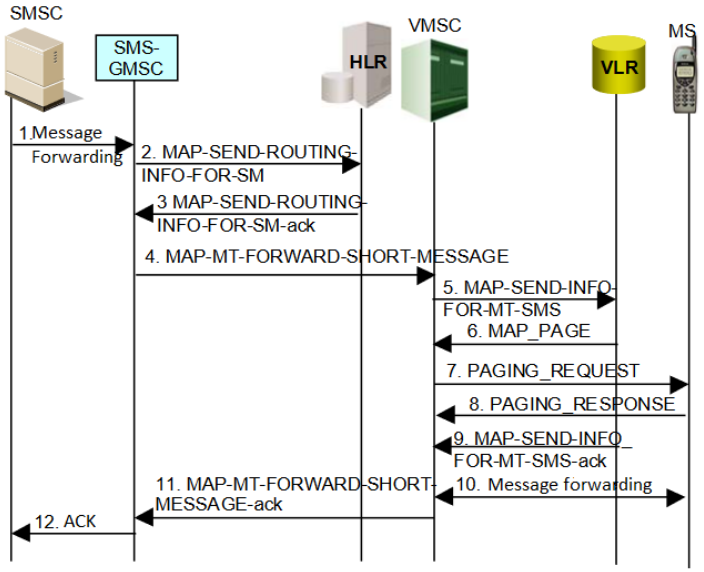

Figure 9 - Steps on a SM-MT Transaction (credit : efort.com)

1. The SMS sends a request to the SMS-GMSC **function. This function is mostly located within the MSC itself.**
2. This function asks to the HLR the inaccurate localisation of the recipient and its potential stage : running, turned off, but also how to reach the recipient (which MSC and so which BSC etc.)
3. These information are sent back to SMS-GMSC (e.g the MSC who is connected to the SMSCenter).
4. This MSC send to the proper MSC (which is welcoming and managing the recipient at this moment) the SMS to send.

5. The (V)MSC closer to the recipient ask to its own VLR the more accurate – but still not precise - localisation of the recipient.
6. This information is sent back to the (V)MSC.
7. Localisation testing to check if the mobile is at the right place designated by the VLR. This is the paging request.
8. Answer from the recipient phone …
9. … And so the (V)MSC eventually knows what is the exact location of the phone.
10. At last the message is sent to the phone and an ACK is generated to the recipient
11. And forwarded throughout the network
12. Until the SMSC, which can remove the SMS.

Finally, the equipment and functions mandatory for the SMS service are :

- The BTS to spread the signal.
- The BSC to manage the BTS and the radio front-end.
- The MSC with its VLR :
    - Forward the messages.
    - Ensures the functions (IW-MSC, SMS-GMSC) and the role ((V)MSC).
- The HLR with its AUC, to store crucial information about phones.
- The SMSC, or closer to us, a SMPP Server.

3.3 Send a SMS from/to an ESME to/from a MNO phone

We can expect that our gateway will be an ESME in SMPP terms: a device who connect to the SMSC via an IP connection to send and forward SMS to any external entities, as the Mesh Network). So, we have to analyse how a SMS can be sent and forwarded, or received.

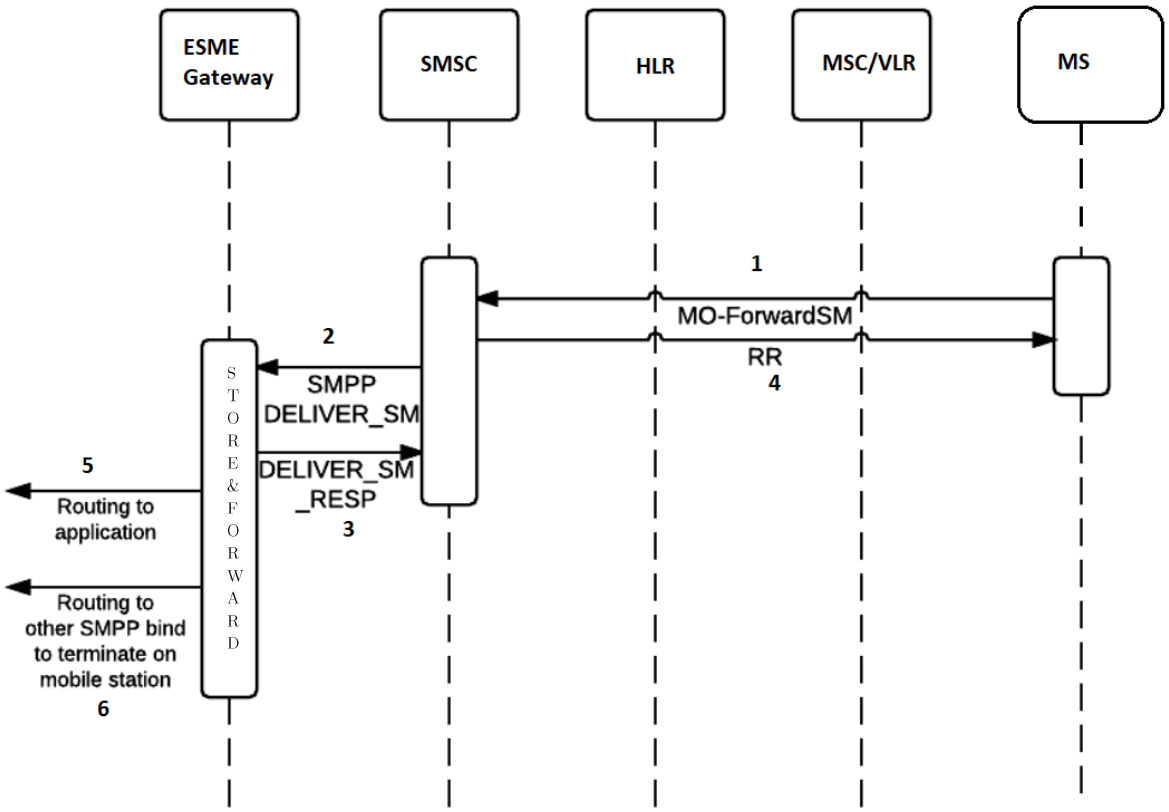

Figure 10 - Simplified Mobile Network-originated SMPP transaction including the gateway (credit : github.com/EMnify)

After the classic way to SMSC, the ESME will act as a SMS center. It will store the sms, send an acknowledgement and determine the best moment to forward this message to its proper recipient in the Serval Mesh.

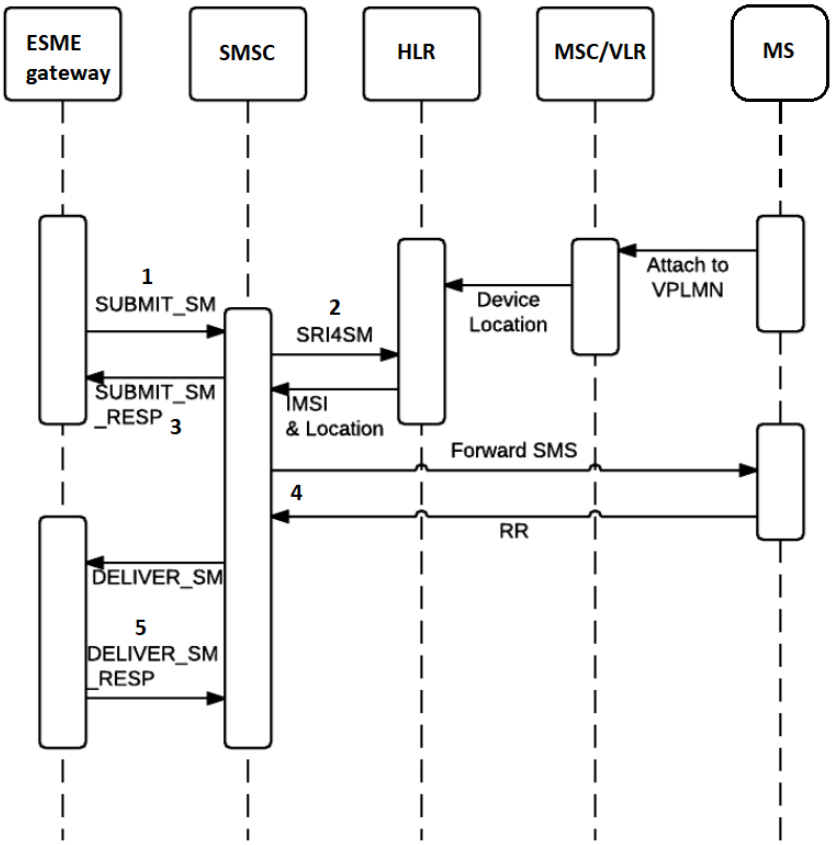

Figure 11 - Simplified Mesh-originated SMPP transaction including the gateway (credit : github.com/EMnify)

1. The ESME has a message from the mesh to forward to a MNO customer and so send it to SMS center.
2. The classic MT process as explained above happen to find the recipient.
3. At the same time, the SMSC tells the ESME gateway that it has the message and will take it over….
4. … Through the store and forward process.
5. When the message is eventually delivered to the proper recipient, an acknowledgement is sent to the ESME.

Chapter 4. Solutions for implementing this environment

The main difficult part is to assembly all the different elements described above. All these elements must be implemented for running comprehensive and realistic tests. Before the description and the analyse of our own gateway, we will introduce the different solutions which has been chosen to reproduce the environment.

4.1 The back-end mobile network

Here is considered as back-end all mobile network equipment except the physical radio part (the BTS). To implement these equipment, there are few solutions :

- Proprietary software/hardware from manufactures : Ericsson, Huawei as example which are expensive because they are completely adapted to a real integration and hard to manage. This solution cannot be envisaged.
- Some software (Open-BTS) who can substitute these functions to allow the same services : voice and text messaging but with complete internet-related protocols : SIP, VoIP.
- Some independents projects which has implemented these devices on a computer with Linux : Osmocom Project.

We chose the last solution for two reasons : cost and open-source software written in C programming language, so we can first know what the software are doing and we can also change some parameters if needed.

Each software from the Osmocom team reproduce the behaviour of a device : osmo-msc for the MSC, osmo-hlr etc... From the projects, we will need the following : osmo-hlr, osmo-msc, osmo-stp which is needed for protocols interface between the MSC and the BSC, osmo-bsc. All these softwares can work on a single computer but can also be splitted on several computers, which could be interesting to analyse resources usage (processor and memory). At last, all the configuration files can be found in THE APPENDIX.

All these programs can almost automatically manage (if configuration files are correct) the different mobile network functions : mobile registrations, signalling messages, data etc.

4.2 Front-end radio

Once the mobile network back-end is set up, it must be connected to radio antennas to broadcast the wireless signals. In the previous part, we have set up all the equipment until the BSC. Then the BTS is still missing.

First, broadcast such a signal is regulated by the Australian government because Frequency spectrum is regulated and belongs to the government and non-authorized signals can interferes with MNO's cells. We have to ensure that the following rules are fulfilled : ALL OF THESE CAUTION PARAMETERS CAN BE FOUND HERE.

- The Power output emission must be around 100 mW (e.g. this is the power output of the Wi-Fi). It strongly limits the risks of interferences with live mobile network especially because Tonsley Building behaves like a Faraday cage (due to its structure and its components).
- Disable the network discovery by non-authorized phones. In that way, the phones in the range cell can't connect to our cell test except the phones that we want to test.
- Disable the emergency calls on our network test.

One experimented solution is an Open-BTS hardware, the 5150 from RangeNetworks company :

Figure 12 - RangeNetwork 5150 series (credit : dailywireless.org)

This equipment embed the OpenBTS software, which reproduces the mobile network services : voice and text messaging by using the protocol SIP, softwares like Asterisk (for VoIP on computer). As we just want to use the radio part of the device we don't need these functionalities.

However, just to test the radio part and its behaviour and stability, the Range Network with its initial functionalities described above has been launched, and no mobile

network was detected. Our different experimentations and tests did not find the issue concerning this hardware.

Nevertheless, the following steps would have been to disable the embedded functionalities. Then, a project from Osmocom, called osmo-bts-trx, must be installed on the 5150 Operating System. This program will make the interface between the radio interfaces and Osmo-BSC (the back-end). These guidelines have not been tested due to lack of time.

If the hardware is still not working on that way, other SDR like the Ettus USPR N310 has been challenged and works within this environment and these conditions.

4.3 Open-source SMPP library

An open-source SMPP library is a library which allows us to create our own SMPP functionalities or programs with some pre-defined functions. Some libraries exists in different programming languages :

- OpenSMPP : A library in Java
- Cloudhopper SMPP (forked from Twitter) : A flexible java implementation
- SMPP-Server : based on Cloudhopper
- PHP-SMPP : A library in PHP
- SMPP.NET : A library in .NET
- LibSmpp34 : A library in C

We chose the LibSmpp34 library for different reasons. Written in C, it would be easier to integrate within the environment : The Serval Core is mostly written in C just as the Osmocom Projects and it is the language I feel most comfortable with.

This library, as it is mentioned in the official documentation, does not manage TCP connection and SMPP session but just provide functions for PDUs handling. These functions just allow us to create our PDU, and we must make some modifications to the PDU to translate it in the right format, then it can be sent throughout a network. We will provide more details about in the following sections. Obviously, integrate this library will allows us to create all SMPP v3.4 packets : BIND, UNBIND, ENQUIRY LINK, SUBMIT etc.

Create a packet need the following code (depending of which type of packets you will need) :

```
bind_transceiver_t a;
bind_transceiver_t b;

memset(&a, 0, sizeof(bind_transceiver_t));
memset(&b, 0, sizeof(bind_transceiver_t));

/* Init PDU ***********************************************************/
b.command_length    = 0;
b.command_id        = BIND_TRANSCEIVER;
b.command_status    = ESME_ROK;
b.sequence_number   = 1;
snprintf((char*)b.system_id, sizeof(b.system_id), "%s", "system_id");
snprintf((char*)b.password, sizeof(b.password), "%s", "pas");
snprintf((char*)b.system_type, sizeof(b.system_type), "%s", "syst");
b.interface_version= 0x34;
b.addr_ton          = 2;
b.addr_npi          = 1;
snprintf((char*)b.address_range, sizeof(b.address_range), "%s", "address_range");
```

Figure 13 - Code for BIND_TRANSCEIVER PDU generation (credit : github.com/osmocom/libosmpp34)

Here we can enter the parameters for the SMPP connection : ID, password, the address. With this kind of codes, you can also easily indicate the content of a SMS. We have a variable called “TEXT” and two variables for phone numbers :

```c
#define TEXTO "Raul Antonio Tremsal"

int
main( int argc, char *argv[] )
{

    submit_sm_t a;
    submit_sm_t b;
    tlv_t tlv;

    memset(&a, 0, sizeof(submit_sm_t));
    memset(&b, 0, sizeof(submit_sm_t));
    memset(&tlv, 0, sizeof(tlv_t));

    /* Init PDU ***********************************************************/
    b.command_length    = 0;
    b.command_id        = SUBMIT_SM;
    b.command_status    = ESME_ROK;
    b.sequence_number   = 1;
    snprintf((char*)b.service_type, sizeof(b.service_type), "%s", "SMS");
    b.source_addr_ton   = 2;
    b.source_addr_npi   = 1;
    snprintf((char*)b.source_addr, sizeof(b.source_addr), "%s",
                                                    "09000011111");
    b.dest_addr_ton     = 2;
    b.dest_addr_npi     = 0;
    snprintf((char*)b.destination_addr, sizeof(b.destination_addr), "%s",
                                                  "1121312309000");
```

Figure 14 - Code for SUBMIT_SM PDU generation (credit : github.com/osmocom/libosmpp34)

As we can notice, the PDU is a structure in C, a group of variables and we cannot send this throughout the network. We have to translate it into hexadecimal characters. The library can do the first step and represent the PDU and its interpretation :

```
The PDU bind_transmitter is packet in
   00 00 00 26 00 00 00 02   00 00 00 00 00 00 00 01   ...&.... ........
   75 73 65 72 00 70 61 73   73 00 74 79 70 65 00 34   user.pas s.type.4
   02 01 31 32 33 00                                   ..123.
```

Figure 15 - A SMPP PDU content and its interpretation

Then we still have to get the characters “00” “26 etc. and consider as real hexadecimal characters and not as characters strings. At last, we can send these hexa characters.

4.4 SMPP software for comprehensive tests

To test the behaviour of our own gateway that we will implement, and to compare it to real SMPP transactions, we need a SMPP server (a SMS center) and some clients (ESME or SME) to generate messages. On that way, they were several possibilities (non-comprehensive list) and they are really close (or the same) to the previous we introduced above. After compiling, installing and testing, ce chose the following softwares to do so :

- SMPPSim, from Selenium Software, which is the server
- Kannel, which can be the clients but is a real versatile software

In computer science, these software are well-known to work together and so some configuration files are already available to make them communicate with each other. They are also simple to use. These software can allow us to generate automatically some PDU for the connection, to send SMS, to maintain connection. Then we can compare the content of these PDU with our own PDUs and so it will be easy to investigate if an issue occurs. Kannel can connect as a transceiver, just like our gateway will.

Chapter 5. Implement the gateway

Our gateway in a mobile network environment will be considered as an ESME : any wired entity which transmits SMPP message to a SMS center. This ESME will be connected as a transceiver. But the gateway will also have other components :

- A SMPP library
- The core program of the network written in C
- A database which will contain a mapping table between the SID of the Serval Mesh and their MSISDN (fictive)
- The Serval DNA core

One concern that we have is to try to keep the program as modular as possible. Any functions should be written when it's possible in a different file. A program will be first a real live text program and it will be entirely autonomous as it will not require interaction with user. In the future, the program could be converted in a daemon to allow it running in background.

Also, C is a procedural and imperative language which means that the program is made of instructions and these instructions will modify the state of the program and will be executed one by one. On a gateway which could manage several messages from two network, that could be a problem. We have to keep in mind the issue and in this part, we will sequentially describe the different parts of the program.

```
gateway@gateway-NUC6CAYS:~/Modulable$ ./main
SMPP Server IP Address : 192.168.8.196
address :192.168.8.196

TCP connection done
parse smpp34_dumpBuf()

------------------------------------------------------------
   00 00 00 2F 00 00 00 09   00 00 00 00 00 00 00 01   .../.... ........
   73 6D 70 70 63 6C 69 65   6E 74 31 00 70 61 73 73   smppclie nt1.pass
   77 6F 72 64 00 27 56 4D   41 27 00 34 00 00 00      word.'VM A'.4...
 Sending the SMPP PDU BIND_TRANSCEIVER
now connected to the SMPP Server
```

Figure 16 - An actual example of the gateway live program

5.1 TCP Connection

After including the SMPP library in our program (a step which we will not describe because it's purely based on programming language), the first instruction is the TCP connection. The SMS Center is waiting for ESME Connection on the port TCP/2775. We have to implement a client socket which will connect to the SMSC. Some libraries in C already implement the main functions to allow us to do so and this, according to the three-way handshake.

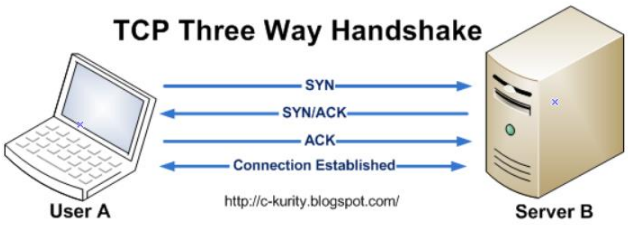

Figure 17 - TCP Three Way Handshake process scheme (CREDIT : c-kurity.blogspot.com)

We don't have to implement the disconnection function as we assume that the server will close automatically the TCP connection after an inactivity lap of time. About the state of the program, it is not associated with a network TCP socket which is connected to a server.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.8.182	192.168.8.196	TCP	74	48004 → 2775 [SYN] Seq=0
2	0.000044340	192.168.8.196	192.168.8.182	TCP	74	2775 → 48004 [SYN, ACK]
3	0.002658633	192.168.8.182	192.168.8.196	TCP	66	48004 → 2775 [ACK] Seq=1

Figure 18 - TCP Three Way Handshake with a packet analyzer

5.2 SMPP Connection

Once the TCP socket is connected, we are able to send our first SMPP packets. As expected in any SMPP transaction between an ESME and a server, a connection is necessary and for this, we will have to send a BIND_TRANSCEIVER packet. In this packet, we will have to indicate some information :

- The ID_command which is BIND_TRANSCEIVER.
- The System_ID which is the identifier of the ESME.
- The password to connect to the SMPP server.

- The Interface version which is 3.4 in our project.
- Type of numero (TON) : format address/number. If unknown, this will be provided by the SMPP server.
- Numbering Plan Identification (NPI). If unknown, this will be provided by the SMPP server as well.

These information can be directly written on the program as they will not frequently change and the library allow us to do that, as show in the Figure XXX. However, even if the PDU is created we are not able to send it in this format. Indeed, our PDU is a structure (a set of variables in C language) and we have to convert this structure in a hexadecimal set of data. For this, the library can help us to do that but cannot entirely convert it. Let's start from the beginning of the PDU generation. Our structure is the following one :

```
U32( instancia, command_length , valueDec_32          );
U32( instancia, command_id     , str_command_id       );     Header
U32( instancia, command_status , str_command_status   );
U32( instancia, sequence_number, test_sequence_number );

   C_OCTET( instancia, system_id, 16 );
   C_OCTET( instancia, password, 9 );
   C_OCTET( instancia, system_type, 13 );
       U08( instancia, interface_version, test_interface_version );
       U08( instancia, addr_ton, str_addr_ton );
       U08( instancia, addr_npi, str_addr_npi );
   C_OCTET( instancia, address_range, 41 );
```

Figure 19 - Bind Transceiver Structure with header

The library has some functions which can unpack and analyse this structure to make its field readable :

```
gateway@gateway-NUC6CAYS:~/Modulable$ ./main
SMPP Server IP Address : 192.168.8.196
address :192.168.8.196

TCP connection done
parse smpp34_dumpPdu()
[command_length:0000002F(OK)][command_id:00000009(OK)][command_status:00000000(OK)][
sequence_number:00000001(OK)][system_id:smppclient1(OK)][password:password(OK)][syst
em_type:'VMA'(OK)][interface_version:34(OK)][addr_ton:00(OK)][addr_npi:00(OK)][addre
ss_range:(OK)]
------------------------------------------------------------
command_length                [0000002F] - [47]
command_id                    [00000009] - [BIND_TRANSCEIVER]
command_status                [00000000] - [ESME_ROK]
sequence_number               [00000001] - [1]
system_id                     [smppclient1]
password                      [password]
system_type                   ['VMA']
interface_version             [34]       - [OK]
addr_ton                      [00]       - [TON_Unknown]
addr_npi                      [00]       - [NPI_Unknown]
address_range                 []

------------------------------------------------------------
parse smpp34_dumpBuf()

------------------------------------------------------------
   00 00 00 2F 00 00 00 09   00 00 00 00 00 00 00 01   .../.... ........
   73 6D 70 70 63 6C 69 65   6E 74 31 00 70 61 73 73   smppclie nt1.pass
   77 6F 72 64 00 27 56 4D   41 27 00 34 00 00 00      word.'VM A'.4...
 Sending the SMPP PDU BIND_TRANSCEIVER
now connected to the SMPP Server
```

Figure 20 - Unpacked and readable SMPP PDU BIND_Transceiver

Here we can obviously find again the information that we put in the program but also, we have the hexadecimal representation of the PDU. This is the right format we want to get to be able to communicate with the server. Even if this is hexadecimal characters, the program is the library is printing them as a string and not as real hexadecimal characters. To clearly explain this issue, let's make an analogy with characters and numbers :

```
int main (int argc, char argv[]){

    int a = 1;
    char b = "1";

    return 0;
}
```

Figure 21 - Analogy with number and string

The first line declares an real number and the second one declares the character/string "1". We can't make any operations (plus, minus) with the second one. This is the same thing for the hexadecimal. We can't send it but they are still considered as string.

And as there is no variable type to define hexadecimal, the C language consider them as a string. We have to convert and to create a string with real hexadecimal character inside. Here is an example of what we want to get :

```
char BindTransceiver[] = "\x00\x00\x00\x2F\x00\x00\x00\x09\x00\x00
```

Figure 22 - Variable with real hexadecimal characters

Concatenate an "\" and a "x" letter before each group of two characters is not operative, as the C language will consider as the real \ and x characters, and not as the format for the following character. We still have to solve this issue. Also, we will have to do this for any SMPP PDU we will send, that's why this translation will be part of another file and so function.

5.3 SMPP Disconnection

In this part, let's talk about disconnection. The packet is not hard to generate and has one field that we have to indicate : the operation, which is UNBIND_TRANSCEIVER. However, we must decide the right moment to send it. If the program receives a keyboard command from a user which order it to stop, it has first to check if there is no messages waiting coming from both networks. After that all the messages have been forwarded, then the program can be interrupted and send the SMPP PDU disconnection.

5.4 Serval DNA core

To be part of the Serval Mesh, the gateway must run the Serval DNA core program. This program is a daemon, a program running in background in UNIX Operating Systems and so we can interact with it by typing commands. These commands offer different options : print the messages received, send messages etc.

Our program will be able to launch these commands and to get the result in a variable with the C function **popen.** This element is crucial to be able to get the messages and its sender and recipient, mandatory information for our gateway. With these 3 information in our gateway, our program is now fully able to transmit from the Mesh to the Mobile network.

5.5 The core loop program gateway

At this stage, our gateway is full connected to the SMSC via the SMPP protocol. Now the program is functional to take over messages and to continuously ensure this behaviour, the program will need a loop. We propose a while loop separated in three parts and the stop condition will be the detection of an interruption signal (from a user). Our program will have to integrate UNIX signal function managements and a database for the SID/MSISDN mapping map.

5.5.1 Getting the messages from the Mesh MS

This part has been explained in the section 5.4 but we will describe the sequence.

- First, we must launch the Serval DNA command to get the messages.
- Then we will have to get the content of this message, its recipient and its sender (which will be SIDs as the message comes from the Mesh).
- Once we have the recipient SID, the gateway must check in its database if it has a matching MSISDN for the SID recipient.
- If there is a match, a SMPP PDU (the SUBMIT_SM) will be created to submit the message. The field of this packet will obviously be filled by the information that we got from previous steps.
- After the conversion in the right hexadecimal format, the message is sent and the gateway is waiting for the acknowledgement.

5.5.2 Getting the messages from mobile network

A second part of the loop will be to get the messages from the mobile network. These messages can arrive at any time and so they will be queued in the TCP socket.

- The program will first check in there is any message waiting in the socket.
- If there is one or several, the program will unpack these messages to get the content of the mandatory fields : content and phone numbers.

- As in the first part, the gateway will look for a match in its database. If a SID has matched to the MSISDN, the content will be encapsulated in a Mesh Message and forwarded through the Mesh.

5.5.3 Maintain the SMPP connection

At the end of the loop, the ESME gateway must maintain the link with the SMPP PDU ENQUIRY_LINK. The time interval will be established by the SMSC and must be known and set before the program has been launched. The PDU generation is not a big issue as the SMPP library has planned it.

5.6 Mapping map database

We have talked about a mapping map database between SID and MSISDN (real reachable phone numbers). After discussions, we thought that a possible implementation of this database could a simple text file because it is easy to implement and maintain. Adding data is attainable, by hand or by the program. C language can also easily read inside the file to find a match.

Chapter 6. Future work and unsolved questions

6.1 Phone numbers and potential duplicate

We have talked in previous sections about phone numbers and a database mapping map. However, we have to clarify which phone numbers will be used for the Serval Mesh devices. Indeed, are these phone numbers have notional values ? Will they have real phone numbers ? In any cases, we and the MNO must check if the MSISDN are not used by another customer or even another mesh device to prevent any conflict or any recipient's mistake.

6.2 First set up in a production environment ?

Related to the first unsolved questions, the procedure to test the final gateway is quite simple. Let's assume that the first question is solved and so the database will be entirely and properly filled out. The gateway will know the devices and their phone numbers because we would have fill its database. Nevertheless, if a future device wants to join the mesh and be able to communicate with the live mobile network, we can wonder how the subscription will occur because the gateway will have to manage it automatically and without any further help from the live mobile network or from a human.

A conceivable solution could be the following one. Before the first setup, the MNO send us a list of possible phone numbers and we implement the gateway with this list. At the beginning of each loop iteration, the gateway will check throughout its Serval Core DNA if any new device has joined the mesh. If so, it would automatically assign a phone number not already used to this new SID in its own database. On that way, the device is registered on the database gateway and on the live mobile network, as the list of MSISDN has been planned for this use.

6.3 Getting the messages from Serval Core DNA

As explained in the 1.3.3 Section Part, all messages are broadcasted to all the devices and the proper recipient is able to decrypt with a derived key from the SID. So, we still have to solve the fact that the gateway will be able to decrypt all the messages.

6.4 Loop duration

Our customized ESME will send during the loop the SMPP PDU ENQUIRY_LINK to maintain the SMPP connection. If the ESME has to send it every 30 seconds according the SMSC's rule, that means that the loop cannot be longer than 30 seconds to allow the gateway to send this packet. The duration of the loop will logically depend on the amount of message to forward. In the worst-case scenario, the gateway could be overloaded and so the loop will last more than 30 seconds. That's why a use of a clock could be very useful. The C language allows us to introduce such a clock. At the end of the clock, the gateway will stop to forward messages and will end its current iteration.

Another solution could be to implement a counter and to measure how much time the gateway will take to convert and forward a message. Then we will be able to know how many messages the gateway could manage in a 30 seconds loop iteration. Again, the C language allow us to measure how much time some instructions will take to be executed.

Results

At this stage of the SMPP Gateway's project, it seems that this project is entire achievable. We did researches on every aspects and elements of this project, from the SMPP protocol to the mobile network simulation or the gateway development. Each of this point has been studied and we can identify some solutions for each of them. However, the links and relations between all of these elements remain the most difficult parts and some issues are remaining.

Nevertheless, the project has plan to simulate a real mobile network and the gateway has its own SMPP library. We are now able to communicate with a SMPP Server even if we can't send our own messages yet. Here is an example of connection (seen from the SMPP server) :

```
2018.06.29 15:21:29 826 INFO    23 Lifecycle Service: OutboundQueue is empty  - waiting
2018.06.29 15:21:29 826 INFO    22 InboundQueue: empty  - waiting
2018.06.29 15:21:29 826 INFO    25 Starting DelayedInboundQueue service....
2018.06.29 15:22:29 827 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:23:29 827 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:24:29 828 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:25:29 828 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:26:29 829 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:27:29 829 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:28:29 830 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:29:29 831 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:30:29 831 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:31:29 832 INFO    25 Processing 0 messages in the delayed inbound queue
2018.06.29 15:32:14 930 INFO    12 StandardConnectionHandler accepted a connection
2018.06.29 15:32:14 955 INFO    12 Protocol handler is of type StandardProtocolHandler
2018.06.29 15:32:14 956 INFO    12 : BIND_TRANSCEIVER:
2018.06.29 15:32:14 956 INFO    12 Hex dump (47) bytes:
2018.06.29 15:32:14 956 INFO    12 0000002F:00000009:00000000:00000001:
2018.06.29 15:32:14 956 INFO    12 736D7070:636C6965:6E743100:70617373:
2018.06.29 15:32:14 957 INFO    12 776F7264:0027564D:41270034:000000
2018.06.29 15:32:14 957 INFO    12 cmd_len=47,cmd_id=9,cmd_status=0,seq_no=1,system_id=smppclient1
2018.06.29 15:32:14 957 INFO    12 password=password,system_type='VMA',interface_version=52,addr_ton=0,addr_npi=0
2018.06.29 15:32:14 957 INFO    12 address_range=
2018.06.29 15:32:14 957 INFO    12
2018.06.29 15:32:14 958 INFO    12 StandardProtocolHandler: setting address range to
2018.06.29 15:32:14 958 INFO    12 New transceiver session bound to SMPPSim
2018.06.29 15:32:14 958 INFO    12 : BIND_TRANSCEIVER_RESP:
2018.06.29 15:32:14 958 INFO    12 Hex dump (24) bytes:
2018.06.29 15:32:14 959 INFO    12 00000018:80000009:00000000:00000001:
2018.06.29 15:32:14 959 INFO    12 534D5050:53696D00:
2018.06.29 15:32:14 959 INFO    12 cmd_len=0,cmd_id=-2147483639,cmd_status=0,seq_no=1,system_id=SMPPSim
2018.06.29 15:32:14 960 INFO    12
2018.06.29 15:32:14 961 INFO    12 1 receivers connected and bound
2018.06.29 15:32:14 961 INFO    22 InboundQueue: empty  - waiting
2018.06.29 15:32:29 833 INFO    25 Processing 0 messages in the delayed inbound queue
```

Figure 23 - Bind Transceiver received by the SMPP Server

The gateway can more specifically : connect with TCP protocol, connect with the SMPP protocol, create its own SMPP packet and disconnect SMPP.

Conclusion

As this stage of its development, the Serval Project can bring a way to communicate everywhere even in hard conditions. Also, it can operate entirely independently of existing networks which is its main strength and at the same time, its main limitation because it has no real way to extend or to evolve. This project aims to build a customized gateway between a Serval Mesh and a live mobile network to allow SMS service between the mesh users and the MNO's customers. This will bring an answer to the limitation without losing the advantage of independence.

Most of mobile networks is using SMPP v3.4 as the SMS protocol, the first step of the project was to acquire knowledge about the protocol : how it works, the different equipment which can be involved, what are the fields of a PDU. Also, being familiar with some SMPP libraries is mandatory because one of these libraries will be required. Indeed, the gateway will have to generate its own SMPP packets and the functions within the library will be used to do so. To allow comprehensive and complete tests, the project and the gateway need to be used within a real mobile network with the proper equipment (e.g. MSC, HLR etc.). Some sub projects from the OSMOCOM project can reproduce these equipment on any computer and allow the same basic functions : phone subscriptions, phone services as voice calls and text messaging service. On that way, the gateway would be integrated in this network and we could test different scenarios and possibilities, and performances.

With all these knowledges, we can now start to implement the gateway, which will be a complete ESME according to the SMPP environment. The structure of the gateway is easy to understand because it is quite short, but a lot of steps are crucial. After a TCP then a SMPP connection, the gateway will have to integrate a loop for getting the messages from both networks and then maintain the connection to the SMSC. At the stage of the project, the gateway can connect to the SMPP Server and the other functions must be written.

About the project, there is some remaining questions. The first one is about the phone numbers used by the serval mesh devices. We must make sure that these numbers can be fictional, or the MNO will need to provide a list of MSISDN. The gateway must be entirely autonomous to manage new phones in the mesh as well. At last, the gateway must keep under control the time of the loop to ensure a viable stability and connection to the SMSC.

This project was a interesting project and is still at an early stage. Indeed, the project can be splitted in two main parts : the mobile network simulation and the gateway itself. After this report, we can have a global view about the project and the following elements: the architecture, the working principles of the gateway, the main issues (performances, implementation).

Appendix

All the following instructions has been tested on GNU/Linux Ubuntu 17.10. Howerver, any GNU/Linux distribution should be able to run the softwares.

Github link : https://github.com/Endast974/SMPPGateway

1. Osmocom project and configuration files

The Osmocom projects have been experimented to simulate a live mobile network. The config files for each software can be found on the .zip file or on github.

Here is the link which describes how to install one component from Osmocom projects: https://osmocom.org/projects/cellular-infrastructure/wiki/Latest_Builds

There is an order to launch the different programs. First, we have to launch the MSC, then the HLR, the STP and at last the BSC. The command to launch is similar for all the projects :

Sudo <osmocom-project> -c <configfile.conf>

One example could be : ***sudo osmo-hlr -c osmo-hlr.conf***

2. The SMSC with the software SMPPSIM

This software is quite simple to install as it required only one configuration file. The configuration file can be found in the zip or on github.

The user guide and the software can be found at the following link : http://www.seleniumsoftware.com/downloads.html

To run the software, we should check the parameters on the file **conf/smppsim.props** and if the parameters are correct, we can launch the software by running the following command :

Sudo sh startsmppsim.sh

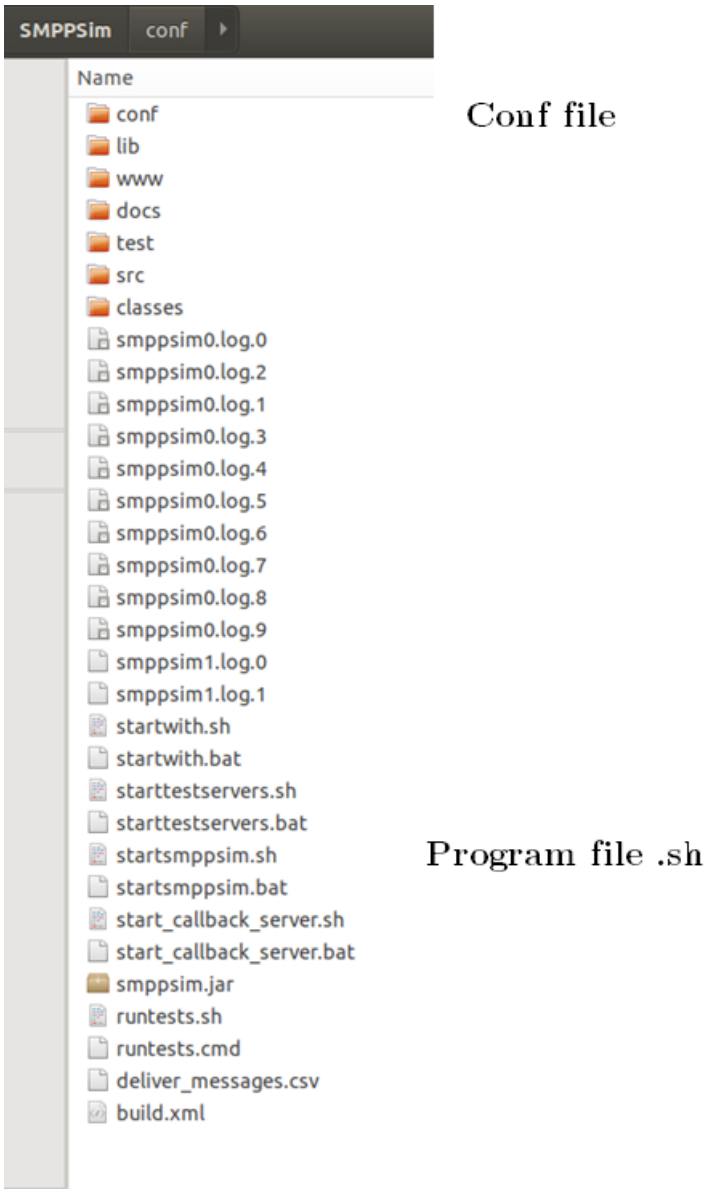

Figure 24 - Tree of SMPP Sim Software

3. Kannel : The software for reference SMPP transaction

The software and the user guide can be found at the following links : https://www.kannel.org/download/1.4.5/userguide-1.4.5/userguide.html and https://www.kannel.org/download.shtml . We recommend to work with the stable release.

After building the software according to the user guide, you will have the following folder :

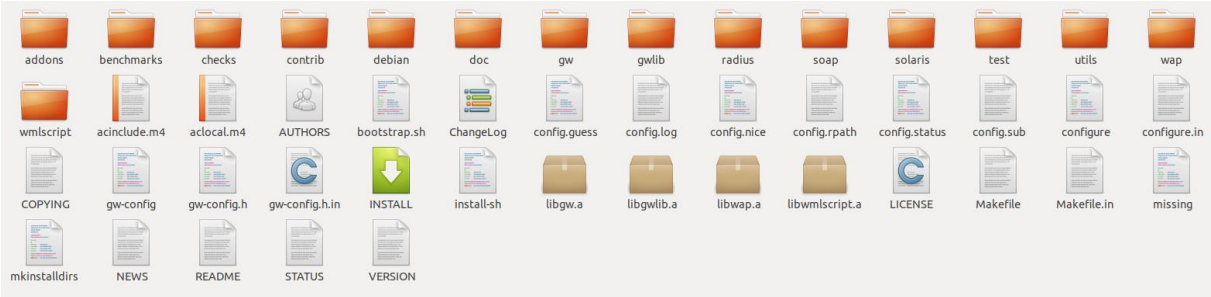

Figure 25 - Kannel's folder and files

The most important folder is the folder called **gw**. We will find in this folder the configuration file (**smskannel.conf**, to connect to SMPPSim) and the software which must be run. The config file is on the zip and on github.

Launching the Kannel software is in two parts. First, we have to launch the main core program, called **bearerbox** which is the interface with the phones (with the smsbox). The second program is the box you will need, so the smsbox. Both of them can be launched with the following commands (in this order) :

Sudo ./bearerbox -v 1 smskannel.conf

Sudo ./smsbox -v 1 smskannel.conf

-v 1 indicates the degree of details for the logs messages. Launch a smsbox without the bearerbox will not work.

4. Gateway Source Code

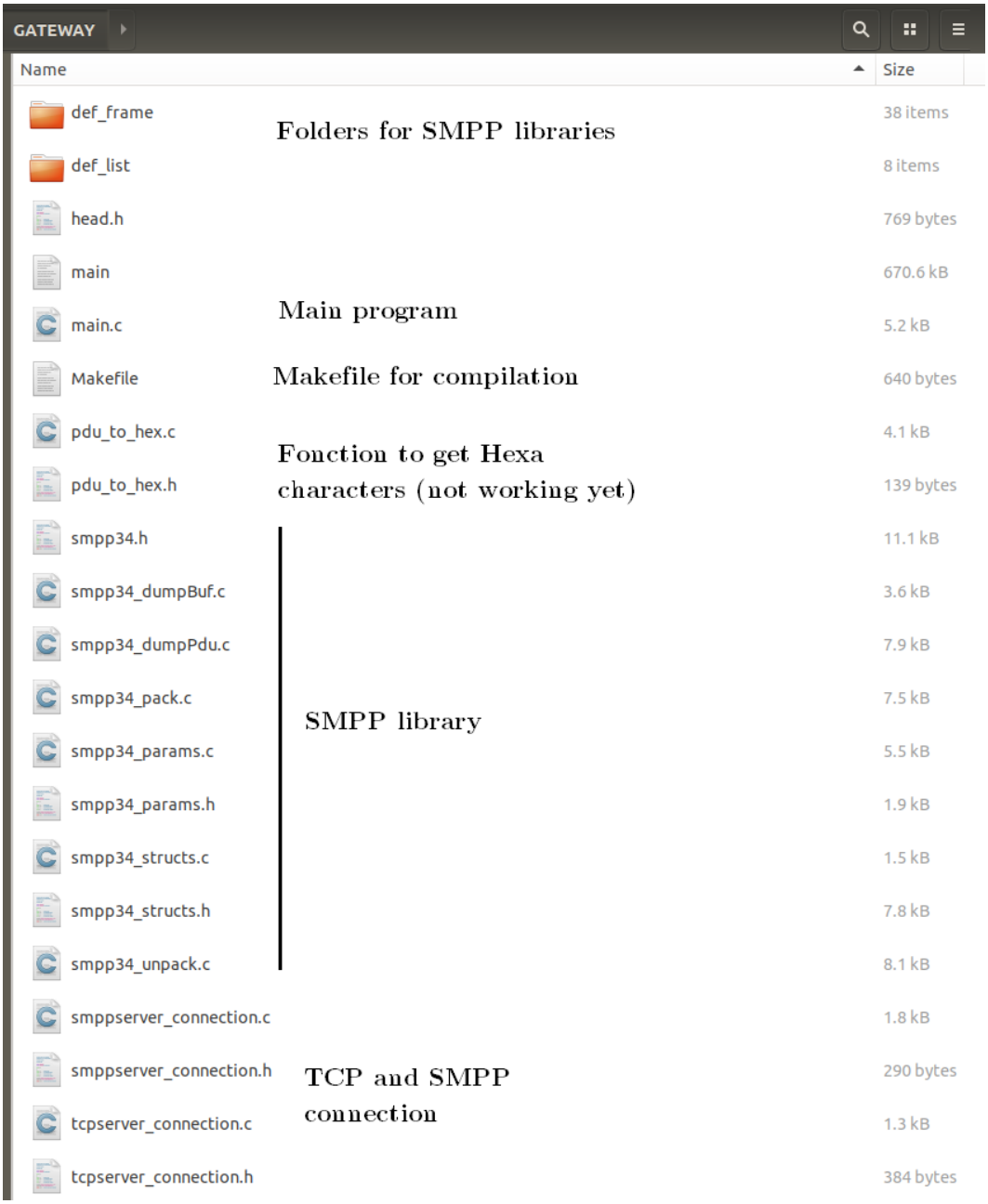

Figure 26 - Gateway Folder

- To clean the folders (remove .o files), launch ***make clean***.
- To compile, launch ***make***.

References

[1] ***The Serval Project : Practical Wireless Ad-Hoc Mobile Telecommunications***. Dr. Paul Gardner-Stephen. Rural, Remote & Humanitarian Telecommunications Fellow, Flinders University and Founder, Serval Project, Inc. 22 July 2011

[2] ***The Serval Mesh: A Platform for Resilient Communications in Disaster & Crisis***. Paul Gardner-Stephen, Romana Challans, Jeremy Lakeman, Andrew Bettison, Flinders University, Adelaide, Australia. Dione Gardner-Stephen, Serval Project Inc., Adelaide, Australia. Matthew Lloyd, New Zealand Red Cross, Wellington, New Zealand. IEEE, 2013

[3] ***An Experimental Evaluation of Delay-Tolerant Networking with Serval***. Lars Baumgartner∗, Paul Gardner-Stephen†, Pablo Graubner∗, Jeremy Lakeman†, Jonas Höchst∗, Patrick Lampe∗, Nils Schmidt∗, Stefan Schulz∗, Artur Sterz∗, and Bernd Freisleben∗

*Department of Mathematics & Computer Science, University of Marburg, Germany

†School of Computer Science, Engineering & Mathematics, Flinders University, Australia

[4] ***Digital cellular telecommunications system (phase 2+); Technical realization of the Short Message Service (SMS) Point-to-Point (PP) (GSM 03.40) ; European Telecommunications Standards Institute ; 2001-12***

[5] Serval Project, 2013, '*Serval Project Home Page*', accessed 28 June 2018, <**http://www.servalproject.org**>

[6] Serval Project, 2013, '*The Serval Project wiki*', accessed 28 June 2018, <http://developer.servalproject.org/dokuwiki/doku.php>

[7] Serval Project, 2013, 'The Serval Project on GitHub', accessed 28 June 2018, < https://github.com/servalproject>

[8] SMSForum, 30 May 2002, '*The SMPP v3.4 Protocol Implementation guide for GSM/UMTS, Version 1.0*', accessed on 28 June 2018, < http://opensmpp.org/specs/smppv34_gsmumts_ig_v10.pdf>

[9] 3GPP, 1999, '*3rd Generation Partnership Project; Technical Specification Group Terminals; Interface protocols for the connection of Short Message Service Centres (SMSCs) to Short Message Entities (SMEs) (3G TS 23.039 version 2.0.0)*', accessed on 28 June 2018, <http://www.3gpp.org/ftp/tsg_t/tsg_t/tsgt_04/docs/pdfs/TP-99128.pdf>

[10] SMPP Developers Forum, 1999, '*Short Message Peer to Peer Protocol Specification v3.4'*, accessed on 28 June 2018, <http://docs.nimta.com/SMPP_v3_4_Issue1_2.pdf>

[11] Raul Tremsal, 2012, '*libsmpp34 – C library for SMPP 3.4'*, accessed on 28 June 2018, <https://github.com/osmocom/libsmpp34>

[12] Raul Tremsal, 2012, '*C open smpp-3.4 Library'*, accessed on 28 June 2018, <http://c-open-smpp-34.sourceforge.net/out-1.10/web/c-open-libsmpp34_en/index.html>

[13] ActiveXperts Software, N.D., '*Short Message Peer to Peer Protocol Specification v3.4'*, accessed on 28 June 2018, <https://www.activexperts.com/sms-component/smpp-specifications/pdu-type-format-definitions/>

[14] EFFORT, 2009, '*Short Message Service ; Principes et Architecture'*, accessed on 28 June 2018, <http://www.efort.com/r_tutoriels/SMS_EFORT.pdf>

[15] Martin Giess (EMnify), 2015, '*SMPP Integration Guide'*, accessed on 28 June 2018, <https://github.com/EMnify/doc/wiki/SMPP-Integration-Guide>

[16] OSMOCOM, N.D, '*Open Source Mobile Communications'*, accessed on 28 June 2018, <https://osmocom.org/>

[17] OSMOCOM, N.D, '*OsmoBTS'*, accessed on 28 June 2018, <https://osmocom.org/projects/osmobts>

[18] OSMOCOM, N.D, '*OsmoTRX*', accessed on 28 June 2018, <https://osmocom.org/projects/osmotrx>

[19] OSMOCOM, N.D, '*OsmoHLR*', accessed on 28 June 2018, <https://osmocom.org/projects/osmo-hlr>

[20] OSMOCOM, N.D, '*OsmoMSC*', accessed on 28 June 2018, <https://osmocom.org/projects/osmomsc>

[20] OSMOCOM, N.D, '*OsmoSTP*', accessed on 28 June 2018, <https://osmocom.org/projects/osmo-stp>

[21] OSMOCOM, N.D, '*OsmoBSC*', accessed on 28 June 2018, https://osmocom.org/projects/osmobsc

[22] Kannel Group, N.D, '*Kannel : Open Source WAP and SMS gateway*', accessed on 29 June 2018, <https://www.kannel.org/>

[23] StackOverFlow, N.D, '*Stack OverFlow – Where Developers Learn, Share, & Build Careers*', accessed on 28 June 2018, <https://stackoverflow.com/>