Engineering Internship Report

01/03/2015 ~ 01/09/2015

Application Development in the Area of Device Management(Home Automation Box)

Author:

FENG Siyao

A report submitted in fulfilment of the requirements for the degree of Engineer in Electronics and Computer Science

at

Polytech Paris-UPMC

August 2015

Supervisors

Industrial Supervisors:

Matthieu Anne

Ingénieur Développement, Orange/IMT/OLPS/SMA/DM2E/CARE

Tel : 04.76.76.42.20 E-mail : matthieu.anne@orange.fr

Academic Supervisor:

Andrea Pinna

Directeur de EI-SE

Université Pierre et Marie Curie - LIP6

Tel : 01.44.27.96.35 E-mail : andrea.pinna@lip6.fr

Acknowledgements

The internship opportunity I had with CARE group was a great chance for learning and professional development. It's here at Orange Labs I strengthened my willing to future research careers. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am sincerely grateful for having a chance to meet so many wonderful people and professionals who led me through this six-month internship.

First and foremost, I would like to express my sincere gratitude to my supervisor M. Matthieu ANNE, for the continuous support of my internship, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this report. I could not have imagined having a better superadvisor and mentor for my internship.

My sincere thanks also goes to M. Julien Roland, M. Marc Douet, who provided me their knowledge and experience and precious time, only with their help I can get better work enviroment.

Last but not the least, I would like to thank all the CARE team members. Thank you for the fabulous leisure time we had together!

Contents

List of Figures

Chapter 1

Introduction

TR-069 (Technical Report 069) is a technical specification published by the Broadband Forum and entitled CPE WAN Management Protocol (CWMP)[1]. It defines an application layer protocol for remote management of end-user devices. As a bidirectional SOAP/HTTP-based protocol, it provides the communication between customer-premises equipment (CPE) and Auto Configuration Servers (ACS). It includes both a safe auto configuration and the control of other CPE management functions within an integrated framework. The protocol addresses the growing number of different Internet access devices such as modems, routers, gateways, as well as end-user devices which connect to the Internet, such as set-top boxes, and VoIP-phones. The TR-069 standard was developed for automatic configuration and management of these devices by Auto Configuration Servers (ACS). TR-069 was first published in May 2004, with amendments in 2006, 2007, 2010, July 2011 to version 1.3. and November 2013 to version 1.4 (am5)

On 23 October 2014, the new Smart Home product of Orange –Homelive– has entered to the market. Homelive is a unique solution that can link to the connected objects in the home, allowing client to manage the appliances remotely. There are a range of connected devices: weather monitors, thermostats, light switches, sound and movement detectors, and smoke detectors. They are connected using the Z-Wave protocol. For Orange support team, managing and monitoring devices remotely is critical to reduce the maintenance fee.

The present report describes the work I have done during my six months' internship at Orange Labs. In order to reduce the maintenance fee of sending support engineer to client homes, my mission was to evaluate the adaptation of TR-069 in the Homelive in order to remotely manage devices using an ACS. The following part of this chapter at first gives a basic conception of the context, then my internship objectives are presented. In the last section, the outline of the report will be listed.

1.1 Context

1.1.1 Smart Home

Home automation [2] is the residential extension of building automation. It is automation of the home, housework or household activities. Home automation may include centralized control of lighting, HVAC (heating, ventilation and air conditioning), appliances, security locks of gates and doors and other systems, to provide improved convenience, comfort, energy efficiency and security.

The popularity of home automation has been increasing greatly in recent years due to much higher affordability and simplicity through smartphone and tablet connectivity. The concept of the "Internet of Things" has tied in closely with the popularization of home automation. There are currently many communication protocol[1] for home automation.

A home automation system [3] integrates electrical devices in a house with each other. The techniques employed in home automation include those in building automation as well as the control of domestic activities, such as home entertainment systems, houseplant and yard watering, pet feeding, changing the ambiance "scenes" for different events (such as dinners or parties), lighting control system, and the use of domestic robots. Devices may be connected through a home network to allow control by a personal computer, and may allow remote access from the internet. Through the integration of information technologies with the home environment, systems and appliances can communicate in an integrated manner which results in convenience, energy efficiency, and safety benefits.

Orange contribution in standards aims at preparing an enhanced experience for the end-user, with a simplified approach in terms of in-home connectivity of the devices. It also targets an interoperable infrastructure, through a reference smart home architecture

[1]Like Z-Wave, ZigBee, En-Ocean, etc.

attracting various application providers, because the variety of relevant applications is a key element for the smart home market to take-off. This implies the availability of unified, open APIs proposed to the developers that do not want to deeply study each specific way to access the functionalities from all possible underlying technologies. Harmonization of data models is also a key standardization objective for Orange to propose smart home services in a seamless and progressive way to the end-user.

1.1.2 Homelive Offer

Homelive is a smart-home convenience system that puts their home at the customer's fingertips when away. Using a single application accessed by a smartphone tablet or computer, Orange Homelive allows consumers to adjust settings and to control remotely all connected home appliances from any place in order to enhance home security, improve home comfort, and manage energy consumption in the home. Homelive includes support for a broad range of devices, and Orange is continuing to expand the types of services and devices supported. The home monitoring subscription service is available for 9.99 €per month, and sends alerts when devices such as motion detectors and smoke alarms are triggered to owners via SMS or email.

The Homelive pack is sold for 79 €($99.00) with a special cash back offer of 78 €offer is valid for all new Homelive customers. It includes a central unit and 3 devices — a smoke sensor, a door/window sensor and a motion sensor (Figure 1.1). To enjoy the features of the Homelive solution, the user pays only 9.99 €per month with a 12-month commitment. The subscription includes unlimited SMS alerts, continuity of service monitoring and alerts in case of internet failure (subject to Orange mobile coverage), dedicated technical support and storage of video and data for a month. The offer is not restricted to Orange customers. Homelive can be installed in any home in France with an active Internet connection.

1.1.3 Device Management

A growing number of devices are connected in the user's home network and require device management. Service providers are faced with the new challenge of managing

FIGURE 1.1: Homelive "Au Cas Où"

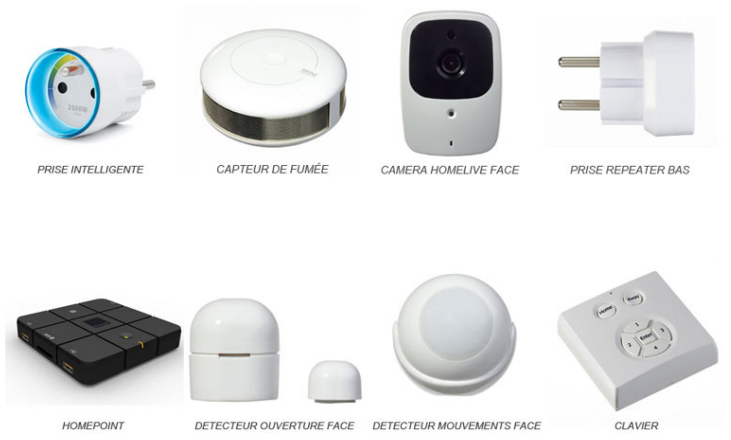


FIGURE 1.2: Homelive Connected Object

the increasing complexity of the home devices. Home Device Management addresses the technical device management and includes the following operations:

- *Setup and configuration of services (auto-provisioning),*
- *Managing connected devices (local or remote management): basic management, configuration management, software management, performance monitoring,*
- *Giving Service Providers more control (firmware upgrade),*
- *Cost reduction through remote management (avoid problem).*

Concerning the management of the connected devices, management could mainly be defined as follow:

- ***Basic Management :*** *Enable reboot, baseline reset, logging and basic test features such as ping, traceroute, nslookup and self-test action.*
- ***Configuration Management :*** *Includes Data Model access and setting, i.e. parameter setting and object creation.*
- ***Software Management :*** *Enable Deployment Unit, i.e., installation/uninstallation/update of software package; Execution Unit, i.e., active unit, start and stop.*
- ***Performance Monitoring :*** *Include time base mechanism to retrieve information on the device behavior, i.e., raw, threshold or aggregated.*

Software vendors proposes several solutions to achieve these goals based on proprietary or standardized protocols. There are several standardization initiatives but the Broadband Forum is the more interesting proposal to address remote Home Device Management needs from the operator point of view.

The core protocol CWMP (CPE Wan Management Protocol, also known as TR-069) is specified in the Broadband Forum. The data models are produced by the Broadband Forum based on other organization inputs. Standard organization or forum like HGI[2] or ETSI TISPAN[3] proposes implementation profiles requiring certain parameters as mandatory.

Nevertheless, TR-069 is not well adopted by manufacturers, especially consumer electronic vendors. To achieve local Home Device Management the UPnP Forum has created a Working Committee dedicated to Device Management namely: **UPnP Device Management**. Orange is currently co-chairing it with Samsung. This working committee is defining a local management protocol which is able to cover the remote management scope, i.e., from a local/inner point of view, and to manipulate Data Model definitions coming from other forum such as Broadband Forum or OMA[4].

[2]Home Gateway Initiative, smart home gateway provider.

[3]Telecoms & Internet converged Services & Protocols for Advanced Network.

[4]Open Mobile Alliance who defines mobile phone specifications & standards, enabling advancements in mobile across the globe.

1.2 Objective of Internship

Device management is a key asset for Orange used to upgrade the Livebox or Homelive firmware for new services deployment; it also used for VoIP activation and next generation liveradio & IPTV set-top box. Orange is operating a TR69 platform named Karma. Under the situation that Homelive is growing in the market, device management based on TR-069 is become critical for Orange.

The principal objective of this internship is divided into two parts:

1. *Deploy the TR-069 client onto the Homelive Box of Orange and implementer the RPC methods.*

2. *Integrate the new data model Z-Wave in the TR-069 Client, and evolve it work under the HardWare//Firmware of Homelive Box.*

1.3 Outline

In the first part(chapter 2) the enterprise Orange, and also the Orange Labs and the team CARE where I accomplish my internship will be introduced.

chapter 3 will cover the several most important technical backgrounds. It first explains the *CPE WAN Management Protocol (CWMP)* which defines several data models. Then is the Luup and MMS management platform which are provided by the main partner (Mios[5]) of Orange on the Smart Home project. Also the *NAT traversal* will be presented after Mios, it is a computer networking methodology with the goal to establish and maintain Internet protocol connections across gateways that implement *Network Address Translation (NAT)*. At last, the Homelive box will be analyzed, including the hardware analyze, operating system OpenWRT and Cross-Compiling between working machine and the embedded system OpenWRT.

The following chapter(chapter 4) focuses on global architecture of Homelive Management Platform. The procedure of the communication between *Auto Configuration Server*

[5]MiOS, LTD. is a global software and hardware company represented in over 60 countries, and focused on developing and distributing advanced control and monitoring solutions for the home and small enterprise markets.

(ACS) and *Customer-Premises Equipment (CPE)*will be described. Then, it goes over the position of my work in the whole project of the team.

chapter 5 explains my work on the first objective of this internship—TR-069 Client. Firstly, due to the highly modularization, the architecture of the program TR-069 client will be introduced. Then is the problems I meet during the implementation of TR-069 Client and the solutions I proposed. The result of completion of the RPC methods will be listed at last.

The final chapter 6 will present the second objective which is to integrate the data model Z-Wave in order to build the home network of connected objects and allow them to communicate with each other. The protocol and data model Z-Wave will be presented at first. Then is the procedure of synchronization of Z-Wave data model in Homelive firmware, TR-069 client and Automatic Configuration Server.

Chapter 2

Company Presentation

2.1 Presentation of Orange

Orange, formerly France Telecom[4], is a world-renowned French company specialized in the telecommunications sector. Orange has a rather eventful history due to multiple redemptions, but when France Telecom bought Orange PLC in August 2000, the group goes global. Orange has become the single brand of the group for the Internet, television and telephony (replacing Wanadoo, Itineris, Ola, Mobicarte ...). In 2006, Orange Business Services (OBS) appears to offer products and services to businesses and governments worldwide. At July 2013 that the France Telecom Group — Orange takes the official name of **Orange**[5].

2.1.1 An International Operator

Orange is the number three mobile operator and the number 1 ADSL television in Europe. The group is now present in 32 countries worldwide and strives to continue to deploy its foreign products and services. The operator carries much attention to the development of mobile services in Africa[6] and the Middle East where it totals 88 million customers at 31 December 2013 (approximately 37% of its customers). Some services Such "Orange Money" are deploying well abroad (8.9 million customers in Africa). Finally, since the arrival of 4G, Orange also invested in the coverage of European countries where customers are numerous (2 million customers in Poland and the UK in January 2014). Accordingly, Orange has 236 million customers worldwide (32 countries in 2014) while confused Service (76% in mobile client) and employs 165000 people. The band has annual major case (about 41 billion euros in 2013).

2.2 Presentation of Orange Labs

Innovation is a major growth lever for the Orange group. Research and development centers, also known as *Orange Labs* form the group's innovation base. Today, 3,500 experts working within 18 Orange locations in 10 countries. In 2013, Orange continued its efforts in research and innovation by allocating 1.9% of its turnover (780 million euros). Orange holds about 7,500 patents and apply 250 every year. Within the whole group, Orange employs 3700 people in research and development, and 200 PhD students and postdocs per year. The missions of research and development centers are:

- The development of new quality products and services for the group,
- The release of new sources of growth with strong potential,
- The anticipation of technical and technological developments,
- The imagination of the solutions of the future with anticipation of long-term issues.

Orange is also widely involved in the many technical activities such as in standardization groups (e.g. 3GPP, IEEE or HGI) to be a major player in worldwide innovation.

2.3 Structure

The company Orange is structured around various entities each with a specific role and specific occupations. Thus, the group consists of management entities, human resources management, communication, design, etc. ... The entity Innovation, Marketing and Technologies (IMT) is responsible for giving a body to the operator in Internet era. Thus, IMT aims to digitize the operator in his close customer behavior, renewal of the various infrastructures and open the operation base station on the outside, based on innovation. This entity thus relies on innovation to differentiate themselves from the competition on marketing, to provide simple and reliable services and technologies and enrich its infrastructure.

IMT within various specialized research centers (Orange Labs Research), network (Orange Labs Networks), services (Orange Labs Products and Services), marketing (Technocentre), etc. ... are responsible for delivering products and services focused on clients and their uses.

Composed of more than 3,000 people, Orange Labs Products and Services (OLPS) has overall technical responsibility for the products and services offered by Orange, the maintenance of implemented solutions in the world. Thus OLPS decomposes services that are more specific, such as SMA (SMart Access) which specializes in the domestic environment, SOFT which is specialized in the development of software components or UCE (Users and Customer Experience), which specializes in customer relationship. Each of these new services to decompose into smaller entities.

The chart below briefly presents how is structured Orange. There is more detailed for entities specialized in the design of new products and services, especially in the domestic field. The last level is the team working on the offer Smart Home Orange.

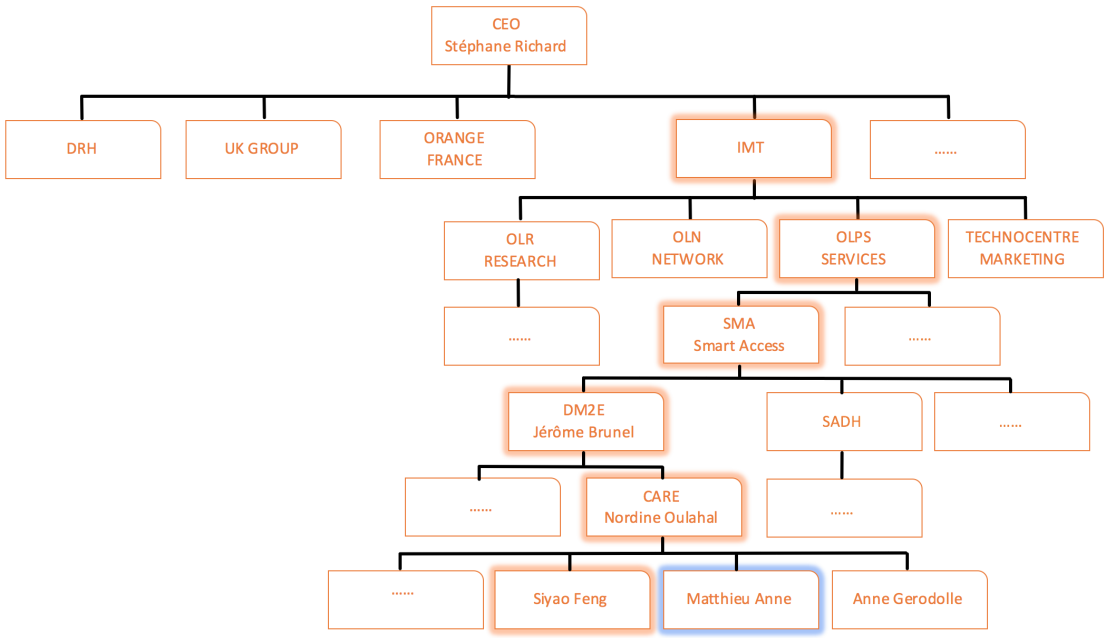


FIGURE 2.1: Partial flow chart showing the structure of Orange

2.4 Team CARE

I was integrated in the team CARE during my internship, which is for *Cloud enablers for Administration of Residential Equipment*, the main purpose are:

- Promote a consistent technical strategy between Tools and Devices on “I need help” process,
- Improve Orange customer satisfaction and reduce the cost of Broadband Services operation.

The Scope of “CARE” technical domain are:

- Broadband Services
  - Expertise and management of in-life Home Devices problems and bugs fixing with manufacturers
  - Delivery and generalization of Home Devices corrective and functional (enhanced) versions
  - Development of Home Devices diagnostic tools for Orange shops and for Customers
- Mobile Services
  - Collect, monitor and measure Mobile network quality and services (Qualimetre, CEM/CBM)

Chapter 3

Background

In this chapter, some important technical backgrounds will be covered. First it explains the *CPE WAN Management Protocol (CWMP)* which defines several data models. Then is the Luup and MMS management platform which are provided by the main partner of Orange on the Smart Home project. Also the *NAT traversal* will be presented after Mios, it is a computer networking methodology with the goal to establish and maintain Internet protocol connections across gateways that implement *network address translation (NAT)*.At last, the Homelive box will be analyzed, including the hardware analyze, operating system OpenWRT and Cross-Compiling between working machine and the embedded system OpenWRT.

3.1 CWMP

CPE WAN Management Protocol (CWMP) is a technology specification initiated and developed by the Digital Subscriber's Line DSL Forum (now Broadband Forum). CWMP is numbered TR-069 by the forum and is thus also called the TR-069 protocol. It defines the general frame, message format, management method, and data model for the management and configuration of home network devices in next-generation networks.

CWMP is mainly applied to DSL access networks, which are hard to manage because user devices are located at the customer premise, dispersed, and large in number. CWMP makes the management easier by using an auto-configuration server (ACS) to perform remote centralized management of customer premises equipment (CPE).

3.1.1 CWMP Protocol

CWMP is a text based protocol. Orders sent between the device (CPE) and auto configuration server (ACS) are transported over HTTP (or more frequently HTTPS)(Figure 3.1). At this level (HTTP) CPE is behaving in the role of client and ACS in the role of HTTP server. This essentially means that control over the flow of the provisioning session is the sole responsibility of the device.

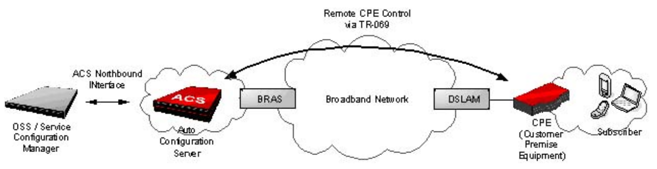

Remote CPE Control via TR-069
ACS Northbound INterface
OSS / Service Configuration Manager
ACS
Auto Configuration Server
BRAS
Broadband Network
DSLAM
CPE (Customer Premise Equipment)
Subscriber

FIGURE 3.1: Remote CPE Control via TR-069

3.1.1.1 Provisioning session

All communications and operations are performed in the scope of the provisioning session. The session is always started by the device(CPE) and begins with the transmission of an Inform message. Its reception and readiness of the server for the session is indicated by an InformResponse message. That concludes the session initialization stage. The order of the next two stages depends on the value of the flag HoldRequests. If the value is false the initialization stage is followed by the transmission of device requests, otherwise ACS orders are transmitted first. The following description assumes the value is false.

In the second stage, orders are transmitted from the device to the ACS. Even though the protocol defines multiple methods that may be invoked by the device on the ACS, only one is commonly found - TransferComplete which is used to inform the ACS of the completion of a file transfer previously issued Download or Upload request. This stage is finalized by transmission of empty HTTP-request to the ACS.

In the third stage the roles change on the CWMP level. The HTTP-response for the empty HTTP-request by the device will contain a CWMP-request from the ACS. This will subsequently be followed by an HTTP-request containing a CWMP-response for the previous CWMP-request. Multiple orders may be transmitted one-by-one. This stage (and the whole provisioning session) is terminated by an empty HTTP-response from the ACS indicating that no more orders are pending.

3.1.1.2 Security and authentication

As vital data (like user names and passwords) may be transmitted to CPE via CWMP, it is essential to provide secure transport channel and always authenticate the CPE against the ACS. Secure transport and authentication of the ACS identity can easily be provided by usage of HTTPS and verification of ACS certificate. Authentication of the CPE is more problematic. The identity of the device is verified based on a shared secret (password) at the HTTP level. Passwords may be negotiated between the parties (CPE-ACS) at every provisioning session. When the device contacts the ACS for the first time (or after a factory-reset) default passwords are used. In large networks it is the responsibility of the procurement to ensure each device is using unique credentials, their list is delivered with the devices themselves and secured.

3.1.1.3 Connection request

The following exampleFigure 3.2 illustrates how CWMP works. The scenario: There are two ACSs, active and standby in an area. The active ACS needs to restart for system upgrade. To ensure a continuous monitoring of the CPE, the active ACS needs to let all the CPE in the area connect to the standby ACS. The whole process is as follows:

1. Establish a TCP connection
2. SSL initiation, and establish a security mechanism
3. The CPE sends an Inform request message to initiate a CWMP connection. The Inform message carries the reason for sending this message in the Eventcode field. In this example, the reason is “6 CONNECTION REQUEST”, indicating that the ACS requires the CPE to establish a connection.
4. If the CPE passes the authentication of the ACS, the ACS returns an Inform response, and the connection is established.

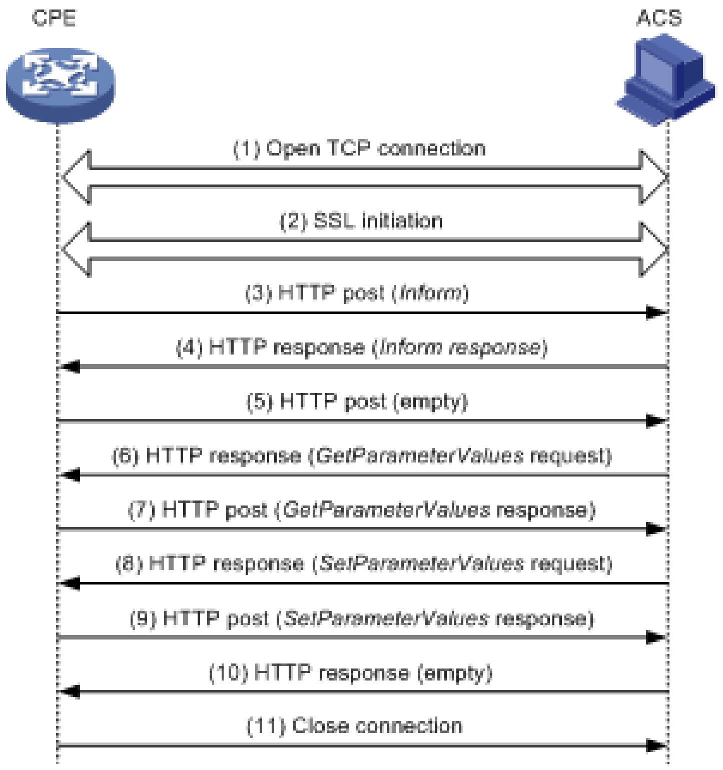

CPE
ACS
(1) Open TCP connection
(2) SSL initiation
(3) HTTP post (*Inform*)
(4) HTTP response (*Inform response*)
(5) HTTP post (empty)
(6) HTTP response (*GetParameterValues* request)
(7) HTTP post (*GetParameterValues* response)
(8) HTTP response (*SetParameterValues* request)
(9) HTTP post (*SetParameterValues* response)
(10) HTTP response (empty)
(11) Close connection

FIGURE 3.2: CWMP Connection Scenario

5. Receiving the Inform response, the CPE sends an empty message if it has no other requests. The CPE does this in order to comply with the request/reply interaction model of HTTP, in which CWMP messages are conveyed.
6. The ACS queries the value of the ACS URL set on the CPE.
7. The CPE replies the ACS with the obtained value of the ACS URL.
8. The ACS finds that its local URL value is the same as the value of the ACS URL on the CPE. Therefore, the ACS sends a Set request to the CPE to modify the ACS URL value of the CPE as the URL of the standby ACS.
9. The setting succeeds and the CPE sends a response.
10. The ACS sends an empty message to notify the CPE that it has no other requests.
11. The CPE closes the connection.

After this, the CPE will initiate a connection to the standby ACS.

3.1.1.4 CR over NAT

The CWMP protocol also defines a mechanism for reaching the devices that are connected behind NAT (e.g. IP-Phones, Set-top boxes). This mechanism, based on STUN and UDP NAT traversal, is defined in document TR-069 Annex G (formerly in TR-111).

Amendment 5 of the protocol introduces alternative method of executing Connection Request via NAT based on XMPP (see Annex K of TR-069 Amendment 5 for details).

3.1.2 Data Model

The Broadband Forum defines several data models for use with the CPE WAN Management Protocol (TR-069 Amendment 5). These data models contain objects and parameters that describe the many different functions and capabilities available to devices and services that are manageable via CWMP.

CWMP data models are divided into two types: Root and Service. The root data model, Device1, is used to describe the major functions of a network aware device, including interfaces, software/firmware, diagnostics, components common to CWMP and other services, and the basic device information necessary to CWMP.

Service data models describe modular functionality that allow the extension of the root data model on a device (under Device.Services.) to provide particular services, such as a voice service, set top box service, network attached storage, etc.

Each data model is defined by a Name:Version syntax. A device defines its data model by defining a device type, an XML document that maps to (imports) BBF official data model objects and/or vendor specific objects. A full explanation of how to develop compliant CWMP data models can be found in TR-154.

Most of the configuration and diagnostics is performed through setting and retrieving the value of the device parameters. These are organized in a well defined hierarchical structure that is more or less common to all device models and manufacturers. Broadband Forum publishes its data model standards in two formats - XML files containing a detailed specification of each subsequent data model and all of the changes between their versions and PDF files containing human-readable details. Supported standards and extensions should be clearly marked in the device data model. This should be in the field Device.DeviceSummary or InternetGatewayDevice.DeviceSummary which is required starting from Device:1.0 and InternetGatewayDevice:1.1 respectively. If the field is not found InternetGatewayDevice:1.0 is implied. As of Device:1.4 and InternetGatewayDevice:1.6 new field (¡RO¿.SupportedDatamodel) for supported standard specification was introduced.

The model is always rooted in the single key named Device or InternetGatewayDevice depending on the manufacturer's choice. At each level of the structure objects and parameters (or array-instances) are allowed. Keys are constructed by concatenating the names of objects and parameter using '.'(dot) as a separator, e.g. InternetGatewayDevice.Time.NTPServer1 .

Each of the parameters may be marked as writable or non-writable. This is reported by the device in GetParameterNamesResponse message. The device should not permit the change of any parameter marked as read-only. Data model specifications and extensions clearly mark required status of most of the parameters.

Values applicable for the parameter, their type and meaning are also precisely defined by the standard.

Multi-instance objects :Some parts of the data model require the existence of multiple copies of the subtree. The best examples are those describing tables, e.g. Port Forwarding Table. An object representing an array will only have instance numbers or alias names as its children.

A multi-instance object may be writable (enabling dynamic creation and/or removal of its children) or read-only depending on the data represented. If for example the object represents four physical ports on a switch it should not be possible to add or remove them from the data model. If an instance is added to an object an identifier is assigned. After being assigned, identifiers may not change during the life-cycle of the device except for factory-reset.

3.2 Mios

Mios, LTD. is a global software and hardware company represented in over 60 countries, and focused on developing and distributing advanced control and monitoring solutions for the home and small enterprise markets. Founded in 2008, Mios has created the technology platform that bridges many different devices to produce hardware and software solutions for home control networks. Now in its fifth generation, the Mios platform allows users to remotely control, monitor and automate their households and businesses with products that are currently available from any vendor.

At January 6th, 2014. Mios has been selected as the platform partner for Orange's Smart Home initiative. This announcement followed a successful launch with Mios as the platform for Orange Poland's Smart Home program. The Orange initiative will be initially offered in several countries in Europe.

3.2.1 Luup

Luup (Lua-UPnP) is Mi Casa Verde(Mios)'s new software engine which incorporates *Lua*, a popular scripting language, and *UPnP*, the industry standard way to control devices. Vera is built on Luup.

On this platform, since the API includes drivers for Z-Wave, Insteon, etc., and talks to infrared and serial devices, home automation is the first thing that we can do on Luup. However, Lua is a full-featured language and not limited just to scripting. Also, we can write modules in C/C++ that run on Vera, which the main Luup engine will also aggregate and control.

3.2.2 MMS

MMS is the management platform based on http request. To be able to use the MMS, we have to authenticate to a valid user of an account:

Here is a login example:

```
    https://orangefut autha1.com/autha/auth/username/bob?SHA1Password=86e739edcc&
        PK_Oem=33
```

And the body of the response (Identity and IdentitySignature are truncated):

```
"Identity": "eyJFeHBpcmVzIjoxNDAw...YW5nZV9tYXN0ZXIifQ==",
"IdentitySignature": "L5H6babTjsZwk7RiAeRP...W91tWSMIbI9x7jJmsQ==",
"Server_Event": "orangefut event12.com",
"Server_Event_Alt": "orangefut event11.com",
"Server_Account": "orangefut account12.com",
"Server_Account_Alt": "orangefut account11.com"
```

3.3 NAT Traversal

NAT traversal is a computer networking methodology with the goal to establish and maintain Internet protocol connections across gateways that implement network address translation (NAT). NAT breaks the principle of end-to-end connectivity originally envisioned in the design of the Internet.

NAT traversal techniques are required for certain client-to-client network applications, such as peer-to-peer file sharing and Voice over IP.[7]

3.3.1 Principals

Many techniques exist, but no single method works in every situation since NAT behavior is not standardized. Many NAT traversal techniques require assistance from a server at a publicly routable IP address[8]. Some methods use the server only when establishing the connection, while others are based on relaying all data through it, which adds bandwidth costs and increases latency, detrimental to real-time voice and video communications.

NAT devices are commonly used to alleviate IPv4 address exhaustion[9] by allowing the use of private IP addresses on home and corporate networks behind routers with a single public IP address facing the public Internet. The internal network devices communicate with hosts on the external network by changing the source address of outgoing requests to that of the NAT device and relaying replies back to the originating device.

The NAT traversal techniques available are as following:

- *Socket Secure (SOCKS)* is a technology created in the early 1990s that uses proxy servers to relay traffic between networks or systems.
- *UPnP IGD* is supported by many small NAT gateways in home or small office settings.

- *Interactive Connectivity Establishment (ICE)* is a technique used in VoIP, peer-to-peer communications, video, instant messaging, and other interactive media applications. It uses Session Traversal Utilities for NAT (STUN).
- *Application-level gateway (ALG)* is a component of a firewall or NAT that allows for configuring NAT traversal filters.[10]
- *NAT hole punching* is a technique that exploits how NATs handle some protocols (for example, UDP, TCP, or ICMP) to allow previously blocked packets through the NAT.

At Orange Labs, we use *UPnP IGD* and *STUN*to manage NAT traversal.

3.3.2 UPnP IGD

Universal Plug and Play (UPnP)[11] is a set of networking protocols that permits networked devices, such as personal computers, printers, Internet gateways, Wi-Fi access points and mobile devices to seamlessly discover each other's presence on the network and establish functional network services for data sharing, communications, and entertainment. UPnP is intended primarily for residential networks without enterprise-class devices.

The UPnP technology is promoted by the UPnP Forum, a computer industry initiative to enable simple and robust connectivity to stand-alone devices and personal computers from many different vendors. The Forum consists of over eight hundred vendors involved in everything from consumer electronics to network computing.

The concept of UPnP is an extension of plug-and-play, a technology for dynamically attaching devices directly to a computer, although UPnP is not directly related to the earlier plug-and-play technology. UPnP devices are "plug-and-play" in that when connected to a network they automatically establish working configurations with other devices.

Internet Gateway Device (IGD) Standardized Device Control Protocol is a protocol for mapping ports in network address translation (NAT) setups, supported by a certain number of NAT-enabled routers.[12] It is a common communications protocol of automatically configuring port forwarding, and is part of an ISO/IEC Standard [13] rather than an Internet Engineering Task Force standard.

Applications using peer-to-peer networks, multiplayer gaming, and remote assistance programs need a way to communicate through home and business gateways. Without IGD one has to manually configure the gateway to allow traffic through, a process which is error prone and time consuming. Universal Plug and Play (UPnP) comes with a solution for network address translation traversal.

IGD makes it easy to do the following:

- Learn the public (external) IP address
- Requesting for a new public IP address[14]
- Enumerate existing port mappings
- Add and remove port mappings
- Assign lease times to mappings

3.3.3 STUN

STUN (Session Traversal Utilities for NAT) is a standardized set of methods and a network protocol to allow an end host to discover its public IP address if it is located behind a NAT. It is used to permit NAT traversal for applications of real-time voice, video, messaging, and other interactive IP communications. It is documented in RFC 5389[15]. The STUN URI scheme is documented in RFC 7064. STUN is intended to be a tool to be used by other protocols, such as ICE.

The STUN protocol allows applications operating behind a network address translator (NAT) to discover the presence of the network address translator and to obtain the mapped (public) IP address (NAT address) and port number that the NAT has allocated for the application's User Datagram Protocol (UDP) connections to remote hosts. The protocol requires assistance from a third-party network server (STUN server) located on the opposing (public) side of the NAT, usually the public Internet.

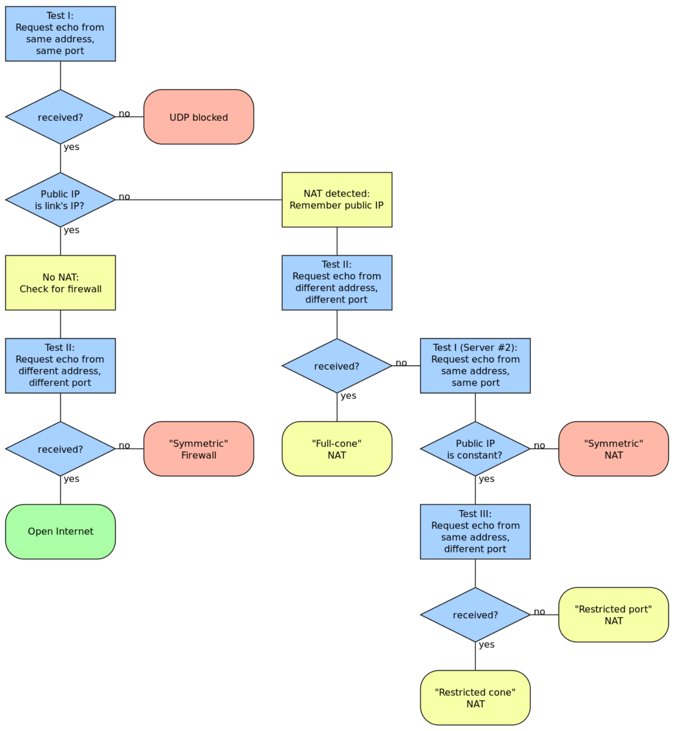

Test I:
Request echo from same address, same port
received?
no
UDP blocked
yes
Public IP is link's IP?
no
NAT detected: Remember public IP
yes
No NAT: Check for firewall
Test II: Request echo from different address, different port
Test II: Request echo from different address, different port
received?
no
"Symmetric" Firewall
yes
Open Internet
received?
no
Test I (Server #2): Request echo from same address, same port
yes
"Full-cone" NAT
Public IP is constant?
no
"Symmetric" NAT
yes
Test III: Request echo from same address, different port
received?
no
"Restricted port" NAT
yes
"Restricted cone" NAT

FIGURE 3.3: STUN Algorithm to track the Public Address

3.4 Homelive Box

Homelive is the Smart Home set-top box of Orange, which is the center of the Z-Wave network.

3.4.1 Hardware

FIGURE 3.4: Homelive Box of Orange

Here is the list of Homelive components:

Item	Value
Architecture	MIPS64
Operating System	OpenWRT BarrierBreaker 14.07
Firmware	Luup (Lua & UPnP)
Processeur	MT7620A (580 MHz)
Capacité mémoire	128 MB DDR2, 128 MB capacité flash NAND
Connectivité Cellulaire	2G quad bandes
Connectivité RF	Z-Wave Serie 500
Connectivité Wi-Fi	2.4 GHz 11n 2x2
Connectivité réseau	1 port RJ45 10/100
Connectivité multimédia	1 port USB2.0
Carte Sim	1 Slot externe
Dimensions	174 mm x 129 mm x 33 mm (L x l x h)
Poids	421 g
Alimentation externe	12 V
Consommation électrique	3 W

The MIPS64 architecture is based on a fixed-length, regularly encoded instruction set, and it uses a load/store data model. It is streamlined to support optimized execution of high-level languages. Arithmetic and logic operations use a three-operand format, allowing compilers to optimize complex expressions formulation. Availability of 32 general-purpose registers enables compilers to further optimize code generation by keeping frequently accessed data in registers.

3.4.2 OpenWRT

OpenWRT[16] is a highly extensible GNU/Linux distribution for embedded devices (typically wireless routers). Unlike many other distributions for these routers, OpenWRT is built from the ground up to be a full-featured, easily modifiable operating system for your router. In practice, this means that you can have all the features you need with none of the bloat, powered by a Linux kernel that's more recent than most other distributions. The main components are the Linux kernel, util-linux, uClibc or musl, and

BusyBox. All components have been optimized for size, to be small enough for fitting into the limited storage and memory available in home routers.

OpenWRT is configured[17] using a command-line interface (ash shell), or a web interface (LuCI). There are about 3500 optional software packages available for installation via the opkg package management system.

OpenWRT can run on various types of devices, including *CPE routers*, residential gateways, smartphones (e.g. Neo FreeRunner), pocket computers (e.g. Ben NanoNote), and laptops (e.g. One Laptop per Child (OLPC)). It is also possible to run OpenWRT on ordinary computers, which are most commonly based on the x86 architecture. Many patches from the OpenWRT codebase have been included upstream in the Linux kernel mainline.

3.4.3 Cross-Compiling

Cross compiling is the process that capable of creating executable code for a platform other than the one on which the compiler is running. For example, a compiler that runs on a Windows 7 PC but generates code that runs on Android smartphone is a cross compiler.

In our case, in the reason of the source of hardware in Homelive is very limited, we should finish the compilation of source code under the Linux Machine. To achieve this, we should get prepared with the compile tool chain which is provided by OpenWRT. The steps are:

1. Download OpenWRT package from official site
2. Compile OpenWRT on Linux machine and select the packages needed for Homelive Box
3. Generate the tool-chain of compilation
4. Compile the TR-069 source code with OpenWRT tool chain
5. Transfer executable files to Homelive

The disadvantage of cross compiling is that we cant́ use debug tools like *GDB*.

Chapter 4

HomeLive Management Platform

In this chapter, the Homelive Management Platform structure will be introduced. The following Figure 4.1 shows an global view for CWMP. With TR-069 protocol, ACS can communicate and manage devices easily.

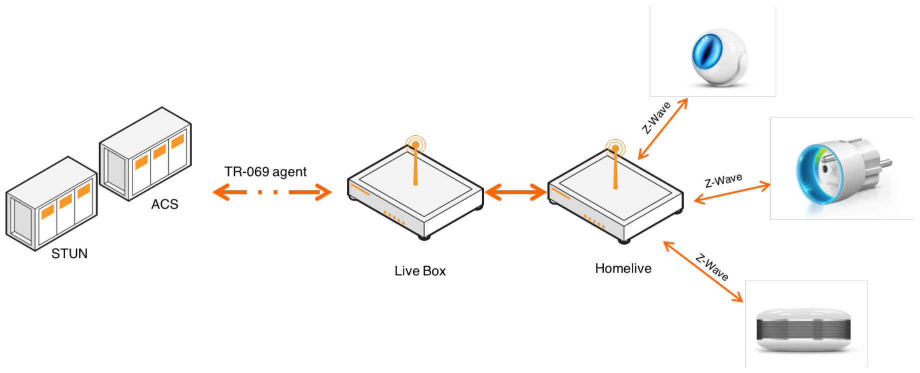


FIGURE 4.1: Homelive Management Platform

ACS is the server which is accessible by the support engineers of Orange. Support engineers are capable to build a connection remotely with Homelive box using protocol TR-069. The connection is build by two programs. The *TR-069 server* will be running in ACS as a server side and *TR-069 client* will be running in Homelive as the client side. This essentially means that control over the flow of the provisioning session is the sole responsibility of the device. Through TR-069, the high-level operations possible are:

- Service activation and reconfiguration
  - Initial configuration of the service as part of zero-touch or one-touch configuration process

  - Service re-establishment (ex. after device is factory-reset, exchanged)
- Remote Subscriber Support
  - Verification of the device status and functionality
  - Manual reconfiguration
- Firmware and Configuration Management
  - Firmware upgrade/downgrade
  - Configuration backup/restore
- Diagnostics and monitoring
  - Throughput (TR-143) and connectivity diagnostics
  - Parameter value retrieval
  - Log file retrieval

4.1 Auto Configuration Server

The ACS of Orange BU[1] and France is named Karma[2], the objective is to manage the Orange devices which are deployed in clients' home.

It is consisted by two PFS[3], one for the TV ecosystem(named Karma STB, only for France), the other one called Historique which is for Orange Livebox on France and for Orange Spain. The server is situated at:

- **Karma LB** situated at the IAC Lyon,
- **Karma STB** situated at the IAC Lille.

Karma provides the means to optimize Livebox or Homelive Box upgrades by targeting identified population groups, such as the customers of a new package, for example. Migrations are a more manageable size and conducted as and when. To make best use of the data held in the upgrade platform, Karma is interfaced with applications

[1]Business Unit
[2]Karma is an ancient Indian word means action, work, ordered. It also refers to the spiritual principle of cause and effect.
[3]PlateForm of Service

in the order-delivery process and after-sales service. This inventory function, Papyrus, provides support agents with valuable information on customers' software upgrades and Voice over IP activation (hardware type, software version, user behavior through the number of boots, etc.). Karma is based on international standards so that it can be used by all the Group's business units.

4.2 Client-Premise Equipment

On the client side of TR-069 is the CPE, which is Livebox and Homelive in our case. The Livebox is connected to Internet and serves as a gateway device, it is responsible to provide the connection between TR-069 server and client. Homelive box is the CPE in the structure, the process view is as Figure 4.2

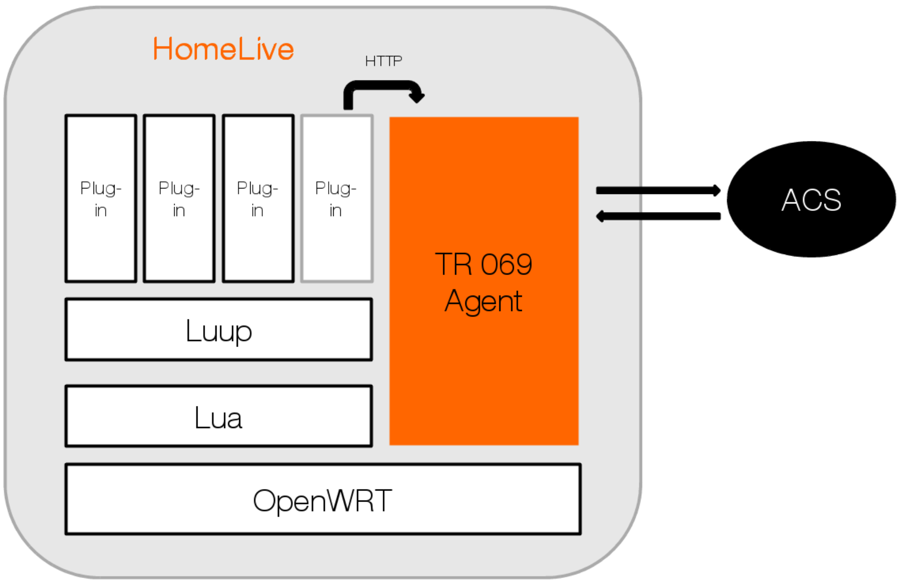


FIGURE 4.2: Homelive Process Structure

The OpenWRT is the operating system at the base, and beyond is Luup based on Lua. The Luup also provides us the way to create the plug-in, on which we can develop the API satisfying our personal needs. An API handle the http request is already integrated in the Luup to which we can push the HTTP Request to retrieve the Mios Data Model. Besides that, is our TR-069 Client process, the TR-069 Client process will push the request at every starting of the TR-069 session in order to synchronize the Mios Data Model and Z-Wave Data Model.

Chapter 5

TR-069 Client

TR-069 Client is implemented by Orange at 2008, but is a generic version for all potential devices. The first part of my internship is to make TR-069 Client for Homelive Box. To be able to provide a generic TR-069 module which is easily portable on different devices, it satisfy the following points:

- Written in ANSI C
- Small memory footprint
- Provide generic API to access device specific modules
- Provide Makefiles to build the binary
- Provide system traces on module activity

On a CPE, there are two modules that are mentioned all along this document and which are responsible of the CWMP:

- **TR-069 Agent**: This agent is responsible of the CWMP sessions. It initializes them with the inform message, realizes the ACS command(s) and execute some feedback command (notably after a firmware upgrade).
- **TR-069 Server**: This is a small HTTP server which listens of the WAN interface for an ACS solicitation (in TR-069, it is known as "connection request"). On a valid connection request, the TR-069 server contacts the TR-069 agent for starting a CWMP session with the ACS.

5.1 Architecture of TR-069 Client

The TR069 Generic Agent is composed of several modules and interfaces. Some modules are generic and can be ported with no modification. Other modules are platform specific (specific libraries usage, specific device API to get/set values, ...) and must implement services declared into generic interfaces.

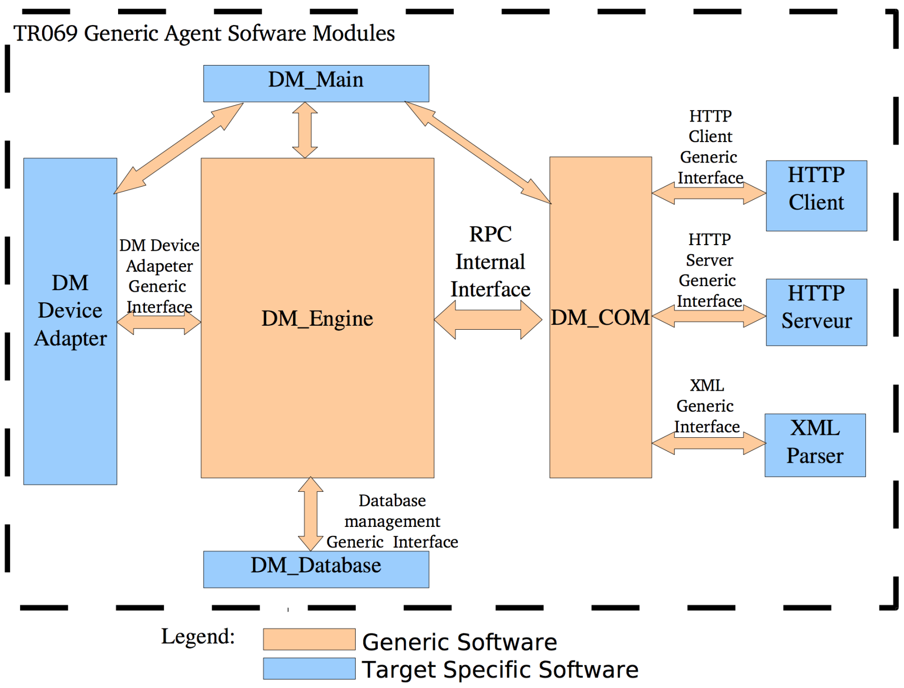


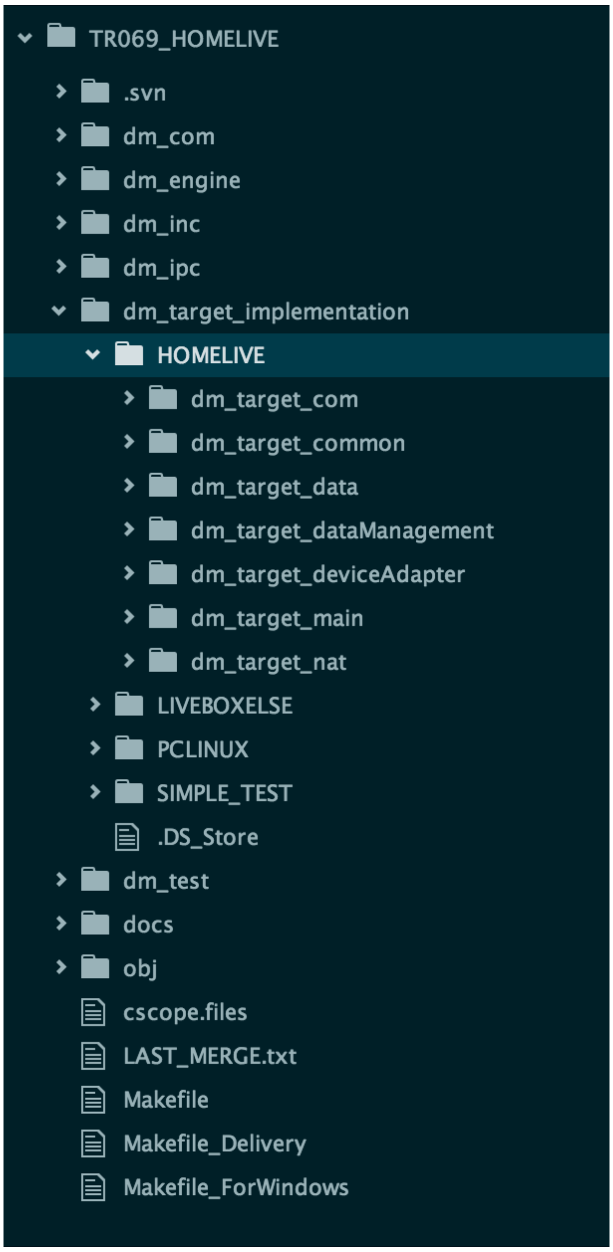
FIGURE 5.1: TR-069 Structure

For each module:

- **DM_ Engine** : In charge of the TR-069 logic,
- **DM_ Com** : Handles the HTTP, SOAP and SSL Protocol with the ACS (Auto Configuration Server),
- **DM_ DeviceAdapter** : Adaptation layer between the DM˙Engine and the device system. It allows the implementation of the RPC commands (Get Parameter Value, Set Parameter Value, Reboot, Download, ...),
- **DM_ Database** : Stores the data using an available storage solution on the device (or thanks to simple file storage),
- **HTTP Client** : Sets up / releases the HTTP connection with the ACS and sends / receives SOAP messages. The HTTP Client also handles SSL protocol,

- **HTTP SERVER** : Perform ACS Connection Request response,
- **XML Parser** : Decode / encode SOAP messages.

The figure below shows the file tree of the project, TR-069 Client defines all functions and APIs at each module. In consideration of easily ported to other devices, it put all target-related implementations in the folder *dm_target_implementation*.



```
TR069_HOMELIVE
    .svn
    dm_com
    dm_engine
    dm_inc
    dm_ipc
    dm_target_implementation
        HOMELIVE
            dm_target_com
            dm_target_common
            dm_target_data
            dm_target_dataManagement
            dm_target_deviceAdapter
            dm_target_main
            dm_target_nat
        LIVEBOXELSE
        PCLINUX
        SIMPLE_TEST
        .DS_Store
    dm_test
    docs
    obj
    cscope.files
    LAST_MERGE.txt
    Makefile
    Makefile_Delivery
    Makefile_ForWindows
```

FIGURE 5.2: TR-069 File Tree Structure

In the folder *dm˙taget˙implementation*, different target has been defined, you can implement the APIs and Modules for different devices, and you specifie the target to compile in compilation, like:

```
$ make Target=TargetName TRACE_LEVEL=7 DEBUG=Y
$ make clean Target=TargetName
```

A csv file must be created for each new platform. This CSV file describes the data model parameters used by TR069 Agent and the default values for the TR069 specifics parameters. This target dependent CSV file is stored into *dm_target_implementation/MYNEWDEVICE/dm_dm_target_data/dm_csv* directory.

The CSV file is only read during the first system start up (or after a factory reset) and is used to generate the TR069 database (i.e. parameters.data and parameters.data˜ files)

5.2 Problems and Solutions

When ported the TR-069 Client to Homelive, we have encountered several problems before it can perfectly run on Homelive Box.

5.2.1 IP Address Error

The first problem is IP Address Error. At every start up of the TR-069 Client, the module will have to detect the IP address of the box, and use it to build connection between STUN server and CPE. But when executing, the TR-069 client can't detect the IP address, so I have located the code which are responsible for IP detection,

```
static const char* ETH_INTERFACE = "eth0";
struct ifaddrs *myaddrs = NULL, *ifa = NULL;
struct sockaddr_in *s4 = NULL;
int  status;
/* but must be big enough for an IPv6 address (e.g. 3ffe:2fa0:1010:ca22:020a:95ff:fe8a:1cf8) */
char buf[64];
memset((void *) buf, 0x00, sizeof(buf));

status = getifaddrs(&myaddrs);
if (status == 0)
{
    for (ifa = myaddrs; ifa != NULL; ifa = ifa>ifa_next)
    {
        if ( (ifa>ifa_addr != NULL)
            && ((ifa> ifa_flags  & IFF_UP) != 0)
            && (ifa>ifa_addr>sa_family  == AF_INET) )
        {
            s4 = (struct sockaddr_in *)(ifa>ifa_addr);
            if ( (inet_ntop(ifa>ifa_addr>sa_family, (void *)&(s4>sin_addr), buf, sizeof(buf))
                != NULL)
               && (strcmp(ifa>ifa_name, ETH_INTERFACE)==0) )
            {
               ipAddress = strdup(buf);
               break;
            }
        }
    }
}
```

In general, the first Internet address should be *eth0*, but in Homelive Box, if you verify with command

```
$ ifconfig -a
```

The default Internet interface is ”br-lan”, which is specific in OpenWRT system, which is bridged Virtual Network Interface, used to make multiple virtual or physical network interfaces act as if they were just one network interface (quasi the opposite of VLANs).

So the solution is to change the macro definition of *ETH_INTERFACE* from *“eth0”* to *“br-lan”*. After the modification, the program can success finding the IP address.

5.2.2 STUN Initialization

When using the default data model provided, TR-069 Client can’t build connection with STUN server. By looking into the log file, the problem aims to no STUN data model has been settled. The solution is to add the items which describe STUN parameters as follow:

```
ManagementServer.UDPConnectionRequestAddress;STRING;0;0;1;2;1;0;;0;0;0
ManagementServer.UDPConnectionRequestAddressNotificationLimit;UINT;0;1;0;0;0;0;;0;0;0
ManagementServer.STUNEnable;BOOLEAN;0;1;0;0;1;0;;1;0;0
ManagementServer.STUNServerAddress;STRING;0;1;0;0;1;0;;161.105.161.211;0;0
ManagementServer.STUNServerPort;INT;0;1;0;0;1;0;;3478;0;0
ManagementServer.STUNUsername;STRING;0;1;0;0;0;0;;test;0;0
ManagementServer.STUNPassword;STRING;0;1;0;0;0;0;;1234;0;0
ManagementServer.STUNMaximumKeepAlivePeriod;INT;0;1;0;0;0;0;;400;0;0
ManagementServer.STUNMinimumKeepAlivePeriod;UINT;0;1;0;0;0;0;;10;0;0
ManagementServer.NATDetected;BOOLEAN;0;0;1;2;1;0;;0;0;0
```

5.2.3 Memory Alignment Error

After the modifications above, every time when running the TR-069 Client, after 2 seconds, the program crash and report *bus error*, a bus error is a fault raised by hardware, notifying an operating system (OS) that a process is trying to access memory that the CPU cannot physically address: an invalid address for the address bus, hence the name. In modern use on most architectures these are much rarer than segmentation faults, which occur primarily due to memory access violations: problems in the logical address or permissions.

Because the TR-069 is generated by cross-compiling, some platforms (in our case MIPS64) can only read or write ints from addresses that are an even multiple of 8 bytes, otherwise they segfault. Even the ones that can handle arbitrary alignments are slower dealing with unaligned data (they have to fetch twice to get both halves), so the compiler will often pad structures to align variables. Treating structures as a lump of data that can be sent to disk or across the network thus requires extra work to ensure a consistent representation.

In the source code, there are *struct* defined like:

```
typedef struct _DM_ENG_Parameter
{
    bool writable; //1 byte
    char* name; // 4 bytes
  int minValue; // 4 bytes

   ....

  struct _DM_ENG_Parameter* next;
} __attribute ((packed)) DM_ENG_Parameter;
```

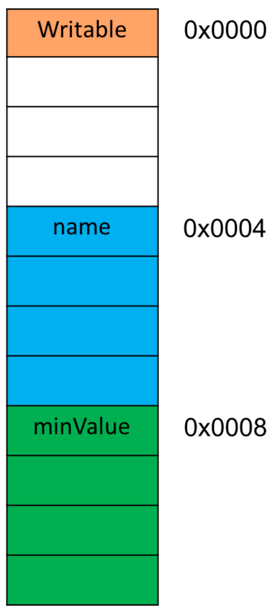


FIGURE 5.3: Memory Algnement under 32bits system

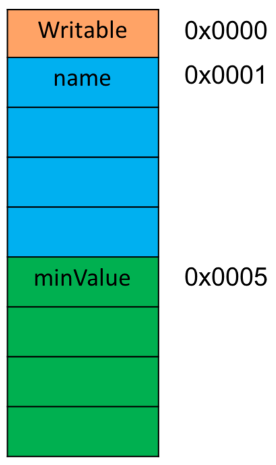


FIGURE 5.4: Memory Algnement under 32bits system with packed

In a 32bits processor, the first three elements of the structure will be aligned like Figure 5.3. But when using the attribute (packed), the memory will become Figure 5.4 to save the memory space.

Under the Homelive architecture, it can only read data from address that is multiple of 4, so when the program tries to read value of name, the address is 0x0001 which is not an valid address, it reports the *bus error*. To solve this bus error, the easiest way is to delete the attribute *packed*, and then the compilator will automatic shift the address to meet the requirements.

5.3 RPC methods

The next step is to verify and implement all the RPC methods, after verification:

RPC Method	Result
Connection Request	Yes
Get RPC Methods	Yes
Set Parameter Values	Yes
Get Parameter Values	Yes
Set Parameter Attributes	Yes
Get Parameter Attributes	Yes
Get Parameter Names	Yes
Add Object	Yes
Delete Object	Yes
Reboot	No
Downloaded	Yes
Factory Reset	Yes
Schedule Inform	Yes
Upload	Yes

When we call *reboot* from ACS, the TR-069 turns out only reboot the program but not the whole box. To repair this, after located the reboot function, we added one line at the end of the function to let the program call the box to reboot after 5 seconds (which is for the TR-069 to shutdown all the process).

```
/*Reboot system after 5 seconds*/
system("(sleep 5 && reboot) &");
```

Chapter 6

Z-Wave

Homelive use a new Smart Home communication technology – Z-Wave. TR-069 Client should be able to retrieve the data in Luup and synchronize with TR-069 data model and ACS.

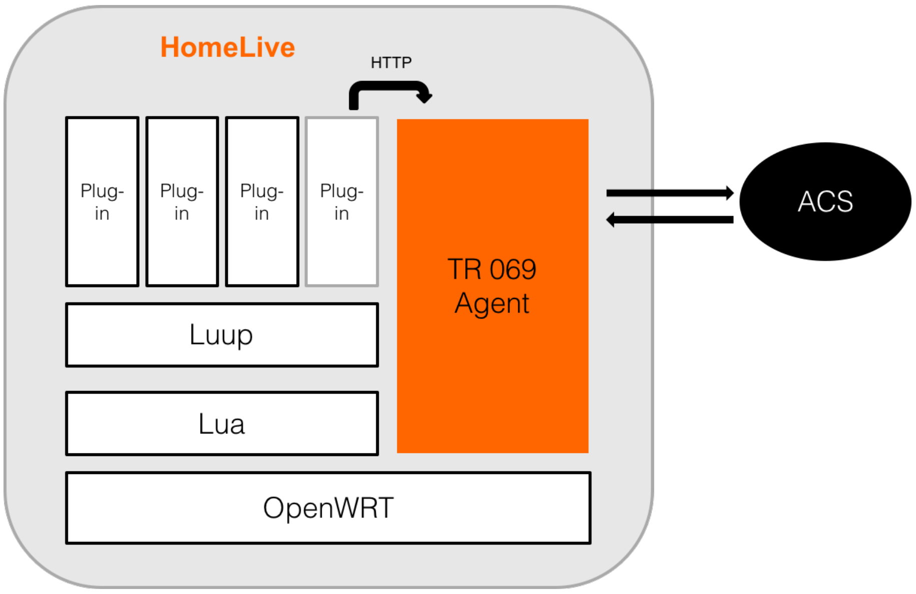


FIGURE 6.1: Homelive Structure

6.1 Protocol Z-Wave

Z-Wave is a standardized protocol for automation wireless habitat solution. It was designed by the Danish company *Zen-Sys* which was later bought by the US company

Sigma Designs in 2008. *Sigma Designs* and the Japanese company Mitsumi provides the Z-Wave chips.

Z-Wave equipment manufacturers are gathered in the Z-Wave Alliance. Founded in 2005, it promotes the protocol and ensures interoperability between devices. Interoperability is true on two levels: Radio layer and application layer. Certified equipment receive the Z-Wave logo (see Figure 6.2). More than three hundred companies have since joined this alliance.

FIGURE 6.2: Z-Wave Protocol Logo

The Z-Wave radio protocol is optimized to communicate with low bandwidth (between 9 and 40 kbps) and applied on stand-alone power supply or mains-powered, as opposed to Wi-Fi, which is intended for exchanges broadband.

Requirements\Protocol	**Z-Wave**	**Zigbee**	**En-Ocean**
Reliability	Yes	Yes	No
Security	Yes	Yes	No
Radio waves Reduction	Yes	Yes	Yes
Simplicity of use	Yes	-	No
Better pricing	Not yet	Not yet	No
Capitalization	Yes	-	Yes
Interoperability	Yes	No	Yes

6.2 Data Model Z-Wave

Orange has defined its own Z-Wave data model, which is divided into three parts:

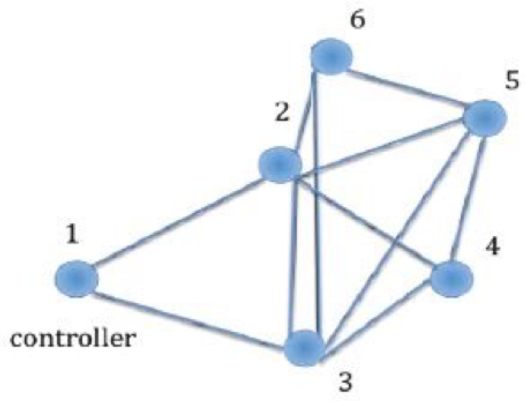


FIGURE 6.3: A Z-Wave Mesh Network Example

- The part *Z-Wave Network* defines the network topology
- The part *Z-Wave Structure* defines the set of information that can be recovered on the Z-Wave network to identify the devices.
- The part *Z-Wave Device* contains architecture and device configuration.

6.3 Update Z-Wave in TR-069

In order to retrieve the Z-Wave data from the Homelive Box, a http request to Homelive firmware can return a file (JSON or XML) which contains all the parameters that generated from lower layer. Then we should write a program to parser the file and save it in the TR-069 Client database.

6.3.1 HTTP Request

In addition to sending requests using standard UPnP, we can also do most things using a simple HTTP requests. Use the built-in URL data˙request, and pass the following on the URL: `http://ip_address:3480/data_request?id=user_data&output_format=xml`.

This returns the configuration data for Vera, which is a list of all devices and the UPnP variables which are persisted between resets as well as rooms, names, and other data the user sets as part of the configuration.

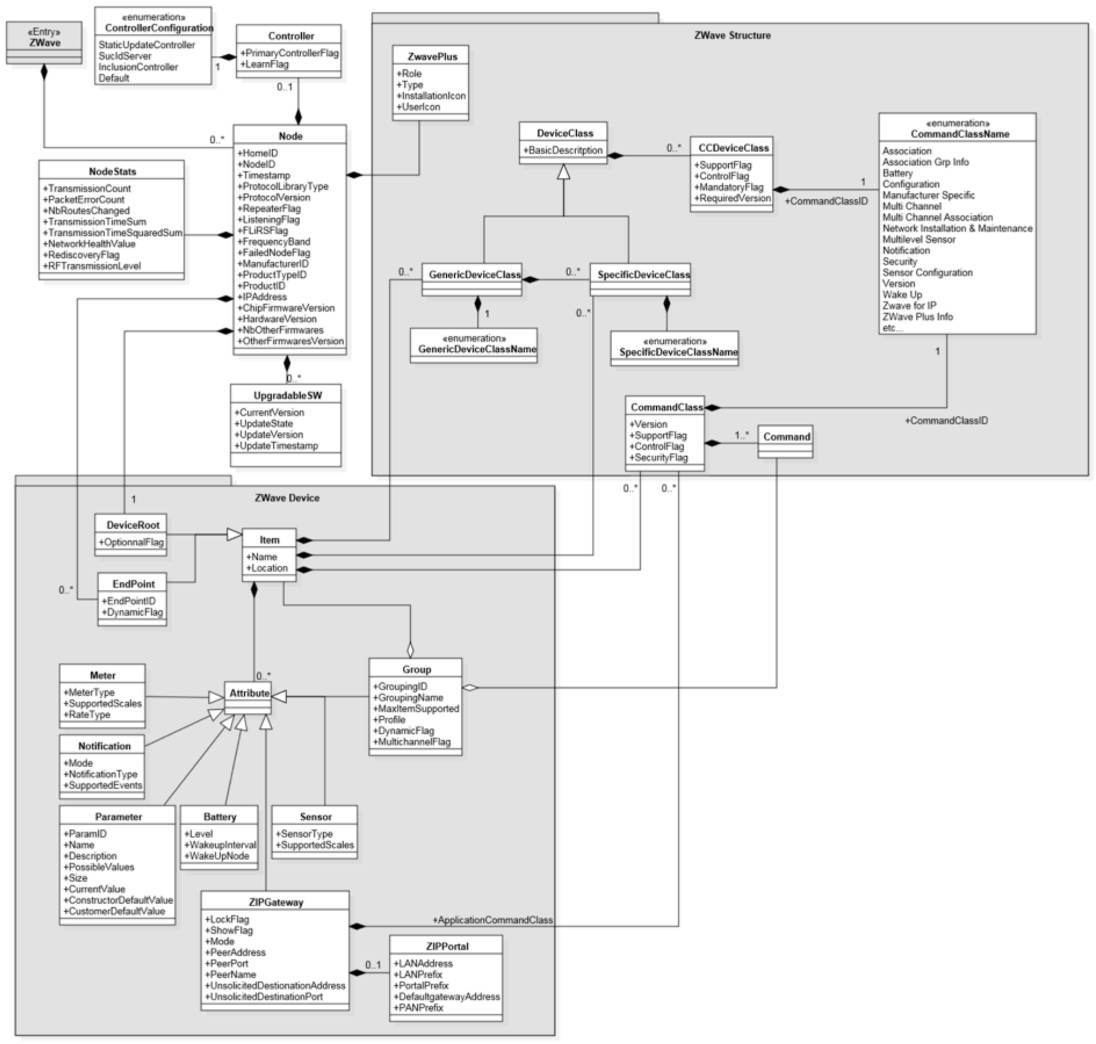


Figure 6.4: Z-Wave Node Data Model

In our case, we should retrieve the data of *status*, so we change *user˙data* to *status*. The http request is being called in the Homelive, so we can replace it with the local address: http://127.0.0.1/. And in consideration of parser, we set the output format to *JSON*. The request url become:

http://127.0.0.1:3480/data_request?id=status&output_format=json

We can call this http request by using the API already provided in TR-069 Client:

```
int DM_HttpGetFile(IN httpGetFileDataType * httpGetFileDataPtr)
```

After this, we will have the JSON file in format of Luup data model, the next step is to parse the file.

6.3.2 JSON Parser

JSON parser is not a default library of language C, so we have to find a JSON parser. Fortunately, in the Luup firmware, there is a JSON parser library already included, which is *json-c*. My program is based on *json-c* library and use a iterative structure to parse the JSON file we retrieved from Luup(See Appendices 2).

After parse the JSON file, we will have a huge name\value pairs, a two-dimension table is declared to saved them in buffer with dynamic memory management.

The next step is to find the correspondence between Z-Wave data model and Luup data model.

6.3.3 Simple file Storage System

TR-069 Client use simple file storage system as its data base. The structure is as following:

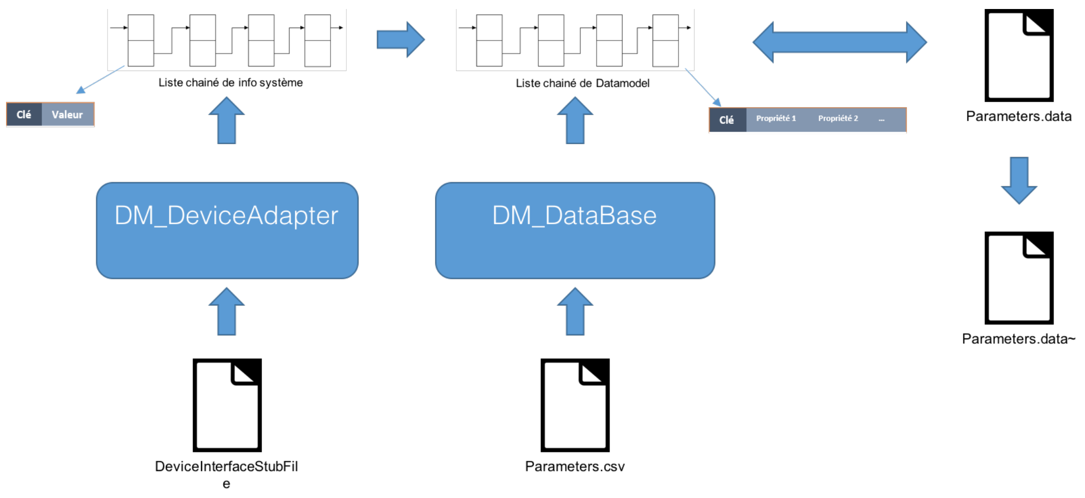


FIGURE 6.5: TR-069 Simple File Storage System

When the program starts, there are two modules are in charge of manage the database. First is the module *DM_ DeviceAdapter*, it reads the configuration file *DeviceInterfaceStubFile*, which contains the system parameters(Manufacturer, ProductClass, SerialNumber, etc..), and it will create a linked list to store the name\value pairs.

After that, module *DM_ Database* will read the data model file *parameters.csv* and build a linked list from it. In the *parameters.csv*, there are name\value of the data model and also 10 properties (Datatype, Writable, Notification type, etc..). When finish building the long linked list, it will read the value from the short linked list(*DeviceInterfaceStubFile*) and copy them into the long linked list.

In the runtime, a file called *parameters.data* will be created, and TR-069 will store all the data model in this runtime file in format of CSV (Comma-separated values) as the storage base.

The backup file is called *parameters.data˜*, before every modification made to data model, the *parameters.data* will save all the information into *parameters.data˜* as a backup and update its own value.

6.3.4 Synchronization Data model Luup and Z-Wave

The data model of Luup and Z-Wave have different in approach but equally contains the data we need. For each data in Z-wave data model, we should find the correspondence in Luup data model. For example, in Luup there is a data called *ZWave.devices.{i}.states.15.variable WakeupInterval*, the correspondence we found in Z-Wave is *Device.Zwave.Interface.{i}.WakeUpInterval.* To synchronize the two data model, the value of *WakeInterval* in Luup should be given to Z-Wave *WakeInterval.* We have studied 100 parameters in Z-wave, and there are 27 are directly accessible from Luup data model, 26 can be calculated from Luup data model. For instance, the translation is finished for those all.

Chapter 7

Conclusion

The past six months of my internship have been very instructive for me. It's the first time that I participated in an industrial research project. This opportunity allows me to learn and develop myself in many areas. I gained a lot of experience, especially in CWMP and Z-Wave Protocol. A lot of activities are familiar with what I have learned at school. They enhanced my competence in computer science and inspired my interests in industrial research work.

On the other hand, I also encountered problems that I had never come across before. It was good to find out what my weaknesses are. This helped me to define which skills and knowledges I have to improve in the coming study time.

One thing I appreciate a lot is the DM2E meeting organized by group members. They invite researchers specialized in one particular area to share their opinions and ideas with other people. Even if I didn't understand all discussions, I did get a global picture for my unknown topics from these presentations. Such experience makes me eager to learn more and explore more; it really broadened my horizons.

From this internship I also learned the importance of time management. Once I realized what I had to do, I organized my day and work so that I did not waste my hours. A Gantt diagram of my internship helps with the time management.

This six-month internship passed too quickly. I am grateful for meeting so many wonderful staff members. I will try my best to apply all I learned here into my future careers and sincerely give all my best regards to my group members.

Appendix A

Installation of OpenWRT and TR-069

This appendix will introduce how to getting and building OpenWRT toolchain, compile TR-069 Client and run the client on the Homelive Box.

A.1 Getting and building OpenWRT toolchain

A.1.1 Get OpenWRT Source

First to do is create your OpenWRT folder and get the source code of OpenWRT using git or SVN:

```
$ tsocks svn co -r44360 svn://svn.openwrt.org/openwrt/trunk/ OpenWRT
```

or

```
$ tsocks git clone git://git.openwrt.org/openwrt.git
```

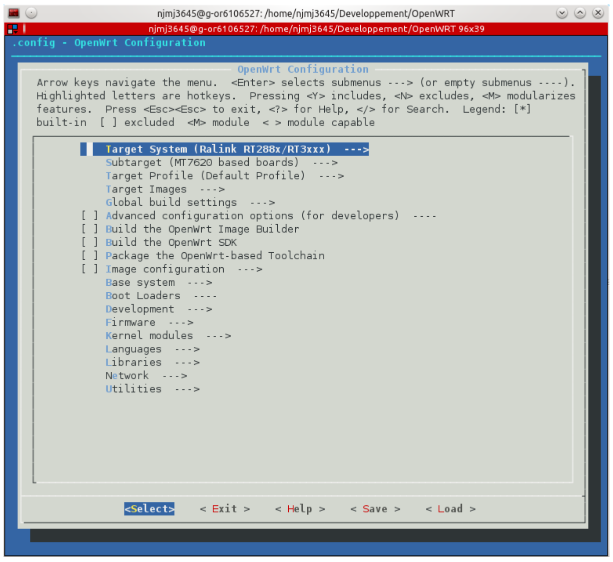
We choose the specific version 44360 because OpenWRT added unsupported libraries after this.

A.1.2 OpenWRT Buildroot Configuration

Then is to run configure interface to personalize your OpenWRT image:

```
$ make menuconfig
```

It will show a interface like below:



```
njmj3645@g-or6106527: /home/njmj3645/Developpement/OpenWRT
njmj3645@g-or6106527: /home/njmj3645/Developpement/OpenWRT 96x39
.config - OpenWrt Configuration
                              OpenWrt Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
  built-in  [ ] excluded  <M> module  < > module capable

          Target System (Ralink RT288x/RT3xxx)  --->
          Subtarget (MT7620 based boards)  --->
          Target Profile (Default Profile)  --->
          Target Images  --->
          Global build settings  --->
      [ ] Advanced configuration options (for developers)  ----
      [ ] Build the OpenWrt Image Builder
      [ ] Build the OpenWrt SDK
      [ ] Package the OpenWrt-based Toolchain
      [ ] Image configuration  --->
          Base system  --->
          Boot Loaders  ----
          Development  --->
          Firmware  --->
          Kernel modules  --->
          Languages  --->
          Libraries  --->
          Network  --->
          Utilities  --->

             <Select>    < Exit >    < Help >    < Save >    < Load >
```

FIGURE A.1: OpenWRT Menu Config Interface

- In the **Target System** section, select **Ralink RT288x/RT3xxx**,
- In the **Subtarget** section, select **MT7620 base boards**.
- In the **Libraries** section, select **libcurl**.
- In the **Libraries/SSL** section, select **libopenssl**.
- In the **Base system** section, select **lipthread**.

Then save the new configuration and exit menuconfig.

A.1.3 Build the OpenWRT Toolchain

```
$ tsocks make
```

If there are errors during the compilation. After solving the problem, you can continue the compilation process by using commands like:

```
$ tsocks make package/install
```

or

```
$ tsocks make package/libs/curl/install
```

The compiling process may take a while. When finished, your OpenWRT toolchain is ready to use.

A.2 Building TR-069 Client

A.2.1 Get the source code

Getting the source code by using the SVN of Orange:

```
$ tsocks svn co https://www.forge,orange-labs.fr/svnroot/tr069agent
```

Before using OpenWRT toolchain, there are some path to set:

```
$ export STAGING_DIR=$HOME/path/to/openwrt/staging_dir
$ export PATH=$PATH:$STAGING_DIR/toolchain-mipsel_24krec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin
```

Then, the compiling command is:

```
$ make Target=HomeLive CC=$STAGING_DIR/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc
    -0.9.33.2/bin/mipsel-openwrt-linux-uclibc-gcc CWMP_APPLICATION_NAME=cwmpd LIB_HEADER_INC=-
    I$STAGING_DIR/target-mipsel_24kec+dsp_uClibc-0.9.33.2/usr/include/ CWMP_USED_LIBRARY_FLAGS=
    "-L$STAGING_DIR/target-mipsel_24kec+dsp_uClibc-0.9.33.2/usr/lib/ -lcurl -lpthread -
    lpolarssl -ljsonc"
```

You can find exactable file *cwmpd* in ./obj.

A.3 Connection to the Homelive Box

A.3.1 Port status checking

First change your network to Livebox LAN. Then use nmap to find the Homelive address:

```
$ sudo nmap -sP 192.168.1.0/24
```

Then check if the Homelive HTTP and SSH port are open:

```
$ sudo nmap -Pn 192.168.1.10 //Homelive address
Starting Nmap 6.00 ( http://nmap.org ) at 2015-08-20 10:43 CET
Nmap scan report for pc5.home (192.168.1.10)
Host is up (0.0019s latency).
Not shown: 996 closed ports
PORT    STATE    SERVICE
22/tcp  filtered ssh
53/tcp  open     domain
80/tcp  filtered http
443/tcp filtered https

Nmap done: 1 IP address (1 host up) scanned in 40.73 seconds
```

To open the ssh and http port permanently, you can put a script in /root and add the following line to contab:

```
*/2 * * * * /root/permissions.sh >/dev/null 2>&1
```

This line will execute the script every 2 minutes.

A.3.2 Connect to Homelive

We will use SSH to connect Homelive box:

```
$ sudo ssh root@192.168.1.10:/root
```

A.4 Running cwmpd on Homelive

Copy the *cwmpd* bin file and *parameters.csv*, *DeviceInterfaceStubFile* using scp.

A.4.1 Library Dependencies

List of the needed libraries for cwmpd:

```
$ objdump -p cwmpd | grep NEEDED
NEEDED               libcurl.so.4
NEEDED               libpthread.so.0
NEEDED               libpolarssl.so.7
NEEDED               libgcc_s.so.1
NEEDED               libc.so.0
```

We need to get the correct version of libpolarssl: libpolarssl.so.3. A temporary hack is to define a symbolic link from libpolarssl.so.7 to libpolarssl.so.3. On the HomeLive:

```
$ cd /usr/lib
$ ln -s libpolarssl.so.3 libpolarssl.so.7
$ ls -l libpolarssl.so*
lrwxrwxrwx    1 root     root            16 Dec  3 10:49 libpolarssl.so -> libpolarssl.so.3
-rwxr-xr-x    1 root     root        223675 Feb 21 2014 libpolarssl.so.1.2.9
lrwxrwxrwx    1 root     root            20 Dec  3 10:49 libpolarssl.so.3 -> libpolarssl.so.1.2.9
lrwxrwxrwx    1 root     root            16 Jan 13 16:55 libpolarssl.so.7 -> libpolarssl.so.3
```

You can run cwmpd using:

```
$ ./cwmpd -p path/to/parametercsvfile/folder
```

Appendix B

JSON parser in C

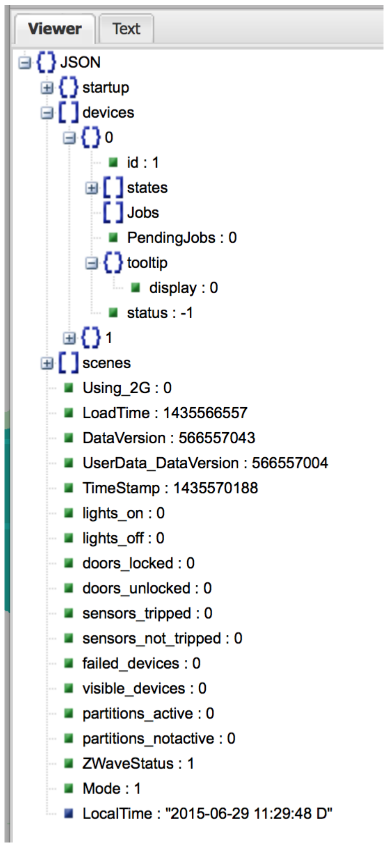
This appendix will explain how to implement the JSON parser in C language using library JSON-c.

B.1 Objective of JSON Parser

The objective of our JSON parser is to parser a JSON file generated by Luup HTTP Request. The website Online JSON Viewer can be used to visualize the JSON file. Below is a JSON example of our JSON file named **status.json**:

In order to import the data into TR-069, the JSON format should be convert to Z-Wave format which is like:

```
ZWave.devices.1.id 1
ZWave.devices.1.states.1.id 208
ZWave.devices.1.states.1.service urn:micasaverde com:serviceId:ZWaveNetwork1
ZWave.devices.1.states.1.value 1
ZWave.devices.1.states.2.id 209
ZWave.devices.1.states.2.service urn:micasaverde com:serviceId:ZWaveNetwork1
ZWave.devices.1.states.2.variable UseMR
ZWave.devices.1.states.2.value 1
ZWave.devices.1.states.3.id 210
ZWave.devices.1.states.3.service urn:micasaverde com:serviceId:ZWaveNetwork1
ZWave.devices.1.states.3.variable LimitNeighbors
ZWave.devices.1.states.3.value 0
ZWave.devices.1.states.4.id 211
```



```
JSON
  startup
  devices
    0
      id : 1
      states
      Jobs
      PendingJobs : 0
      tooltip
        display : 0
      status : -1
    1
  scenes
  Using_2G : 0
  LoadTime : 1435566557
  DataVersion : 566557043
  UserData_DataVersion : 566557004
  TimeStamp : 1435570188
  lights_on : 0
  lights_off : 0
  doors_locked : 0
  doors_unlocked : 0
  sensors_tripped : 0
  sensors_not_tripped : 0
  failed_devices : 0
  visible_devices : 0
  partitions_active : 0
  partitions_notactive : 0
  ZWaveStatus : 1
  Mode : 1
  LocalTime : "2015-06-29 11:29:48 D"
```

FIGURE B.1: JSON File in JSON Viewer Online

```
ZWave.devices.1.states.4. service  urn:micasaverde com:serviceId:ZWaveNetwork1
ZWave.devices.1.states.4. variable  LastDongleBackup
```

First part is the name and second is the value.

B.2 JSON Format Characteristic

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by: (colon) and the name/value pairs are separated by, (comma).

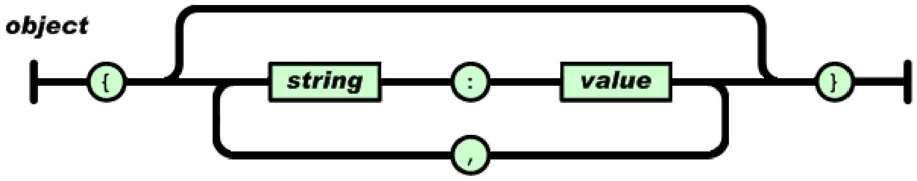


FIGURE B.2: JSON Object Structure

An array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by, (comma).

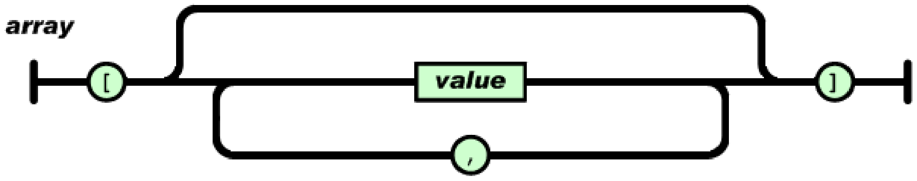


FIGURE B.3: JSON Array Structure

In the array case, the index must be added after the last level (e.g. ZWave.devices.1.states.2.id 209). The parser works in an iterative way.

B.3 Implementation of JSON Parser

At first, we set a buffer to store the path for each parameter. Then use the json-c library function *json_object_from_file* to save all the file content into a JSON object:

```c
char path_json[70] = "ZWave.";  //set the buffer to save the name of parameter, also as the head

struct json_object* jobj = json_object_from_file(fichierLu);  //fichierLu set to the file path to parse

_jsonParser(path_json, jobj);  //jobj is the json object will be parsed
```

_jsonParser is the private function to parse the JSON object. There is a essential json-c library function called *json_object_object_foreach(jobj, key, val)*, it allows to traverse each object in the JSON object. Before that, the current path should be added into a string.

```c
/*Parsing the json object*/
void _jsonParser(char *path, json_object * jobj)
{
    enum json_type type;

    char parse_path[100];                                   //define the local variable for path
    strcpy(parse_path,path);

     json_object_object_foreach(jobj, key, val)     /*Passing through every array element*/
    {
        type = json_object_get_type(val);
        switch (type)
        {
            case json_type_boolean:
            case json_type_double:
            case json_type_int:
            case json_type_string:  _printJsonValue(parse_path, val, key);
                                    break;

            case json_type_object:  sprintf(&parse_path[strlen(parse_path)], "%s.",key );
                                    jobj = json_object_object_get(jobj, key);
                                    _jsonParser(parse_path, jobj);
                                    break;

            case json_type_array:   _jsonParseArray(jobj, key, parse_path);
                                    break;
        }
    }
}
```

If the object parsed is a value, jump to the *_printJsonValue* function to save the value in buffer table. If it's a object, redo this function. If it's an array, go to the *_jsonParseArray* function.

```c
/*Parsing the json array*/
void _jsonParseArray( json_object *jobj, char *key, char *path)
{
    void _jsonParser(char *path, json_object * jobj); /*Forward Declaration*/
    enum json_type type;

    char array_path[100];                                    //define the local variable for path
    strcpy(array_path,path);
    json_object * jvalue;



    json_object *jarray = jobj; /*Simply get the array*/

    if(key) {jarray = json_object_object_get(jobj, key); /*Getting the array if it is a key value pair*/}

    int arraylen = json_object_array_length(jarray); /*Getting the length of the array*/
    int i;

    sprintf(&array_path[strlen(array_path)],"%s.",key);

    sprintf(&array_path[strlen(array_path)],"x.");    // fill the path with x. for instance

    for (i=0; i< arraylen; i++)
    {
        jvalue = json_object_array_get_idx(jarray, i); /*Getting the array element at position i*/
        type = json_object_get_type(jvalue);

        if( i <= 9 )
            sprintf(&array_path[strlen(array_path) 2], "%d.",i + 1 );   //replace the 2 last char x.
        else
            sprintf(&array_path[strlen(array_path) 3], "%d.",i + 1 );   //replace the 3 last char xx.



        if (type == json_type_array)
            _jsonParseArray(jvalue, NULL,array_path);
        else if (type == json_type_object)
            _jsonParser(array_path, jvalue);
        else
            _printJsonValue(array_path, jvalue, NULL);
    }
}
```

The main difficulty of parse a array is to add the index. The solution is to add .x at every beginning of Array parse, and then replace the .x according to different object type.

```c
/*At the end of each iteration , write the name/value pair to a 2D table*/
void _printJsonValue( char* path, json_object *jobj, char *key)
{
  enum json_type type;

  type = json_object_get_type(jobj); /*Getting the type of the json object*/

  char* value_buffer ;
   value_buffer = (char*)malloc(150 * sizeof(char));

  char* paramKey;
  paramKey = (char*)malloc((strlen(path)+strlen(key)) * sizeof(char));

  strcpy(paramKey,path);          //build the JSON object name string
   strcat(paramKey,key);

  switch (type)
  {
      case json_type_boolean:  sprintf( value_buffer ,"%s",json_object_get_boolean(jobj)? "true": "false");
                               sdataParameterList[indiceValueStruct] = DM_ENG_newSystemParameterValueStruct(paramKey, value_buffer, NULL);
                               break;
      case json_type_double:   sprintf( value_buffer ,"%lf",json_object_get_double(jobj));
                               sdataParameterList[indiceValueStruct] = DM_ENG_newSystemParameterValueStruct(paramKey, value_buffer, NULL);
                               break;
      case  json_type_int :    sprintf( value_buffer ,"%d",json_object_get_int(jobj));
                               sdataParameterList[indiceValueStruct] = DM_ENG_newSystemParameterValueStruct(paramKey, value_buffer, NULL);
                               break;
      case  json_type_string : value_buffer  = strdup(json_object_get_string(jobj));
                               sdataParameterList[indiceValueStruct] = DM_ENG_newSystemParameterValueStruct(paramKey, value_buffer, NULL);
                               break;
  }
   free( value_buffer );
```

```
    indiceValueStruct++;
}
```

On the *_printJsonValue* function, each element in the buffe table is a TR-069 *System Parameter Value Struct* using the TR-069 Client API default function *DM_ENG_newSystemParameterValu*

Bibliography

[1] Westell Jeff Bernstein, 2Wire Tim Spets. Technical report 069 cpe wan management protocol. Technical Report 2, DSL Forum, 5 2004.

[2] James Gerhart. *Home Automation and Wiring.* McGraw-Hill Professional, 3 1999. ISBN 0070246742.

[3] Richard Harper. *Inside the Smart Home.* Springer, 8 2003. ISBN 1852336889.

[4] Marie Bénilde. Portrait de france télécom en multinationale. *Manière de voir*, 109 (2):86–86, 2010.

[5] Jean-Jérôme Bertolus, Jean-Michel Cedro, and Thierry Del Jesus. *Qui a ruiné France Télécom?* Hachette littératures, 2003.

[6] Bruno Jaffré. En afrique, construire les alternatives aux privatisations. *Benamrane Djilali, Jaffre Bruno et Verschave François-Xavier (sous la dir. de.), Les télécommunications entre bien public et marchandise, Paris: Charles-Léopold Mayer*, 2005.

[7] Martin Stiemerling. Nat and firewall traversal issues of host identity protocol hip communication. 2008.

[8] Jonathan Rosenberg, Rohan Mahy, Christian Huitema, and Joel Weinberger. Stun-simple traversal of udp through network address translators. 2003.

[9] Mohammad Alqahtani. Ipv4 address exhaustion.

[10] Zhou Hu. Nat traversal techniques and peer-to-peer applications. In *HUT T-110.551 Seminar on Internetworking*, pages 04–26. Citeseer, 2005.

[11] Michael Jeronimo and Jack Weast. *UPnP design by example: a software developer's guide to universal plug and play.* Intel Press, 2003.

[12] D Wing, S Cheshire, M Boucadair, and R Penno. P. selkirk," port control protocol (pcp). Technical report, RFC 6887, April, 2013.

[13] Lisa Sherwin. Upnp specifications named international standard for device interoperability for ip-based network devices, 2009.

[14] Pyda Srisuresh and Matt Holdrege. Ip network address translator (nat) terminology and considerations. 1999.

[15] J Rosenberg, R Mahy, P Matthews, and D Wing. Rfc 5389: Session traversal utilities for nat (stun). *Internet Engineering Task Force*, 2008.

[16] Florian Fainelli. The openwrt embedded development framework. In *Proceedings of the Free and Open Source Software Developers European Meeting*, 2008.

[17] Mike Petullo. Building custom firmware with openwrt. *Linux Journal*, 2010(196): 3, 2010.

Abstract

The TR-069 developed by Orange Labs allows to manage communication between customer-premises equipment (CPE) and Auto Configuration Server (ACS).It offers the use of a set of services administration, monitoring and diagnosis while avoiding the problems associated with hardware and software diversity.

Orange Homelive is the solution to manage the home on the mobile. From a single application, Homelive allows to interact with connected compatiable objects in the house. It proposes the Smart Home solution with a set-top box–Homelive.

Z-Wave is a wireless protocol designed for home automation (lighting, private heating) and so-called Smart Home. Z-Wave was bought by the company Americane Sigma Designs in 2008. It is optimized for low bandwidth exchanges and devices on battery. It can be easily integrated in consumer electronics, including remote controls, smoke detectors and safety sensors.

During the six-month internship, my main mission was to adapt the TR-069 Client at the Homelive box by resolving the incompatibility and implementing the RPC methods. Also, integrates the new Z-Wave data model in the TR-069 Client by contacting with the Luup firmware of the Homelive Box. With my internship result, Orange will be allowed to manage Homelive from distance to reduce the expenses on support services.

Résumé

Le TR-069 développé par Orange Labs permet de gérer la communication entre un équipement terminal du réseau local du client et un servert d'autoconfiguration associé dans un même réseau apprtenant à l'opérateur.Il offre à l'utilisation d'un ensemble de services d'administration, de contrôle et de diagnostic tout en évitant les problèmes liés aux diversités matérielles et logicielles.

Orange Homelive est la solution pour piloter sa maison depuis son mobile. A partir d'une seule application, Homelive permet d'interagir avec les objets connectés compatiables dans la maison. Il donne la solution de Smart Home avec un set-top box–Homelive.

Z-Wave est un protocole radio conçu pour la domotique(éclairage, chauffrage) et ce qu'on appelle l'Habitat comminicant. En 2008 Z-Wave a été rachetée par la société américane Sigma Designs. Il est optimisé pour des échanges à faible bande passante et des appareils sur pile ou alimentés électroniquement. Il peut être facilement intégrée dans les produits électroniques de consommation, y compris télécommandes, les détecteurs de fumée et capteurs de sécurité.

Pendant ces six mois de stage, ma mission principale a été d'adapter le TR-069 Client au Homelive box, en résoulvant les incompatibilité et implémentant les RPC méthodes. Ainsi que intégre le nouveau data modèle Z-Wave dans le TR-069 Client en communiquant avec le firmware Luup dans Homelive Box. Avec mon résultat de stage, Orange va être autorisé à gérer Homelive à distance pour réduire les dépenses sur les services de soutien.