Open Source Firmware Customization Problem: Experimentation and Challenges

Thomas Djotio Ndie[1(✉)] and Karl Jonas[2(✉)]

[1] National Advanced School of Engineering, CETIC, University of Yaounde 1, PoBox 8390, Yaounde, Cameroon
tdjotio@gmail.com
[2] Bonn-Rhein-Sieg University of Applied Sciences, Grantham-Allee 20, 53757 Sankt Augustin, Germany
karl.jonas@h-brs.de

Abstract. The great advantages of open source code-based projects like the case of open source firmware (OSF) are flexibility and freedom of customization. However, the difficulty inherent to their development process, which can be seen as a software composition issue, is the lack of structured approach and a teachable methodology for efficiently tackling such project. In this paper, we propose a 5-step pedagogical OSF's customization approach, coupled with an agile development process to guide the learner and ease his comprehension. We experience this approach to prototype WiAFirm, an OpenWRT-based firmware for operating IEEE 802.11x enabled WAP. Two groups of 04 students each were involved. After 2 months of experimentation, the group that applied the approach was able to integrate into the core OpenWRT a custom WifiDog captive portal feature as a built-in package; while the other group had barely understood the goal of customizing an OSF.

Keywords: WiAFirm · Open source firmware · Customization

1 Introduction

The industry of open hardware has given to scientists and researchers, the possibilities to design and develop open source firmware (OSF). It provides to end-users the freedom of customization, to engineers the freedom of improvement, and to developers the opportunity to contribute in the field by experimenting and optimizing existing (or creating new) features. An open source code (OSC) allows independent developers to create derived versions from the original one. Despite this flexibility, their development process is opaque, complex, lack structured documentation and a teachable methodology that can efficiently drive their conception. These shortcomings discourage newbies (e.g. students) to contribute in the field. We are particularly interested in the case of OSF customization issue, which can be assimilated to the software components integration problem. Which rules govern an OSC-based development project in a sustainable way? How can a learner be pedagogically introduced to gradually manage such a project?

K. Arai et al. (Eds.): CompCom 2019, AISC 998, pp. 306–317, 2019.
https://doi.org/10.1007/978-3-030-22868-2_23

This paper aims at proposing a 5-step pedagogical approach inspired from [1], couple to an agile development process [2] to guide the learner in the customization process and to ease his comprehension. The main benefit of it is the better quality control of the OSF development process and the knowledge sharing. We applied it for prototyping WiAFirm, an OpenWRT-based firmware [3] for operating the WiABox appliance [4]. The experience consisted to integrate to an optimized OpenWRT base firmware [5] the WifiDog-based captive portal feature [6] for controlling users requesting wireless access to our campus network. Two groups of 04 students each were involved and, after 02 months, the group that applied the method was able to implement the project while the other group was almost to understand the aim of OSF customization.

The Sect. 2 of this paper presents the problem of OSF customization. The Sect. 3 describes the pedagogical approach used for prototyping WiAFirm (Sect. 4). The Sect. 5 discusses obtained results and briefly raises up some challenges.

2 The Problem of Open Source Code/Firmware Customization

If the problem of embedded system firmware development is formally addressed [2, 7, 8], it is not the case of OSC-based firmware also known as open source firmware. This issue can be formulated as a problem of software composition from third parties [9]. Composing the new firmware can then be simply like applying a puzzle's game strategy. The puzzle's pieces are assimilated to firmware packages available at their respective repositories. The new OSF results from a plan of the player, driven by the architectural concept he wants to build. The development and testing consist of arranging a game. Errors management consist of fixing or rebuilding a specific part of a worked form. The problem is that the puzzle game is intuition-oriented and has no formal rule. To the best of our knowledge, you build the figure you want given the set of pieces. If you want to build complex architectures, then you have to design new specific pieces of puzzle or remodel existing ones. It may require additional tools to glue together old, rebuilt and new parts. That is why learning from a customization experience is a complex issue and experience-oriented.

We illustrate this issue with the idea of creating an OpenWRT-based OSF [10–14]. This can result to play with more than 3000 packages from different repositories and can require more than a real strategy and experience. Some third parties open source packages can be outdated. In addition, some low-level modifications from code inspection can be required. E.g. the bridging and VLANs, intelligent and transparent layers 2 VPN configurations in the case of Freifunk [15, 16]. This implies writing some custom codes for achieving and automating desired functionalities. According to the project's objectives, new features may need to be developed and integrated in a "one-downloadable" and distributed image file. The need to have domain engineering knowledge, to know the system and development environment, the package integration, code inspection, compilation and debugging processes and, testing policies are required.

The customization development approach helps to get what a designer want, and the component composition approach helps to have it fast. This is only feasible if you

are an experienced developer. The drawbacks of customization are dirty hacks, with difficulties for unexperienced developers to easily jump into the development process and hands-on the source code. There are difficulties to maintain and to add new features. Indeed, a package can be subject to bugs and security issues. If the OSF's development process was methodologically addressed, it could have reduced cost, time and, enhanced quality control. The documentation is also a great issue, even when it exists, it is unstructured, uncompleted and it is hard to see technical/scientific one.

Some activities related to the complex software customization process that we have identified are (but not limited to): (1) clone the base source code from its official repository (e.g. github); (2) use the image creation tool to select components (modules and packages) [9]; (3) create the system programming environment (sometimes multiple languages are involved [17, 18]); (4) perform raw compilation, debug, fix errors and integrate patches; develop third-party packages and glue code; (5) integrate and test; (6) decompress the source code, hack the system; (7) recover it from the break, just to name a few. For addressing an OSF project, what is the good starting point for better quality control? In the next section, we propose a pedagogical approach for designing and modeling an OSF-based project.

3 Presentation of the Pedagogical Approach

The process of forming a new OSF lacks an effective method. Most of the time, the project leaders are experts in their respective fields and almost know what they need and want and can directly start by building a prototype. We propose a pedagogical approach (illustrated in Fig. 1) in 5 phases: (1) the initialization phase (2) the conception phase, (3) the realization phase, (4) the termination phase and, (5) the maintenance phase.

Phase 1: The initialization. It comprises two sub-phases: the **system requirement** and the **domain analysis**. The objective of the **requirements analysis** is to draw the system specification and to capture the end user's needs and wants. Here is answered the question "What is the purpose of the system?" in an unambiguous and testable way [1]. The **domain analysis** helps to capture the client's knowledge framework: developing concepts, languages, terminology just to name a few. At the end of this phase, an analysis and planning document is produced.

Phase 2: The conception or design. This phase consists to determine how the OSF will provide all the needs and features. It comprises research and leads to model the system's architecture. The designer has to detect critical parts of the system e.g. those involved in real-time functionality. Here are identified eligible packages from in-depth study of the base OSF framework for customization. Their respective source code repositories are located. The need for domain engineering knowledge is necessary.

Phase 3. The realization. It consists of the effective customization of the eligible existing source codes and to write new codes, modules/packages if necessary, according to the system design requirements and specifications. Customized and new packages are glued together in the new framework.

Phase 4: The termination. It consists of the deployment and the integration of the developed and compiled packages in form of binary files (e.g., *.ipk* and *.bin* in our case). Then various tests (unit and global tests), performance evaluation and validation are made. The system testing, system deployment and, the system evaluation are important steps to validate and insure the quality control of the final build.

Phase 5: The Maintenance. It consists to collect users experience and change management for future improvements and, some can take us back to reviewing design, analysis phases. The documentation is an important artefact during the project lifecycle.

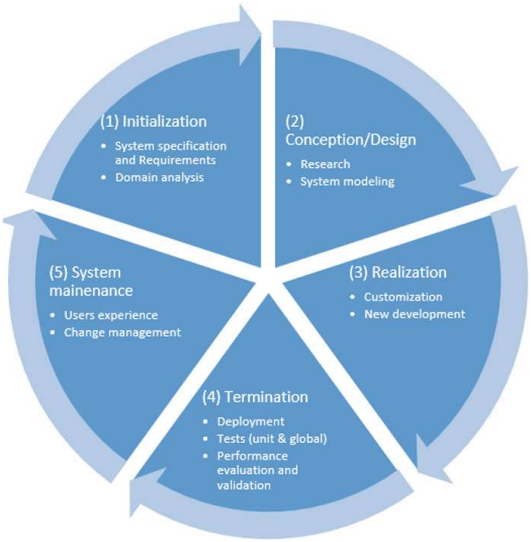

Fig. 1. Proposed model for an OSF-based project development process

In this section, we realize that it is definitely not an easy task to start the development of a new firmware project from existing OSFs daily maintained by free, enthusiastic, passionate and, motivated communities like the case of OpenWRT or Freifunk-Gluon. This is particularly a difficult problem if the project aims to have professional, technological and scientific impact like the case of WiAFirm.

4 On Modeling WiAFirm an OpenWRT-Based Firmware for WiABox Appliance Framework

WiAFirm is a modular OpenWRT-based framework mostly inspired by the Freifunk-Gluon firmware [15] and OpenWISP [18] for operating multi-band IEEE 802.11x wireless access routers. It is part of the WiABox project initiative [4].

4.1 The Initiation Phase: System Requirements and Specification of WiAFirm

WiAFirm will be the base firmware for building different wireless community networks (WCN) with varied expectations, with special focus to hard-to-access areas. It MUST

[19] be easy to setup, customizable, pre-configurable and, compatible with other OpenWRT-like firmware and hardware. It MUST provide a web GUI to non-skill users for configuring and monitoring network settings. For supporting the wireless network access and control (WNAC), WiAFirm MAY implement a customized hotspot and captive portal solution for end-user's authentication through an Authentication, Authorization, Accounting (AAA) backend server. WiAFirm MAY be used to deploy private WLANs. Therefore, the use of different credential types including digital certificates, user-names and multifactor passwords will be possible. It inherits Freifunk-Gluon essential requirements: security, configuration and much more. The anonymity of a **user** MUST be ensured in public places and in-line with local regulations. It MUST guaranty an end-to-end protection of end-user data. It SHALL implement auto-configuration and self-healing features to deliver secure, scalable and reliable service to end-users. It MUST provide (1) geolocation to ease remote and graphical monitoring, (2) mesh VPN on LAN and WAN to OPTIONALY allow someone to share part of his Internet bandwidth connectivity with the community.

In this paper, we choose to implement the WNAC requirement only by building a customized captive portal feature based on WifiDog [6].

4.2 Phase 2: Conception and Design of WiAFirm

We start with an in-depth study of the base OpenWRT OSF and the structure of its package. Table 1 below gives a snapshot of the list of eligible packages needed in regard to the experimentation.

Table 1. List of features for package selection.

Features	List of packages	Source repositories	Notes
Kernel modules	OpenWRT base modules	OpenWRT	IEEE 802.11x Fundamentals
Target system or OHW architecture	Ar71xx-generic	OpenWRT	
EAP-AAA backend server	FreeRadius v3*	FreeRADIUS	802.1x EAP
WNAC: Hotspot & Captive portal	WifiDog	WifiDog	UAM
VPN Mesh protocols	VPN on LAN, on WAN (fastd, tinc, alfred)	Freifunk-Gluon	O-SP*
Enhanced L2&3 routing and mesh protocols	Bird, Batman, Batman-adv (meshclient), babel, bmx6 (meshbackbone)	Freifunk-Gluon	O-SP*
IPV4/IPV6 addressing	radvd	Freifunk-Gluon	O-SP*
Mapping, GIS monitoring services	OWGIS,	Freifunk-Gluon	

In this table, the content commented "O-SP*" mentions the features that are out of the scope of this paper. As above stated, we will limit our experimentation on only the implementation of one feature: the captive portal.

The System Architecture. The design considerations and component's identification leads to the conception of the framework architecture illustrated in Fig. 2. The users of the WiAFirm framework, in addition to benefiting to WiABox wireless community services [4], will always have options to use wireless services natively provided by OpenWRT. The concept of plugins inherited from Freifunk-Gluon is used to extend the framework by customizing some existing features and developing new community-based services.

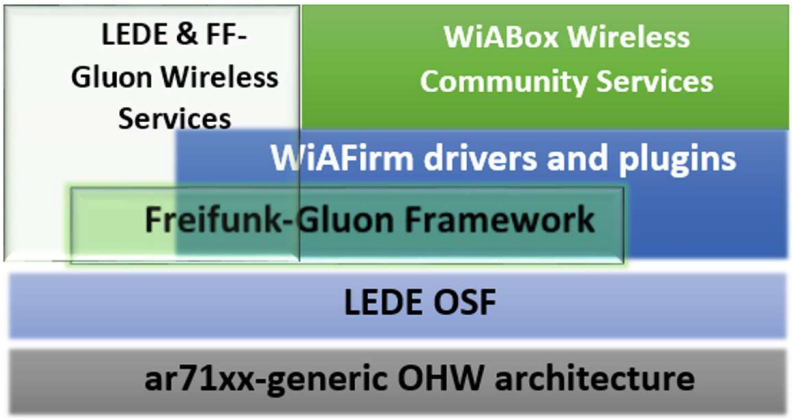

Fig. 2. The WiABox Appliance framework's architecture.

Conceptual architecture of the Captive Portal: WiAGate. Its architecture is illustrated in Fig. 3 and consists of two logical parts respectively running on a Web server and on the wireless router: The WiAGate_Server and the WiAGate_CP. Its complete presentation is out of the scope of this paper. It is the full hotspot and captive portal solution for WiABox project.

WiAGate_CP is a WiFiDog-based package to provide the captive portal service. It communicates with WiAGate server which integrates some specific system processes among which are:

- **WiAGate_Auth.** It intercepts all requests from WiAGate_CP and performs authentication check, sends the response back to it. Based on the response value, the WAP can be authorized or not to connect the end user to the network.
- **WiAGate_Acc.** It provides the data access history and accounting for each hotspot, WISP and end-user for pricing, billing and payment purposes [20].
- **WiAGate_Mon.** It gives the possibility to specify some restrictions such as duration/time limit of the connection, the volume or the maximum usable bandwidth.

4.3 Phase 3: Prototyping of WiAFirm, Design and Integration of WiAGate_CP

Customizing the Candidate Package. The eligible package for customization is WifiDog V1.3.0 [6]. The resulted customized package to be integrated as an OpenWRT build-in feature is WiAGate_CP. The required development resources are: the C programming language (gcc on Linux Kernel), the source code of the WifiDog and the source code of the stock LEDE release 17.02.3.

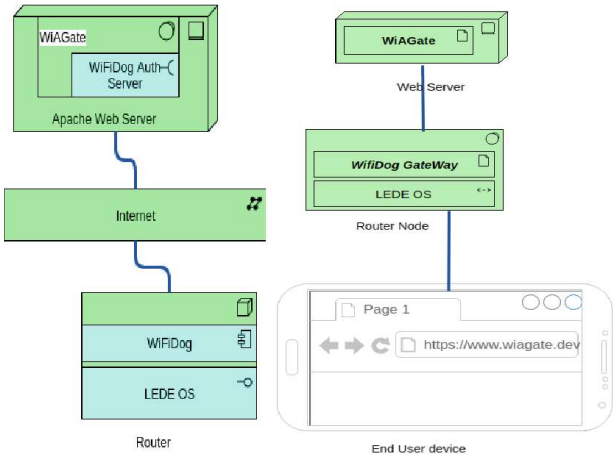

Fig. 3. Interaction between: (a) the WAP (router) and (b) End user device and the WiAGate Server through Internet

The steps to follow for creating and integrating a package as a OpenWRT build-in package are inspired from the "helloword" example tutorial described in the developer manual available here [21]. They are summarized in 12 steps below:

(1) Preparing, configuring and building the necessary tools.
(2) Adjusting the path variable for project to be created.
(3) Creating the source directory and files for hosting the base source code.
(4) Creating a package as a standalone "c" program. Here we cloned the source code of WifiDog into a local directory e.g. the 'wiagatecp' folder in this case. We inspected the source code and brought all the necessary changes and patches according to our expectations.
(5) Compiling and testing the application.
(6) Creating the .ipk package from the application.
(7) Creating the package Manifest file "Makefile" for the resulted package.
(8) Including the package into the OpenWRT system by updating the feeds.conf. default or feeds.conf file.
(9) Update and install feeds from the/home/buildbot/source directory. A web feed is a query-based system designed to bring the user relevant, immediate, useful and regularly updated stream of data or information. It aims to make a collection of web resources accessible in one spot.
(10) Testing WiAGate_CP as a standalone application.

(11) Publishing the WiAGate_CP in a new repository.
(12) Customizing the OpenWRT stock firmware with the generated WiAGate_CP package. We used an extended and optimized stock base LEDE build for TP-Link Archer C7V2 [22]. It includes some preconfigured packages and kernel modules that feature some of our performance needs and wants.

Compiling the Stock LEDE with WiAGate_CP as a Build-in Package. All the 7 steps described hereafter are done on Linux Ubuntu 16.04 LTS. We followed the steps advised on the OpenWRT's official website [13, 21] to process this task. It comprises:

Step 1: Installing the dependencies.
Step 2: Updating the sources and getting the stock LEDE source code with Git.
Step 3: Update and install the source from feeds.
Step 4: Getting the WiAGate_CP package from our repository and unzipped it into a local folder.
Step 5: Configuring the build image by using the builtroot tool build [21].
Step 6: Compiling the build (building WiAFirm image). This can take time, depending to the performance of the host computer. It can lead to fixing errors if any until having an error-free process.
Step 7: Resetting the device with WiAFirm, the resulted customized build. The target wireless access router is TP-Link Archer C7V2. After a successful reset process, the WiAGate_CP package can be verified in the running build image through the CLI or luci interface. WiAFirm is ready for deployment and testing.

4.4 Termination Phase: Prototype Deployment and Testing

The test environment consisted of a 64 bits Linux OS Ubuntu 16.04 LTS. We used three (03) TP-Link AC1750 C7V2 wireless dual band Gigabit router. The additional tools are programing languages and frameworks: PhP, NodeJs, VueJs and Laravel, FreeRADIUS v3, and some Debian software systems utilities. The WiAGate_Server is up and running on the Linux host. It is the backend server also performing AAA capabilities. Its presentation is out of scoop of this paper.

The test is possible from any browser on any Wi-Fi enabled end-device. Any user will be warmed that an authentication is needed prior getting access to network. A login or a registration is required.

4.5 Maintenance Phase

The system and package testing, system deployment, system maintenance and system evaluation are important steps to assure the quality of the final build. Here we collect users' experience, results and measurements, analyze them and review all the process if necessary. This can take us back to review the complete customization process. The documentation is an important artefact during the project lifecycle, e.g. the design document of the project.

5 Related Works and Business Idea

Almost all the related OSFs OpenWRT, Freifunk-Gluon [15, 16], OpenWISP [17, 18, 23], Chillifire [24], DDWRT [25] just to name a few, lack structured and scientific documentation and consequently raise shortcomings of learning from their experiences. Some are partially closed source like the case of Chillifire and DDWRT. The above-related projects are almost widely in use in their local countries and they each have active developer communities, so it will be a bad idea to think starting from scratch. OpenWRT/LEDE currently manages more than 3000 software packages. When considering each fork, more additional specific packages need to be taken into account. Exploiting such huge list of components is a great challenge. By applying the "pick-on-desire" policy, one just has to select desired packages, configure them and in doing so, build his own architecture. When missing, he can write his own package and integrate it based on specific rule. This approach suits to general-purpose projects with a low degree of customization on low-end devices for building each own solution by means of few configurations. Nevertheless, with no experience, no methodology and required skills, this really leads to great confusion.

The business idea is to build free Wi-Fi and share it with family and friends, in café, in the community and expand it town-wide, then rural-wide then, collect users' experience, surveying and provide valued-added services on top of the built WCNs. Each WCN is autonomous in terms of structuration and organize trainings and professional events for promoting the community-oriented services [26]. End users are (but not limited to), SMBs, researchers, academics, network administrators and developers, commercial, education and health centers, city councils, remote communities living in hard-to-access areas, hotels, camping, cafés, bars, residences, remote schools and colleges. People living in the neighborhood of the above-mentioned places can automatically benefit from the hotspot services providing contributions from other users with their internet connection, bandwidth sharing and their packet forwarding ability. Competitors are private local WISP and cybercafés able to provide Wi-Fi community-wide.

6 Discussion and Challenges

It may happen that a package has several feeds. It is recommended to focus on the most active one. The criteria used to select a package are stability, functionality, security, strong community, documentation even if incomplete, development language, openness, maintenance and communication. The anatomy of the base OSF code shows that some source packages are natively included in its official repository. Example is the case of OpenWRT and its forks OpenWISP [17, 18, 23] and Freifunk-Gluon [15, 16]. It is not recommended to select them as such to expect good results because they have a separate trunk and are not updated with the same frequency. That is why it is recommended to directly clone the official repository of the candidate source packages, customize them according to our needs and wants, create bin or ipk files and piece them all together with the OSF base kernel. For new developed packages emerged from the

system analysis, we exactly do the same after successful compilation and tests. For integrating all the identified packages, the process is summarized as follows:

```
For each package do:
  search for the right version,
  update the /etc/apt/sources.list file,
  import right keys for validating the downloaded sources,
  update and upgrade the system (in Linux Ubuntu apt-get update
& apt-get upgrade),
  install the dependencies,
  clone the source code,
  configure it in regards of expected results,
  create the build accordingly with the required command (make,
make menuconfig, make install, make image),
  do post-configuration actions: fix errors if any
until process is error-free.
```

The aim of customization is to create working code as quickly and as accurately as possible. However, this process involves code inspection, testing and debugging which are activities most of the time reserved to certain elite, that is predominantly professional.

Most of the features to be implemented in WiAFirm are inspired from Freifunk-Gluon, OpenWISP and Chillifire. These solutions are optimized for urban areas and for developed countries. Some are partially closed source and efforts to use them as such is difficult. They were designed with the assumption that the Internet connectivity will be always available, but in our context it is not the case and, building a business model with low-income people is not an easy task.

As an important element in the definition and design of WCNs, an OSF in its current architecture, despite its flexibility of multi-context adaptation becomes inappropriate in an ubiquitous service environment [5, 27]. This is why we are considering using the SOA paradigm to propose a complete service-oriented architecture as a response to this problem. A package would therefore be seen as a service (**PckaaS**) in the SOA perspective and would instead offer business functionality to other applications or packages hosted in a service repository. Considering WiAFirm as a FaaS (firmware as a service) will allow, instead of integrating the functionality of the application itself at the WAP level, the possibility to implement instead a communication interface with the packet registry in order to use the package it needs on demand.

7 Conclusion and Future Works

In this paper, we have addressed the issue of modeling a new firmware from existing, proven most used and, tuned OSFs like OpenWRT/LEDE and Freifunk-Gluon. Because it lacks a formal methodology to tackle such project, we proposed a 5-phase pedagogical approach as a pragmatic development process model that can be applicable to any OSC-

based project. The approach consists of the: (1) initialization, (2) conception, (3) realization, (4) termination and, (5) maintenance phases. We illustrated it with in-depth steps on how to customize a WifiDog-based captive portal (WiAGate_CP) and, integrate it into an optimized stock LEDE, as a build-in feature for controlling access to end-users in the WCN. The resulted build is the prototype of WiAFirm, inspired from Freifunk-Gluon for operating WCNs in rural DCs.

The main benefit of the proposed approach is the better quality control of the OSF development process and the knowledge sharing. The learner is methodologically introduced and guided step-by-step in the OSF engineering paradigm.

Regarding the difficulties encountered and to benefit from the great potential of WCNs in a ubiquitous service environment, we consider in the future using the SOA paradigm to model WiAFirm as FaaS (firmware as a service). Thus, instead of integrating the application functionality itself at the WAP level, it will instead implement a communication interface with the package registry to use the package it needs on demand.

Acknowledgments. Special gratitude respectively goes to Alexander von Humboldt Foundation; GIZ-CIM Organisation; the University of Applied Sciences Bonn-Rhein-Sieg and the CETIC Project of the University of Yaounde 1 for their various supports.

References

1. Barr, M.: Firmware architecture in five steps (2009). http://www.embedded.com/design/prototyping-and-development/4008800/1/Firmware-architecture-in-five-easy-steps. Accessed Nov 2019
2. Greene, B.: Agile methods applied to embedded firmware development. In: AGILE Development Conference 2004, pp. 71–77. IEEE, Salt Lake City, Utah, USA (2004)
3. LEDE homepage. https://lede-project.org/. Accessed June 2017
4. Djotio, N.T., Jonas, K.: WiABox 2507: Project Initiative for an Open Wireless Access Technology alternative for Broadband Internet access in rural areas. In: WACREN Workshop, TANDEM Project Dakar Senegal (2016)
5. Brown, E.: OpenWRT adds IPv6, preps for IoT future (2014). http://linuxgizmos.com/openwrt-adds-ipv6-and-prepares-for-iot-future/. Accessed June 2018
6. WifiDog Captive Portal Suite project homepage. http://dev.wifidog.org/wiki/doc. Accessed may 2018
7. IPA/SEC.: Embedded System Development Process Reference guide. Version.2. Software Engineering Center, Technology Headquarters, Information-Technology Promotion Agency, Japan (2012)
8. Geiger, L., Siedhof, J., Z¨undorf, A.: µFUP: a software development process for embedded systems. In: Proceedings Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme I, Schloss Dagstuhl (2005)
9. Capretz, L.F.: Y: a new component-based software life cycle model. J. Comput. Sci. **1**, 76–82 (2005)
10. NAGY, P.: Configuration of OpenWRT System Using NETCONF Protocol. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology (2016)
11. OpenWrt history. https://wiki.openwrt.org/about/history. Accessed Aug 2018

12. Fainelli, F.: OpenWrt/LEDE: when two become one. LEDE (2016). https://events.linuxfoundation.org/sites/events/files/slides/ELC_OpenWrt_LEDE.pdf. Accessed Oct 2017
13. OpenWRT homepage. https://www.openwrt.org/. Accessed Jan 2019
14. Capoano, F.: Do you really need to fork OpenWRT? Sharing our experience with OpenWISP Firmware (2015). https://prplworks.files.wordpress.com/2015/10/do-you-really-need-to-fork-openwrt.pdf. Accessed Oct 2017
15. Freifunk Gluon Documentation. https://media.readthedocs.org/pdf/gluon/latest/gluon.pdf. Accessed Aug 2018
16. Freifunk gateway. https://wiki.freifunk-franken.de/w/Freifunk-Gateway_aufsetzen. Accessed June 2018
17. Ferraresi, A., Goretti, M., Guerri, D., Latini, M.: CASPUR Wi-Fi Open Source. In: GARR Conference 2011 (2011)
18. Openwisp home page. http://openwisp.org/. Accessed Aug 2018
19. RFC 2119: Key words for use in RFCs to Indicate Requirement Levels (1997). https://tools.ietf.org/pdf/rfc2119.pdf
20. Djotio, N.T.: An entity-based black-box specification approach for modeling wireless community network services. In: Proceedings of SAI Computing Conference 2019 (2019)
21. OpenWRT Developer guide. https://openwrt.org/docs/guide-developer/start. Accessed Dec 2018
22. Optimized LEDE. https://forum.lede-project.org/t/gcc-6-3-build-optimized-tp-link-archer-c7-v2-ac1750-lede-firmware/1382. Accessed Dec 2018
23. Gijeong, K., Sungwon L.: Openwincon: open source wireless-wired network controller software defined infrastructure (SDI) approach for fixed-mobile-converged enterprise networks. In: ICN 2016: The Fifteenth International Conference on Networks (includes SOFTNETWORKING 2016), pp. 58–59 (2016)
24. Support.chillifire.net: Chillifire Hotspot Router Installation Guides. http://www.chillifire.net/. Accessed July 2018
25. DDWRT homepage. https://dd-wrt.com/. Accessed Jan 2019
26. Freifunk homepage. https://freifunk.net/. Accessed Jan 2019
27. Open vSwitch. http://openvswitch.org/. Accessed June 2018

An Entity-Based Black-Box Specification Approach for Modeling Wireless Community Network Services

Thomas Djotio Ndié(✉)

National Advanced School of Engineering, CETIC,
University of Yaounde 1, PoBox 8390, Yaounde, Cameroon
tdjotio@cetic.cm

Abstract. Wireless community networks are emerging as a better alternative to bridge the digital divide in underserved areas. As such, they can stimulate a proximity economy and allow the emergence of local digital service operators like wireless service providers (WISPs). To achieve this goal, we propose in this paper, an entity-based black-box specification approach to formally describe: (1) the concept of wireless community network service (WCNS), (2) the mechanisms of interaction with the latter through the authentication-authorization-accounting (AAA) model, and (3) the profitability mechanisms that complement the AAA model which are: pricing, billing and payment (PBP). Our case study is a campus-type WCN. We have defined an abstract representation of the network service concept through characteristics common to all services. The access control and profitability functions are described through well-defined and justified formalisms. These functions are also well illustrated by practical cases.

Keywords: Network service · Service access · AAA · PBP · WCN

1 Introduction

The goal of building wireless community networks (WCNs) [1, 2] is usually for social and not-for-profit purpose. Built based on grassroots approach [3] and depending on the project's initiators, they offer free and unrestricted access to network services to users in the need. Economic profitability is therefore not an initial concern for the designers of such projects [4]. Yet, they can be an excellent alternative for the development of the digital proximity economy in developing countries (DC). For this to be possible and especially in a context of ubiquitous network service, it is important to challenge some well-established models such as access control models for interaction with network services, in order to effectively integrate the profitability dimension into community wireless projects adapted to underserved areas.

Many works present the AAA model (authentication, authorization and accounting) [5] as a solution to this concern, except that it is more suitable for business networks that require more security through well-known standard protocols like RADIUS [6] or Diameter [7]. The AAA model was designed without taking into account financial

K. Arai et al. (Eds.): CompCom 2019, AISC 998, pp. 318–335, 2019.
https://doi.org/10.1007/978-3-030-22868-2_24

appreciation functions, functions that could be very useful in a service-oriented WCN context. To this shortcoming we propose the PBP model for pricing, billing and payment. We obtained a more complete model called 3A-PBP, composed of 06 sub-models or components, each of which fully describes input and output spaces for the full model.

Our problem is thus reduced to that of the description of the input and output spaces of a system, as well as the functions linking the outputs to inputs. The realization of the functions allows to operate the system. Our solution is based on the entity-based black box specification method, which is more appropriate, for intuitively modeling WCN's services, by using mathematical formalisms to describe the properties and the behavior of our system. This would solve the problem of the variety of interpretations and offer flexibility and neutrality in terms of the implementation environment of the 3A-PBP model. Entities are the elements of the A-A-A-P-B-P protocol; the outputs produced by a component are the inputs for the next component. This method requires 04 steps: (1) The definition of the entry and the output spaces; (2) The definition of the system entities; (3) The definition of constraints and; (4) The definition of the relationship between inputs and outputs. Figure 1 illustrates the architecture of the whole system and highlights the actors and its different components.

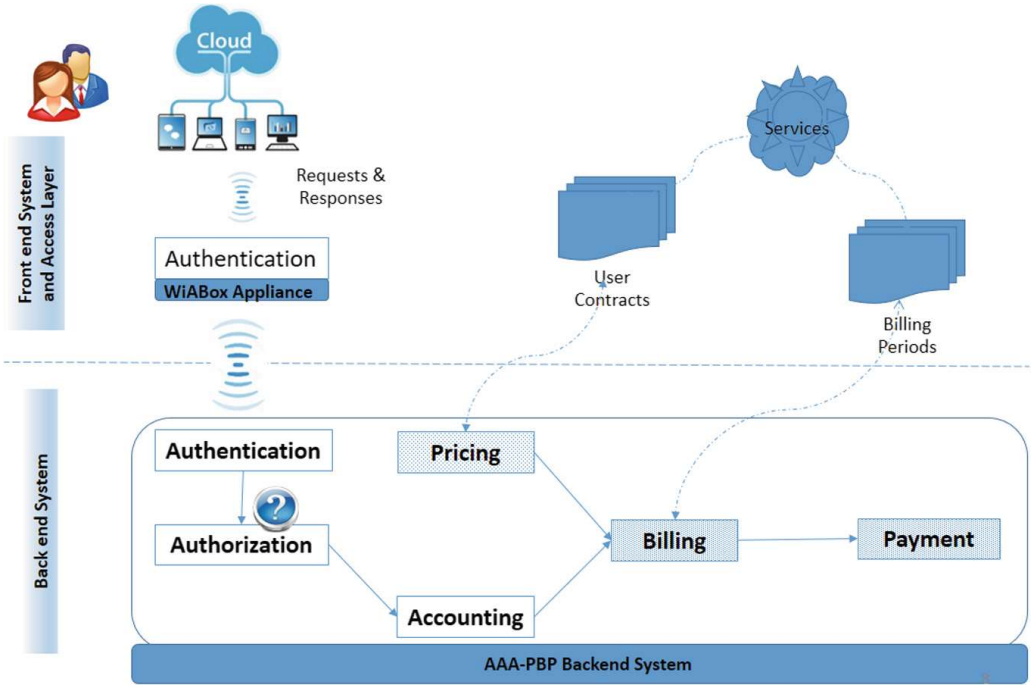

Fig. 1. The 3A-PBP architecture model.

The Sect. 2 defines and characterizes the wireless community network service concept. The Sect. 3 describes WiABox.Net, our virtual WISP for the simulated practical cases and set the assumptions as well. The Sect. 4 models the 3A-PBP ecosystem through its 06 entities using well-defined functions and properties. Before concluding, the Sect. 5 raises up some discussions and challenges.

2 The Concept of Service: Definition and Characterization

2.1 Definition of the Service Concept

Just like a product, a service is something that can be sold, but is intangible. Talking about sales involves 03 actors: a producer, a supplier and a consumer. In the context of computer networks, a producer creates and administers digital services through an administration system. The provider subscribes to the services and makes them available to consumers through a platform. The consumer requests the services provided by the platform according to certain predefined constraints. In the concern of this paper, we will consider only two actors: the supplier and the consumer (also called user or customer).

2.2 Dimension of a Service

The concept of service has 04 intrinsic characteristics that distinguish it from the notion of product: intangibility, inseparability, perishability and heterogeneity.

Intangibility means that the service cannot be materialized, e.g., in the case of a digital library service, this is illustrated by the fact that access to the library is not materializable. Inseparability characterizes the fact that the production of the service and its consumption are simultaneous. For example, the bandwidth provided by a Wireless Internet Service Provider (WISP) is instantaneously consumed (totally or partially depending on the network throughput). Perishability means the service cannot be returned or stored. The example of the Internet bandwidth illustrates this property as well. Heterogeneity denotes the fact that the quality is not constant from one service to another; and the performance of a service, even with the same provider, varies with the user. For example, the actual Internet throughput will not be the same for 02 users who have subscribed to the same package.

2.3 Actors: Service Provider (SP) vs Service Consumer (SC)

We only consider two main actors: the SP and the SC, who both interact and trade at the business level. The SP indefinitely produces services and earns money. The SC indefinitely consumes services and spends money. In the context of a WCN for example, the profit made by a SP can be financially nil, but socially important. Several factors come into play. We are interested in evaluable mechanisms or methods for controlling and accessing provided network services. The SPs must provide authenticatable, traceable, accountable, priceable, billable and payable services. The system must enable relevant analysis, must be error-free so that SCs can usefully spend their money for the requested services. The access to services is constrained by the availability of resources whose allocation must comply with certain formal rules. SPs must allocate resources efficiently and to do so, they will need to study and analyze the trend in the use of available resources, to know for example, how many services they have or can be instantly provided.

The SC is the beneficiary of the service. He requests access to the service, this access is conditioned by the positive verification of a certain number of information, in

particular (1) the capacity of the latter to compensate the service, (2) the capacity of the system to provide the requested service and, (3) the availability of the service itself.

Three service payment alternatives are possible: (a) prepaid, (b) postpaid and, (c) on demand (on the go). SCs are constrained by the throughput or the "quantity" of the provided and available service. In order to be able to equate the service used with the cost to be paid by the SC, a SP must be able to quantify the service used (e.g. by the time of use) in order to charge the SC. Services are served to SCs in general according to several packages or flat rate formats. A package is simply a parameterized service. E.g. for the Internet access, these parameters can be the connection throughput and the exchangeable data volume. Therefore, we can have settings like package1 (2Mbps; 1Go), package2 (1.5Mbps; 2Go as example of packages. This not only allows the SC to have several choices to satisfy his needs, but also for the SPs to offer several pricing options.

2.4 The Concept of Time

The time is the element that helps to differentiate moments or instants [8]. In the collection of information, it is necessary. It allows the study of the trend of the use of the service. The time is much more a statistical tool; it offers the possibility to study the population of SCs and SPs and allows the counting of the "quantity" of the service used.

3 Base Assumptions and Presentation of WiABox.Net, Our Virtual WISP

3.1 Network Service Example and Presentation of WiABox.Net

The network service example that will support our practical illustrations is *Internet access*. This service will be provided according to well-defined constraints by WiABox.Net, our virtual WISP for simulation purpose. To simplify the model, we will assume that WiABox.Net does not operate in a competitive environment. All users of the Internet service (SP) will have the same settings: ID; Password; Name; FirstName; Email; Package; RegistrationDate and, Address.

3.2 Package Concept

WiABox.Net supports SCs based on multiple packages. A package is simply a combination of the bandwidth and the amount of data that are allocated to a user. WiABox.Net parameterizes 05 packages named: packageA, packageB, packageC, packageD and packageE. We will assume to simplify the model that WiABox.Net has only 01 long range wireless access point (WAP), and it can only grant access to up to 10 users simultaneously. The users are all distributed (a priori not uniformly) in the perimeter of the WAP.

3.3 Concept of Physical User and Logical User

A physical user refers to a SC or client. He can be identified by his email address and password. In contrast, a logical user is a user account registered in the database. It can

be identified by the combination of his ID, email address, password and/or package. A SC can have one or more user accounts. A package plan is associated with an account that uniquely identifies it. A SC can then subscribe to one or more different packages.

4 On Modeling the 3A-PBP Ecosystem

In this section we model the ecosystem of the 3A-PBP (heard triple A-PBP) through its six principle components (or entities) using well-defined functions and properties.

4.1 Authentication

The authentication is the mapping and comparison of a received identity to that of a stored one (equipment or user) in a system. We apply the OAD process, which consists first to *obtain* the credentials from a host, and then *analyze* them, and finally *determine* if they are associated with the requesting entity. Authenticating a user would then consist to compare and match his identity (e.g. his email address and password) to all the stored information that refers to him (his name, first name, email address, phone number, password, etc.). To authenticate a user of a service, the system therefore needs information that uniquely identifies him and a test function.

The set I of the inputs is the set of information of all [possible] types that can be persistently inserted into the system. A fortiori it contains, e.g. all the strings of characters that can be used to identify a user [9]. If the system needs an email address and a password as input data, then the system requires two strings. Hence, I can be simply seen as the set of strings and the set of input information is then an element of I^2.

Authentication Function. The *Auth* authentication function is the function that tests whether the information entered by a user and processed by a host, correspond to a user included in a certain group for identification purposes. We define the *Auth* function as following:

$$\begin{aligned} Auth : I^n \times \mathcal{P}(U) &\rightarrow \{0,1\} \\ (i,E) &\mapsto Auth(i,E) \end{aligned}$$

- i is an element of I^n, where n is chosen by the SP, it is the set of identifiers;
- E is a part of U, the set of all users of the system;
- When this search is successful, the function returns 1, or 0 otherwise.

The authentication function in most of the cases will be the search and comparison for $i \in I^n$ in a projection, according a subset of the set of users includes in the set E.

Practical Case. Considering WiABox.Net our WISP defined above, only the ID that is automatically generated by the system or the couple (email_address, password) will allow to uniquely identifying a SC. We will then use it for these purposes.

What happens if a SC has multiple packages and uses the same email address and password for all his user accounts?

As stated in the specifications of WiABox.Net, a user is fully defined by his email address, password and package. Thus, a SC with three different packages: f1, f2 and f3, has three user accounts. However, since the system just asks for the email address and password for identification process, this would mean that if a SC has more than one user account, the system will have to make a choice among its accounts. The system will choose his oldest user account.

Since these are strings that are processed, then I is the set of all possible strings. So the global information provided by the client is an element of I^2. In addition, only its users U of the service are concerned. As a WiABox.Net user is an 8-tuple of the type (ID, password, last_name, first_name, email_address, package, registration_date, address), then $U \subset I^8$. Finally, to authenticate a SC who sends i = (email_address, password) as a set of identifiers, the *Auth* function is explicitly defined by:

$$Auth(i, E) = 1_{E_{|val_5, val_2}}(i)$$

- $E_{|val_5, val_2}$ is the projection of E according to the items email_address and password.
- Remember that for a given set E, 1_E denotes the indicator function of E.

4.2 Authorization

In the AAA model, the principle of authorization consists to decide which resources will be allocated to an authenticated user on the basis of verifiable criteria. The answer to the request for using a resource by a user x is positive, if and only if the proposition '$p(x)$*: x is eligible for using the service*' is true.

Now Let us assume that WiABox.Net has only 05 available hosts to offer Internet access and that 07 of its subscribers wish at the same time to consume the service according to their packages. For each customer x, the proposition $p(x)$ is true, but the constraint of available resources dictates that only 05 of them will be eligible. How to objectively choose these 05 customers among the 07, knowing that each deserves as much the service as the others?

We realize that the Boolean logic is limited and inappropriate to provide an effective and relevant solution to this problem. Indeed, the proposition $p(x)$ can only have 02 possible values of truth: True (V) or False (F), which is insufficient to efficiently classify or order SCs while limiting frustrations. We must find other additional criteria such as seniority or type of package, to be taken into account in the decision-making. This leads us to introduce an additional merit variable as the degree of verification of the proposition $p(x)$. The appropriate mathematical tool in this case is the Fuzzy logic [10].

The Fuzzy logic is an extension of the Boolean logic based on the mathematical theory of fuzzy sets [11, 12]. It introduces the notion of the degree of truth-value of a proposition into the classical logic. It defines new operators called fuzzy set operators

e.g. the t-norm operator which corresponds to the fuzzy AND (intersection). Each t-norm has an associated t-co-norm which corresponds to the fuzzy OR (union).

For illustration, let us suppose that we have 02 fuzzy sets "big" and "old". Let an individual x such that $\alpha_{big}(x) = 0.7$ and $\alpha_{old}(x) = 0.5$. How to determine $\alpha_{big \cap old}(x)$? The corresponding fuzzy model is as follows:

Let T be a t-norm and S a t-co-norm, A and B 02 fuzzy sets. They are respectively defined by:

$T : [0, 1] \times [0, 1] \rightarrow [0, 1]$ and, $S : [0, 1] \times [0, 1] \rightarrow [0, 1]$.

The t-norm is then defined by: $\alpha_{A \cap B}(x) = T(\alpha_A(x), \alpha_B(x))$. The t-co-norm is then defined by: $\alpha_{A \cup B}(x) = S(\alpha_A(x), \alpha_B(x))$.

Modeling of the Authorization Process: Authorization's Functions. The authorization is the process of selecting the most deserving users of the service. It does 02 things: *ordering* users according to the merit and *selecting* the most deserving users according to the available resources for service's access.

Ordering the Users. At a time t, there is a set E of users $(E \subset U)$ who wish to access the service. However, the number n of services that can be provided is limited. The aim is to order the users for the system to choose the first n users. For that, the SP defines a number of linguistic variables. Users will be element of sets of Fuzzy values. To simplify the problem, we will only study one fuzzy value: e.g. the value *favorable*.

Let us now consider the universe that contains all the linguistic variables that the SP can consider, and let us define among them the linguistic variable merit that characterizes the merit of each user. In the context of fuzzy logic, it will have a fuzzy value "is deserving" which designates the merit rate of the considered user. The fuzzy set described by the fuzzy value "is deserving" of the linguistic variable merit will be the fuzzy intersection of all fuzzy sets associated with the fuzzy values favorable to each linguistic variable considered by the SP.

Example: Let us consider the linguistic variables *distance* and *age* chosen by an SP, who favors *far* and *aged* users. Then the fuzzy values *favorable* for these 02 variables will be *far* and *aged*, and so the linguistic value "is deseserving" will then be defined by: "*is deseserving*" = *far* ∩ *aged*. In other words, the fuzzy set defined by "is deseserving" will be the fuzzy intersection of fuzzy sets defined by *far* and *aged*.

The linguistic value "is deseserving" involves the logic AND ($\bigwedge$), so we must define the t-norm to be used. From now on, we will instead use the probabilistic t-norm: $T(x, y) = xy$, because it approaches at best the (geometric) average since we want a value that gathers all the truth-values equitably. The SP will then have a m-tuple of characteristic functions $\alpha = (\alpha_i)_{i=1}^{m}$, where α_i is the characteristic function of the fuzzy set whose fuzzy value is v_i. Thus, we introduce the *Ord* function for ordering users defined by:

$$\begin{aligned} Ord \ &: \mathcal{P}(U) \times \left([0, 1]^U\right)^m \rightarrow U^n \\ &(E, \alpha) \mapsto Ord(E, \alpha) = \bar{E} \end{aligned}$$

Where:

- E is the set of users to be ordered, those who wish to access to the service.
- α is the m-tuple of characteristic functions; m being chosen by the SP.
- $\bar{E}$ is a n-tuple of users. The users are then ranked in ascending order of their merits in $\bar{E}$; n being fixed by the SP.

Selection of Users. After ordering the users, it is now necessary to classify them according to the capacity of the SP in terms of the number of available services. An SP f has a number p (which can be infinite) of available services. We define the selection function *Select* by:

$$\begin{aligned} Select : U^n \times \bar{\mathbb{N}} &\rightarrow \mathcal{P}(U) \\ (\bar{E}, p) &\mapsto Select(\bar{E}, p) = E_0 \end{aligned}$$

Where:

- $\bar{E}$ is a n-tuple of users. This is the result of the *Ord* function.
- p is the number of attributable services at a given time.
- $\boldsymbol{E_0}$ is the set of users to whom the service is granted.

$$Property : card(E_0) \leq p$$

Practical Case. Let us consider WiABox.Net our WISP. It is assumed that there are 06 successful authenticated users $(u_1 \ldots u_6)$, who wish to benefit the access to the Internet. For exceptionally simplifying this case study, we will assume that the maximum number of attributable services is 04. Our goal is to determine among 06 users the 04 that deserve or merit the most the access to the service. The specificities of the SP are the following:

- On a scale of 0 to 10, the priority of the packageA is 10, the priority of the packageB is 8, the priority of the packageC is 7, the priority of the packageD is 3 and finally the priority of the packageE is 0.5.
- If x denotes the *distance* (in 10^4 m) between the SC and the WAP, then his priority is given by the function $x \mapsto e^{-x}$. The closer we are, the more priority we have.
- The *seniority* gives the priority to access the service by the fact that if you are a Blue (≤ 3 months of loyalty), we have a priority of 10/10. If you are a Veteran (>3 months and ≤ 5 years), then we have a priority of 3/10. Finally if you are an X-Veteran (>5 years), then we have a priority of 7/10. We assume the following information about the users were collected:
  - u_1: packageC, 4.7 km, 1 month;
  - u_2: packageA, 20 km, 3 years;
  - u_3: packageE, 380 m, 2 months;
  - u_4: packageD, 10 km, 5 years;
  - u_5: packageA, 5.5 km, 10 years;
  - u_6: packageB, 8.5 km, 4 years.

Linguistic Variables. The specificities of WiABox.Net show 03 linguistic variables that are *package, distance, seniority*. Each of these linguistic variables respectively proposes as a favorable fuzzy value: *best_package, close, satisfactory_duration*. We can then define the characteristic functions α_1 for best package, α_2 for close and α_3 for satisfactory duration.

Fuzzy Value 'Best_package': The characteristic function of the fuzzy set defined by this fuzzy value is given by:

$$\alpha_1(u) = \begin{cases} 1 & \text{if } package(u) = packageA \\ 0.8 & \text{if } package(u) = packageB \\ 0.7 & \text{if } package(u) = packageC \\ 0.3 & \text{if } package(u) = packageD \\ 0.05 & \text{if } package(u) = packageE \end{cases}$$

Where $package(u)$ is the package of the user u.

Fuzzy Value 'Near'. The characteristic function of the fuzzy set defined by this fuzzy value is given by:

$$\alpha_2(u) = e^{-distance(u)}$$

Where $distance(u)$ is the distance (in 10^4 m) between the user u and the WAP of the WISP.

Fuzzy Value 'Satisfactory_duration'. The characteristic function of the fuzzy set defined by this fuzzy value is given by:

$$\alpha_3(u) = \begin{cases} 1 \text{ if } fidelity\,(u) \leq 3\, months \\ 0.3 \text{ if } 3\, months < fidelity\,(u) \leq 5\, years \\ 0.7 \text{ if } fidelity\,(u) > 5\, years \end{cases}$$

Where $fidelity(u)$ is the fidelity of the user u.

Calculation of the Merit of the Users.

- *User u1*: The data of the user u_1 are the following: packageC, 4.7 km, and 01 month. Then, the merit of u_1 is $0.7 \times 0.67 \times 1 = 0.469$.
- *User u2*: The data of the user u_2 are the following: packageA, 20 km, and 03 years. Then, the merit of u_2 is $1 \times 0.135 \times 0.3 = 0.040$.
- *User u3*: The data of the user u_3 are the following: packageE, 380 m, and 02 months. Then, the merit of u_3 is $0.05 \times 0.962 \times 1 = 0.048$.
- *User u4*: The data of the user u_4 are the following: packageD, 10 km, and 05 years. Then, the merit of u_4 is $0.3 \times 0.367 \times 0.3 = 0.033$.
- *User u5*: The data of the user u_5 are the following: packageA, 5.5 km, and 10 years. Then, the merit of u_5 is $1 \times 0.576 \times 1 = 0.576$.
- *User u6*: The data of the user u_6 are the following: packageB, 8.5 km, and 4 years. Then, the merit of u_6 is $0.8 \times 0.427 \times 0.3 = 0.102$.

Selection of the Users. The results of the previous calculations are $u_1 : 0.469$; $u_2 : 0.040$; $u_3 : 0.048$; $u_4 : 0.033$; $u_5 : 0.576$; $u_6 : 0.102$. This implies the following:

$$Ord(\{u_1, u_2, u_3, u_4, u_5, u_6\}, (\alpha_1, \alpha_2, \alpha_3)) = (u_5, u_1, u_6, u_3, u_2, u_4)$$

Knowing that the limit number of users of our WISP was 04, we then have:

$$Select((u_5, u_1, u_6, u_3, u_2, u_4), 4) = \{u_5, u_1, u_6, u_3\}$$

In conclusion, the access to the service will be granted to users u_1, u_3, u_5 and u_6.

4.3 Accounting

Definition of the Measured Spaces [13].
Measured space of users. Let us consider U the set of users of our system. This set is influenced by the time. For the reasons of simplicity, we will choose as tribe on U, the discrete tribe $\mathcal{P}(U)$ designating all the parts of U. Since the entities are countable, we choose to define on $\mathcal{P}(U)$ the counting measure that is noted μ_U as a measure on this space. Finally, we obtain the measured space $(\mathrm{U}, \mathcal{P}(\mathrm{U}), \mu_{\mathrm{U}})$: space of users.

Measured space of providers. Let F denoting all the SPs of the system. In a similar way to U, one defines the measured space $(\mathrm{F}, \mathcal{P}(\mathrm{F}), \mu_{\mathrm{F}})$: space of providers.

Measured space of the actors. An actor will therefore designate either a provider or a user. We note A all the actors of the system. Then we have the definition $\mathrm{A} = \mathrm{F} \cup \mathrm{U}$. We thus define in a similar way for U, the measured space $(\mathrm{A}, \mathcal{P}(\mathrm{A}), \mu_{\mathrm{A}})$: space of the actors.

Space of time. In our model, we took into account the time dimension because the state of the entities of our system varies over time. The set of instants is $\mathbb{R}_+$. The tribe on $\mathbb{R}_+$ will be $\mathcal{B}_{\mathbb{R}_+}$, the Borelian tribe [13–15] of $\mathbb{R}_+$. We have taken the measure of Lebesgue λ[16] as the most appropriate one. Thus, the measured space $\left(\mathbb{R}_+, \mathcal{B}_{\mathbb{R}_+}, \lambda\right)$ results to the space of time.

Universe space. What is called the universe here is the resultant set of the Cartesian product of the actors' space and that of the time. If we denote Ω the universe, then by definition we will have $\Omega = \mathrm{A} \times \mathbb{R}_+$. We choose as tribe on Ω, the tribe product of $\mathcal{P}(\mathrm{A})$ and $\mathcal{B}_{\mathbb{R}_+}$, $\mathcal{T}_\Omega$ defined by $\mathcal{T}_\Omega = \mathcal{P}(\mathrm{A}) \otimes \mathcal{B}_{\mathbb{R}_+}$. Before defining the measure here, we must consider the following two facts: the space depends on the time and the time depends on the space.

- ***1st case: The space depends on the time***. In the universe, places are governed by the rules of time. To demonstrate this, we can take the Big Bang theory [8]. In our context, we know that A is countable and its cardinal depends on time. This leads to a chaotic model where we cannot describe in detail the variations of A depending on the time. In the remainder of the work, the notation M_t will be used to show that the set M is a function of time.
- ***2nd case***: ***The time depends on the space***. In fact, Albert Einstein has shown that time passes at different speeds depending on where you are. This is the principle of

the dilation of the time of the theory of general relativity [8]. In our context, we will suppose that the time defined on the set A does not depend on A. Indeed, we are in a situation where the elements of our system will be people or machines, which are in the same reference frame: Repository. In addition, that the nature of the environment is homogeneous for all actors. To suppose space as a variable of time would require us to take considerations irrelevant to our study. In conclusion: A depends on $\mathbb{R}_+$ but $\mathbb{R}_+$ does not depend on A.

Based on the previous assumptions, we can now define a measure on $\mathcal{T}_\Omega$. $\mathcal{T}_\Omega$ being the tribe product from tribes on A and on $\mathbb{R}_+$, it would be wise to define on $\mathcal{T}_\Omega$, the measurement product from the measurements on $\mathcal{P}(A)$ and on $\mathcal{B}_{\mathbb{R}_+}$. We note it μ_Ω and we will finally have: for $M = M_1 \times M_2 \in \mathcal{T}_\Omega = \mathcal{P}(A) \otimes \mathcal{B}_{\mathbb{R}_+}$, $\mu_\Omega(M) = \mu_\Omega(M_1 \times M_2) = \mu_A(M_1) \times \lambda(M_2)$.

μ_Ω defined on $\mathcal{P}(A) \otimes \mathcal{B}_{\mathbb{R}_+}$ self-extends uniquely to a measure on the tribute product $\mathcal{P}(A) \otimes \mathcal{B}_{\mathbb{R}_+}$.

We then obtain the measured space $(\Omega, \mathcal{T}_\Omega, \mu_\Omega)$: universe space.

For $E' \in \mathcal{T}_\Omega$, i.e. any measurable set E'; we define its measurement $\mu_\Omega(E')$ as follows:

$$\mu_\Omega(E') = \int_{E'} d\mu_\Omega$$

If $E' = E_t \times I$ with $E_t \in \mathcal{P}(A)$ *and* $I \in \mathcal{B}_{\mathbb{R}_+}$.

Thus,

$$\mu_\Omega(E') = \int_{E'} d\mu_\Omega = \int_I \left(\sum_{x \in E_t} 1_A(x) \right) dt$$

At this level, since E_t depends on time, then we first sum on the space's dimension, then on the time dimension. The opposite is not possible according to the reasons given above. Where:

$$\mu_\Omega(E') = \int_I \left(\sum_{x \in E_t} 1_A(x) \right) dt = \int_I \left(\sum_{x \in A} 1_{E_t} \cdot 1_A(x) \right) dt = \int_I \left(\sum_{x \in A} 1_{E_t \cap A}(x) \right) dt$$

$$\mu_\Omega(E') = \int_I \left(\sum_{x \in A} 1_{E_t}(x) \right) dt = \int_I \mu_A(E_t) dt = \int_I card(E_t) dt$$

Notes: The calculation here cannot be exhaustive because, until now we do not have any scheme, which "a priori", can help to describe the variations of E_t according to t (the time).

Definition of Functions

The function unit of period. The modeling of the space of time in a continuous way is not only not accessible but also especially unusable on the practical level since the service flows are most often quantified per unit of time defined by the SP and this unit

can itself vary over time. We introduce for this purpose, the function θ that, at any time t, returns the time unit corresponding to the service flow provided at this time. The function unit of period is defined hereafter.

$$\begin{aligned}\theta : \mathbb{R}_+ &\to \mathbb{R}_+ \\ t &\mapsto \theta(t)\end{aligned}$$

Example: For Internet access, θ is usually constant and is worth one second.

Service utilization rate function. It is defined by the function v as follows:

$$\begin{aligned}v : U \times \mathbb{R}_+ &\to \mathbb{R}_+ \\ (u,t) &\mapsto v(u,t)\end{aligned}$$

Where $v(u,t)$ is the utilization rate of the service by the user $\boldsymbol{u}$ at the time t.

Example: Let us consider the Internet access service. Here, the function v will be the bitrate (a measure of the quantity of digital data transmitted per time unit), e.g. $v(u,t) = 42Mbits/s$, $\forall (u,t) \in K \times \mathbb{R}_+$ where $K \subset U$ (Here the unit of period corresponds to 1 s (one second)).

Service Consumption's Measure. As we saw earlier, in practice, the continuous model has limitations. We must then consider that for the user u, the value of v at t is conserved on the unit of period provided by θ, i.e. $v(u,t') = v(u,t)$, $\forall t' \in [t, t+\theta(t)]$. Thus, the unit consumption from t will be worth:

$$unit(u,t) = g(u,(t,t+\theta(t))) = v(u,t) \times \theta(t)$$

It is the consumption at t. This defines the unit function. Finally, the function $\hat{\boldsymbol{g}}$ (discrete modeling) by:

$$\begin{aligned}&\hat{g} : U \times D \to \mathbb{R}_+ \\ &\quad \big(u,(t_i,t_f)\big) \mapsto \hat{g}\big(u,(t_i,t_f)\big) = \sum_{k \in J_{(t_i,t_f)}} unit(u,t_k)\end{aligned}$$

Where the sequence (t_k) is defined by:

$$\begin{cases} t_0 = t_i \\ t_{k+1} = t_k + \theta(t_k)\ \forall k \geq 0 \end{cases} \text{and } J_{(t_i,t_f)} = \big\{k \in \mathbb{N} / t_i \leq t_k \leq t_f\big\}$$

The functions *unit* and $\hat{g}$ being defined, we can now define their respective set versions that are *Unit* and $\hat{G}$.

$$\begin{aligned}Unit : \mathcal{P}(U) \times \mathbb{R}_+ &\to \mathbb{R}_+ \\ (E,t) &\mapsto Unit(E,t) = \sum_{u \in E} unit(u,t)\end{aligned}$$

$$\hat{G} : \mathcal{P}(U) \times D \to \mathbb{R}_+$$
$$\big(E, (t_i, t_f)\big) \mapsto \hat{G}\big(E, (t_i, t_f)\big) = \sum_{k \in J_{(t_i,t_f)}} Unit(E, t_k) = \sum_{k \in J_{(t_i,t_f)}} \sum_{u \in E} unit(u, t_k)$$
$$= \sum_{u \in E} \sum_{k \in J_{(t_i,t_f)}} unit(u, t_k) = \sum_{u \in E} \hat{g}\big(u, (t_i, t_f)\big)$$

Example: In the case of Internet access service, if we consider $\theta(t)$ constant and equal to 1 s, then, for a user u benefiting from a constant bit rate during the time of v $v(u,t) = 2Mb/s$, its consumption at time $t' = 30\,s$ is $unit(u,t') = v(u,30) \times \theta(30) = 2 \times 1 = 2Mb$.

In addition, that from 30 s to 45 s will be:

$$\hat{g}(u,(30,45)) = \sum_{k \in J_{(30,45)}} unit(u,t_k) = \sum_{k \in J_{(30,45)}} 2 = 2 \times (45 - 30 + 1) = 32Mb$$

The function Resource. Considering the WISPs, one defines the resource function noted r that, for a given WISP f, gives the remaining quantity of the principal resource at the time t. We then have:

$$r : F \times \mathbb{R}_+ \to \mathbb{R}_+$$
$$(f,t) \mapsto r(f,t)$$

The principal or main resource is at the origin and of the provided service. Without it, there could be no real, genuine and adequate provision of service. E.g. Let us consider the case of a printing service. If one supposes that the WISP f only has 400 paper sheets at the beginning of the day $(t = 0)$, then $r(f, t = 0) = 400$. The set version of the resource function r is the function R defined by:

$$R : \mathcal{P}(F) \times \mathbb{R}_+ \to \mathbb{R}_+$$
$$(E,t) \mapsto R(E,t) = \sum_{f \in E} r(f,t)$$

The function Service Access Capability. It is necessary to couple to the resource function r, one another called *service access capability* noted c. For a WISP f, the function c specifies the maximum number of critical tools he has for the simultaneous provision of the service. In other words, it characterizes his capacity to simultaneously serve users requesting access to the service. We then define the function c as follows:

$$c : F \to \mathbb{R}_+$$
$$f \mapsto c(f)$$

Example: Let us consider a cybercafé that has 05 computers available for a full-time use. The WISP f, here can only serve simultaneously up to 05 users at the maximum. Thus, we have $c(f) = 5$. The set version of the function capacity c is the function C defined by:

$$\begin{aligned} C : \mathcal{P}(F) &\rightarrow \mathbb{R}_+ \\ E &\mapsto C(E) = \sum_{f \in E} c(f) \end{aligned}$$

The function Accessibility. This is the function that, at any time t, returns the remaining number of instances of the service that a SP f can simultaneously offer at this time. It is noted a, and somehow, the capacity $c(f)$ is its maximum value. In other words, it is the difference between the overall capacity and the number of service instances already offered. For that, let us introduce the following function o which returns, at any time t the number of instances of service already offered by f:

$$\begin{aligned} o : F \times \mathbb{R}_+ &\rightarrow \mathbb{R}_+ \\ (f,t) &\mapsto o(f,t) \end{aligned}$$

Moreover, we will have:

$$\begin{aligned} a : F \times \mathbb{R}_+ &\rightarrow \mathbb{R}_+ \\ (f,t) &\mapsto a(f,t) = c(f) - o(f,t) \end{aligned}$$

Example: Let us take a cybercafé, knowing that the WISP f has a maximum of 05 computers and that 03 of them are currently used by the SCs at the time t: we will have $a(f,t) = 5 - 3 = 2$.

Properties: Let Δ denoting a duration defining a period.

1. $\forall (E,t) \in \mathcal{P}(A) \times \mathbb{R}_+,\ R(E \cap F, t) \geq \hat{G}(E \cap U, (t, t+\Delta))$;
2. $\forall (f,t) \in F \times \mathbb{R}_+,\ r(f,t) = 0 \Rightarrow a(f,t) = 0$

The first property translates the fact that the overall consumption in a locality E of all users over a period can never exceed the services production of their WISPs. The second reflects the fact that one cannot access the service when the resource is exhausted and a fortiori consume it; the accessibility of the service is dependent on the resource.

Vocabulary: Let $(f,t) \in F \times \mathbb{R}_+$ be fixed:

- When $r(f,t) = 0$ then "the service is unavailable".
- When $a(f,t) = 0$ then "the service is inaccessible".
- "An unavailable service is, a fortiori, inaccessible".

4.4 Pricing

Next to the accounting of the services, it is question of fructifying the consumptions of SCs. For that, the WISPs will have to define a pricing policy or a tariff plan, i.e., to associate tariffs with the various predefined packages [17, 18]. A WISP f defines the set FO_f of all his packages for the considered service. It is then a question of defining a function Pr that will link a package to a tariff for a given WISP $f \in F$.

$$\begin{aligned} Pr : FO_f &\rightarrow \mathbb{R} \\ x &\mapsto Pr(x) \end{aligned}$$

Each WISP will define this function Pr, which will thus make a correspondence between its packages and the unit prices of the service.

Note: For the definition of service packages, the WISP will rely on the qualities of the service.

4.5 Billing

The service billing [19] is a part of the PBP model. Let us note that charging in our context consists to apply a price to a consumption. We then define the following function:

$$\begin{aligned} Bil : U \times \mathbb{R}_+ &\rightarrow \mathbb{R}_+ \\ (u, q) &\mapsto Bil(u, q) = p \end{aligned}$$

- u is the logical user whose features include the package;
- q is the consumption of the service by the user u;
- p is the price calculated for the quantity q consumed by the user u.

4.6 Payment: Prepaid, Postpaid and on-the-Go

Prepaid. The access to the service is conditioned by the billing. In other words, the user pays first before actually consuming the service. This billing mode is similar to that of a fuel pump at the service station. When the user pays an amount of money, his tank is topped up with the fuel (the resource in our case). His tank will progressively empty when he will drive during a certain period. As illustrated in Fig. 2, after the fuel finishes or reaches a certain level, he will top up again the fuel in the tank. Here, consuming the service consists to driving the car.

Postpaid: The payment method illustrated in Fig. 3 below is made after the consumption of the service.

On the Go: The payment is made as the user consumes the service.

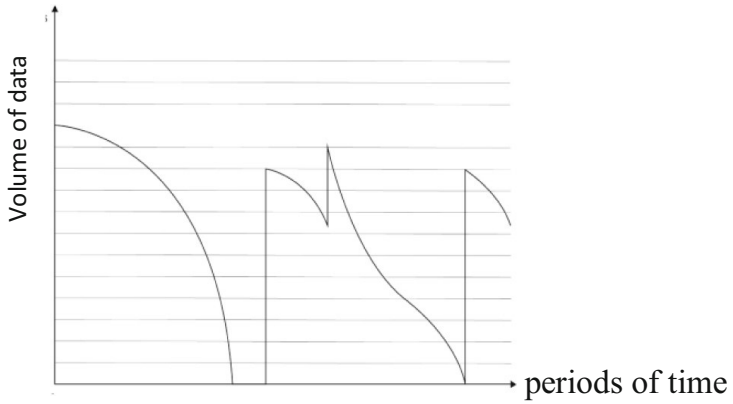

Fig. 2. Illustrative representation of the prepaid mode

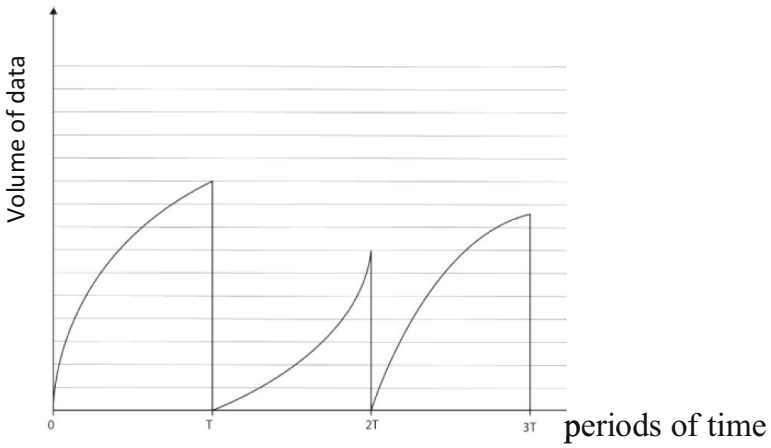

Fig. 3. Illustrative representation of the postpaid mode

5 Discussions and Open Issues

This model can be used to build generic profiles of users in a multi-operator (telecom or internet) context. Consumer scoring can thus be optimized by coupling a machine-learning engine at the Authorization component. To be more efficient, we would obtain classes of users to address offers of targeted services, ranging from free to paid ones. In a rural community, health centers, schools and colleges can benefit from free services, while small businesses can financially contribute to help the former. This is considered by the same platform that implements a unique model: 3APBP.

We can also use this model for scoring merchants and customers of an e-marketplace or e-commerce platform; it would thus serve as a basis of a services' recommendation system to clients and service providers.

Finding a family of mathematical functions to model the Payment component is not easy, that is why we have instead illustrated it with graphs while continuing our investigation.

6 Conclusion and Future Works

In this paper, we have proposed a formal approach based on Black Box Specification Method by Entities for modeling first network services, then the access and control mechanisms and functions of these services. This resulted into the 3A-PBP (authentication, authorization, accounting - pricing, billing and payment) model which includes each element of the AAA standard protocol augmented by the PBP services. Each entity, modeled as a mathematical function, represents one component of the whole system. The output parameters of each component are the input parameters of another one.

We first gave a definition of a network service, then we built the mathematical ecosystem based on set and fuzzy logic theories and finally we brought out all the relevant functions and properties that contribute to the solution of our initial problem.

This work forms the basis of a long-term project for the operationalization of wireless community networks in developing regions. This is why we have designed it more generic to facilitate its scaling for better adaptation. Our wireless community network model must support the operation of proximity and long-range digital services with minimal customization effort. The implementation of this 3APBP model, which extends the best-known AAA model, is in progress, in order to compare its performance with that of some AAA frameworks such as FreeRADIUS.

Acknowledgments. Special gratitude respectively goes to Alexander von Humboldt Foundation; GIZ-CIM Organisation; the University of Applied Sciences Bonn-Rhein-Sieg and the CETIC Project of the University of Yaounde 1 for their various supports.

References

1. Luiz, A.D.: Wireless community networks: evolution and technical challenges. In: Workshop on Nationwide Internet Access and Online, Applications, Dhaka, Bangladesh (2004)
2. Center for Neighborhood Technology: Community Wireless Networks: Cutting Edge Technology for Internet Access (2006)
3. Frappier, M., St-Denis, R.: EB3: an entity-based black-box specification method for information systems. Softw. Syst. Model. **2**, 134–149 (2003)
4. Freifunk. http://freifunk.net/. Accessed December 2018
5. de Laat, C., Gross, G., Gommans, L., Vollbrecht, J., Spence, D.: Generic AAA Architecture, Internet Society, RFC2903 (2000)
6. The FreeRADIUS Project. http://freeradius.org/. Accessed June 2018
7. freeDiameter. http://www.freediameter.net. Accessed June 2018
8. Hawking, S.: A Brief History of Time. Bantam Books, New York (1988). Physics Bulletin
9. Matt, B.: Computer Security: Art and Science. Addison-Wesley, Reading (2004)
10. Dernoncourt, F.: Introduction à La Logique Floue (Introduction to fuzzy logic) (2011). developpez.com
11. Gacogne, L.: Éléments de logique floue (Element of Fuzzy logic). Hermès (1997)
12. Leekwijck, W.V., Kerre, E.E.: Defuzzification: criteria and classification. Fuzzy Sets Syst. **108**(2), 159–178 (1999)

13. Berberian, S.K.: Borel spaces. The University of Texas at Austin (1988)
14. Navara, M.: An algebraic generalization of the notion of tribe. Fuzzy Sets Syst. **192**, 123–133 (2012)
15. Anatolij, D.: MV-observables and MV-algebras. J. Math. Anal. Appl. **25**(9), 413–428 (2001)
16. Cenap, D., Birsen, S.: Non-newtonian comment of lebesgue measure in real numbers. J. Math (2017). https://doi.org/10.1155/2017/6507013
17. Chen, X.: Resource allocation and pricing in virtual wireless networks. Masters Theses 1896, University of Massachusetts (2014)
18. Belghith, A., Trabelsi, S., Cousin, B.: Realistic per-category pricing schemes for LTE users. In: Proceedings of IEEE 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WIOPT) (2014)
19. Lakhtaria, K.I., Jani, N.N.: Design and modeling billing solution to next generation networks. Int. J. Adv. Netw. Appl. **1**(5), 290–294 (2010)