Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2017, Vol. 9, No. 1-2

Design and Implementation of ForCES Protocol

Pedro Luis González Ramírez

Dept. of Electronics, Universidad Central

Cra 5 No. 21-38, Bogotá (Colombia)

E-mail: pgonzalezr1@ucentral.edu.co

Jaime Lloret

Instituto de Investigación para la Gestión Integrada de Zonas Costeras

Universitat Politécnica de Valencia

Camino Vera s/n, 46022, Valencia (Spain)

E-mail: jlloret@dcom.upv.es

Susan Martínez Cordero, Luis Carlos Trujillo Arboleda

Dept. of Electronics, Pontificia Universidad Javeriana of Colombia

Cra. 7 No. 40 B -36, Bogotá (Colombia)

E-mail: susan.martinez@javeriana.edu.co, trujillo.luis@javeriana.edu.co

Received: March 17, 2017 Accepted: May 20, 2017 Published: June 30, 2017

DOI: 10.5296/npa.v9i1-2.10943 URL: http://dx.doi.org/10.5296/ npa.v9i1-2.10943

Abstract

This paper proposes the design and implementation of the Forwarding and Control Element Separation (ForCES) Protocol. Specifically, Logical Point (FP) of the ForCES Architecture, which is strictly the communication between the Control Element (CE) and the Forwarding Element (FE). It is a flexible and reprogrammable architecture that is established within the specifications issued and defined by the ForCES working group and consists of elaboration of a protocol that carries information between both elements. In order to verify the correct functioning of the implemented ForCES protocol, it is we provide a network testbed scenario, which consists of an application client-server. Each device is equipped with the application which based on Java language, that allows the researcher to be able to

compare the typical functionality of a conventional router with a router based on architecture ForCES. It allows taking advantage of the benefits of this architecture to reprogram different and new functionalities.

Keywords: Forwarding and Control Element Separation (ForCES) Protocol, programmable networks and systems, flexible architectures, open architecture, Structs LFBs, testbed ForCES, client-server application.

1. Introduction

Achieving end-to-end communication is one of the challenges of Internet Protocol (IP) networks and therefore it is the success of the great expansion of the internet in the world. According to this principle, most of the functionalities of a network node are essentially directed to packet routing or other tasks in which is seeking to improve the performance. Because of the above, most of the research are directed to improve the performance, the security, measure processing and Quality of Service (QoS).

However, this requirement to develop new network functionalities and services, such as quality of service and others, are not easy to implement with the architecture of the current network nodes. These systems are closed and integrated vertically, this means that the manufacturer distributes both the hardware that is responsible for the communications and the data, as the software that implements the control plane, without the possibility of modification. This dependency with the manufacturer limits interoperability and makes it difficult to develop new services since in order to be able to be implemented they must go through a long and arduous process of standardization and testing before being approved by the manufacturer. This is one reason why relatively recent mechanisms, such as IPv6, have not yet been implemented on a large scale.

In this situation, research was started on programmable networks and systems with flexible architectures, which initially emerged as two lines of parallel research, but which are now interrelated to each other, forming a single one.

Programmable networks are mainly active networks with open interfaces, which allow solving processing needs in the intermediate nodes of the network and its objective is to minimize the time of implantation of new protocols and services, from the definition of a common architecture and flexible network in which routing systems could be reprogrammed, thus adapting the behavior of the network to conditions of growth and change of technology, without affecting processing.

With respect to the above, the standardization bodies have begun to define the standard and open interfaces between the different elements and layers that form a network node. This is intended to promote interoperability between manufacturers and to simplify the development of new protocols so that they could be quickly adapted to systems of different manufacturers. In short, what is defended is the idea of designing programmable network nodes with extended and scalable capabilities.

Under this focus, the following paper seeking to use this new proposal oriented to flexible and reprogrammable architectures, to approach new research using the ForCES architecture and implementing the ForCES protocol [1], defined and created by the Internet Engineering Task Force (IETF) through the work group Forwarding and Control Element Separation (ForCES) [2].

This paper demonstrates the operation of the ForCES Protocol, allowing check that the data that are sent from one end to the other, arrive under the conditions set out in the specifications, are correct and have been programmed without errors.

The rest of this paper is organized as follows. In Section 2, we discuss existing related works. Section 3 shows the basic concept of conventional router architecture and its functions and then it is compared with ForCES router architecture in order to show its advantages. The implementation is explained in Section 4 and Section 5. Section 6 shows the interoperability test based on scenarios. Section 7 proposes a testbed. Finally, Section 8 shows the conclusion and future works.

2. Related works

This section shows some published work introducing to the ForCES protocol and the networks programmable. Also, it provides the results of some implemented schemes and network topologies similar. Most of these works, it is based on the same main: separate the mechanisms of functioning of the data plane and the control plane.

Wang et al. [3] presented a router called ForCES-based router (ForTER). Firstly, the implementation architecture of ForTER is discussed. Then, a layered software model, which well illustrates ForCES features, is proposed. Based on the model, design, and implementation of the CE and FE elements. Moreover, security for ForTER is considered and an algorithm to prevent Denial-of-Service (DoS) attacks is presented. Lastly, experiments of ForTER are illustrated for routing and running routing protocols, network management, DoS attack prevention, etc. The experimental results show the feasibility of the ForTER design. Consequently, the ForTER implementation basically testifies the feasibility of ForCES architecture. These way ForCES gains its competitive advantage over traditional router architectures in flexibility, programmability, and cost-effectiveness.

For instance, Urueña et al. [4] explained that adding new protocols and network services is a long, complex and costly process, due to many causes, but especially because routers are still of proprietary operative systems. In this sense, it presents a low-cost transition architecture for programmable network nodes named Simple Assistant-Router Architecture (SARA), that allows a commercial router to easily extend its capabilities by delegating the advanced packet processing to a cluster of assistants, which greatly simple is the development and dynamic deployment of new network services.

In [5], Takourout et al. presented a study addresses the separation of the control plane and the forwarding plane in Next-Generation Routers. XML syntax is used to model, on the

one hand, the suitable elements of the control plane and the forwarding plane. Implementation and prototyping results indicate that this syntax type for modeling provides a flexible platform, which is truly able to manage the heterogeneity of the material implementing the Forwarding Plane.

Also, the importance about the different ways to configure the LFBs are presented by Wang et al. [6] and [7] where it shows throughout this work the way of solving effectively the display problem of LFB topology in ForCES research, using an application on platform J2EE, with which collection and organization of topology data can be done, together with a new auto-layout arithmetic.

Some of these works have followed the ForCES architecture and its ForCES protocol, but others use this principle to reformulate new solutions, such as the Software Defined Networks (SDN), which has become a new way to make dynamic topologies. They have great potential in both the creation and development of new network protocols. Nunes et al. [8] and Feamster et al. [9] survey the state-of-the-art in programmable networks with an emphasis on SDN, providing a historic perspective of programmable networks from early ideas to recent developments. It includes the OpenFlow protocol which is similar and it is compared with the ForCES protocol to a functional level. Recent investigations are focused on using SDN for multimedia delivery [10].

3. Theoretical framework

In this section, it is shown the existing related between the conventional router architecture [11] and ForCES router architecture [1], through of its basic concepts and its functions and then it is compared as it can see in figure 1.

3.1 Conventional router architecture

A router is an electronic device, capable of distributing each packet of information it receives and deciding the most convenient way to send it to its destination. A router is in charge of performing the follow fundamental tasks: routing and forwarding of packets [11].

3.1.1 Routing

The routing process constructs a view of the network topology and calculates the best routes for that a packet reaches its destination and it does this through information that is exchanged between neighboring routers, using different routing protocols [12] [13]. The best routes are stored in a structure called Forwarding Table.

3.1.2 Forwarded Packets

The process of "Forwarding" of packages moves a package from an input interface of a router to an output interface appropriate, based on the information contained in the forwarding table.

3.1.3 An IP router functionalities

In summary, the routers consist of the following basic components:

Various networks interface for interconnecting networks, processing modules, storage modules and an internal interconnection unit (or a switching structure). Typically, packets are received on an input interface, processed by the processing module and possibly stored in the storage module and then they are sent through an internal interconnect unit to an output interface which transmits them to the next hop and from there to their final destination. As shown in figure 1, the router operates on two different planes:

- **Control plane:** It is the plane where the router report which is the appropriate output interfaces for the transmission of the packets to determinate destinations.

  The Routing Table or Routing Information Base (RIB) is constructed in the Routing Control Plane. The RIB can be used by the forwarding plane to search the external interface for a particular packet, or, depending on the implementation of the router.

  The control plane constructs the routing table with the knowledge of its local interfaces, static routes codes and the exchange of information with other routers routing protocols. The routing table stores the best routes to certain destinations in the network, routing metrics associated with those routes and the road to the next router.

  Routers maintain the State of the routes in the routing table, but this is very different to maintain the State of individual packets that have been passed down.

- **Forwarding plane:** Forwarding plane is also known as data plane. By the function of forwarding of Internet Protocol (IP), the design of routers seeks to reduce to a minimum the state of the information stored in the individual packages. Once a packet is sent, the router must maintain no more than the statistical shipping information. It is the end point of shipment which maintained the error information.

  Among the most important forwarding decisions is deciding what to do when there is congestion. At this level also built the table of forwarding, which is consulted by the router to determine the output interface where you need to forward an incoming packet, then each entry in the table of forwarding maps an IP prefix to an exit interface. Depending on the implementation, entries should contain additional information such as Media Access Control (MAC) addresses, for the next hop and statistics about the number of packets forwarded via an interface.

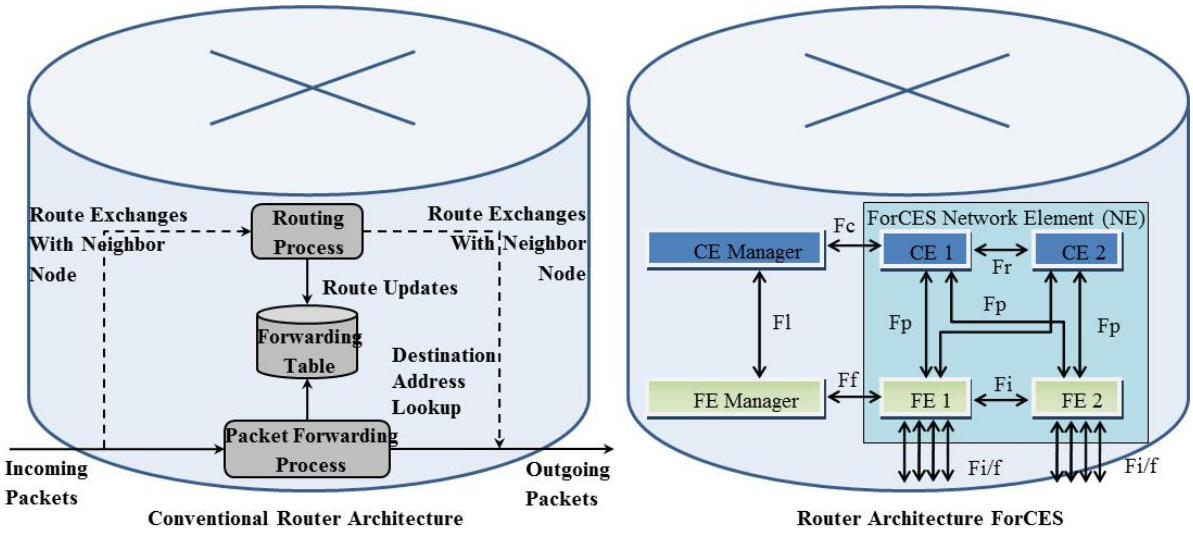

Figure 1. Comparing Architectures

Although internally the network equipment has different data and control planes, this logical separation does not allow each plane to evolve completely independently, since the manufacturer is the only one that can innovate in either field. The implementation of this protocol is a sign of the progress that has been made so far to achieve this separation.

3.2 ForCES architecture

Forwarding and Control Element Separation Architecture, it is a new proposal for the development of network routers. This means that, be separate the mechanisms of functioning of the data plane and control plane and uses the ForCES Protocol to communicate it, with the purpose that any improvement that is made in the elements CE and FE is can make of way separated. Unlike the architecture of the conventional routers, the internal operation between the control plane and data plane only it is known by the manufacturer of the routers and only the manufacturer can make modifications or improvements.

ForCES architecture is a flexible model written and public by Request For Comments (RFC) [2] to which everyone has access and that you can read the document and perform their own implementations allows you to reprogram and improve elements of the router without intervention or permission of the manufacturers because it is an "open architecture" on which a working group cooperates each other so that any other researcher or user can make contributions to this implementation. This article demonstrates the implementation of the ForCES Protocol, which enables connection and transport of information between CE and FE elements.

Within the ForCES architecture, there are several connection points as shown in figure 2. The Logical Point (FP) connection, it is where the development of this article is focused, since this is where the ForCES protocol is implemented. For the implementation of this Protocol following the architecture, be proposing that the element CE, FE and the protocol ForCES that connects them, be programmed using the Java language. Any other connection

point shown in the architecture in accordance with RFC5810 (page 11) [2] specification and the interaction between the FE Manager (FEM) and the CE Manager (CEM) which is part of the pre-association phase, are beyond the scope of this article.

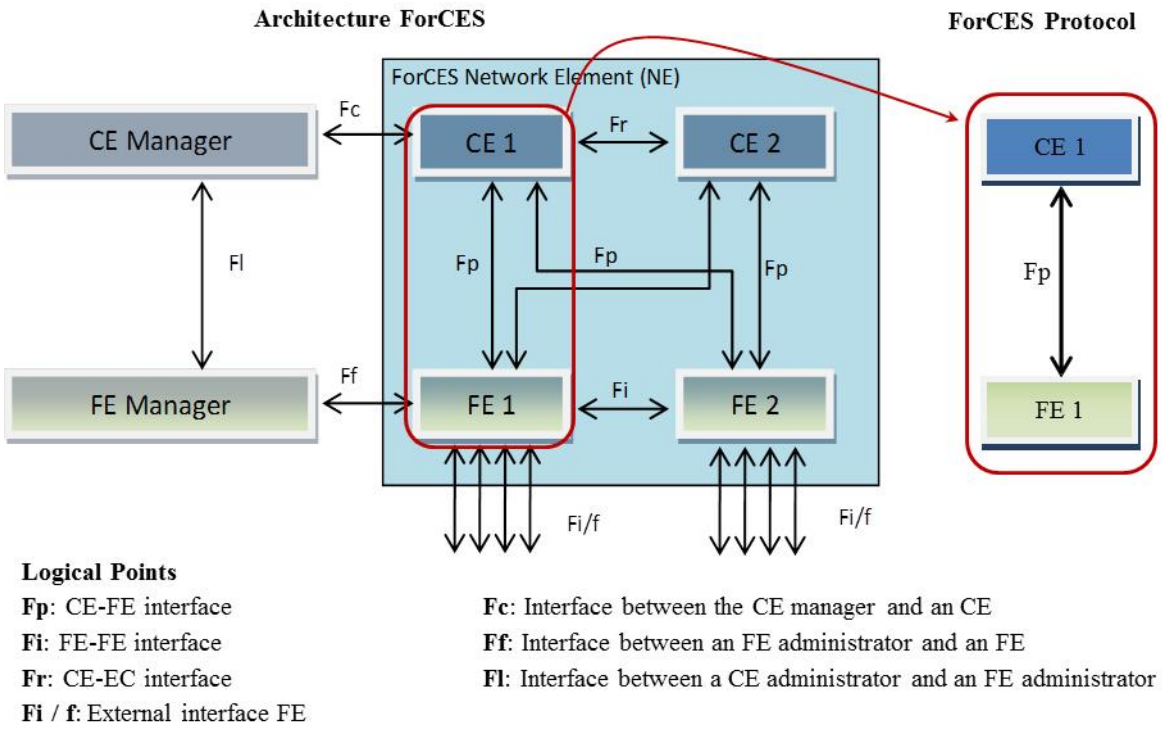

Figure 2. ForCES Architecture, Taken from RFC 5810 (page 11).

4. Proposal development

In this section is present the analysis of the RFCs and general design of the protocol.

In 2001 the IETF created the working group called ForCES to define a standard protocol for communication between the data plane and the control plane of IP routers based on distributed, flexible and programmable architectures, using the same name ForCES. In this work, the main sources of information were the RFCs: 5810, 5811, 5812, 6956, and the draft-ietf-forces-interoperability-04. This allowed approaching the idea of this implementation, reading and following each of its indications for elaboration more the contributions of the authors.

The specification RFC5810 (page 5) defines in its introductory part that the ForCES protocol works in a master - slave mode, where the CEs elements are masters and the FEs elements slaves and that the Protocol should include transport commands to configure the information of the Logical Functional Blocks (LFBs) in the FE and all messages of association, configuration, query, status and notification of events. The architecture also presents the use of the FE model of ForCES [14] where to be defined the Logical Functional Blocks (LFBs), using Extensible Markup Language (XML). All the process of elaboration of

the Protocol is based on the stack of the ForCES Protocol [2] that is has designed for its implementation and that it is can appreciate in figure 3.

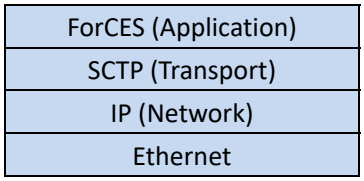

Figure 3. ForCES Protocol Stack.

Because of the above and according to the ForCES architecture and the document "draft-ietf-forces-interoperability-04" [14] is defined the test scenarios of the protocol. Part of the design exposed in this article, consists of the design of elements based on client - server consoles, allowing the establishment of the protocol between the CE and the FE "step by step" through commands. The design of the messages and the files XML for the LFBs is a general proposal of the ForCES group, but the "form and use of the application" and the "distribute and topology configuration of the LFBs" is one of the contributions of the authors of this article.

According to the above and following the RFC 5810 specification that shows the architecture of a flexible router and programmable, it is shown in figure 4 the designing the scenarios "step by step" through of a use case diagram and a diagram exchange of protocol messages between the CE and FE.

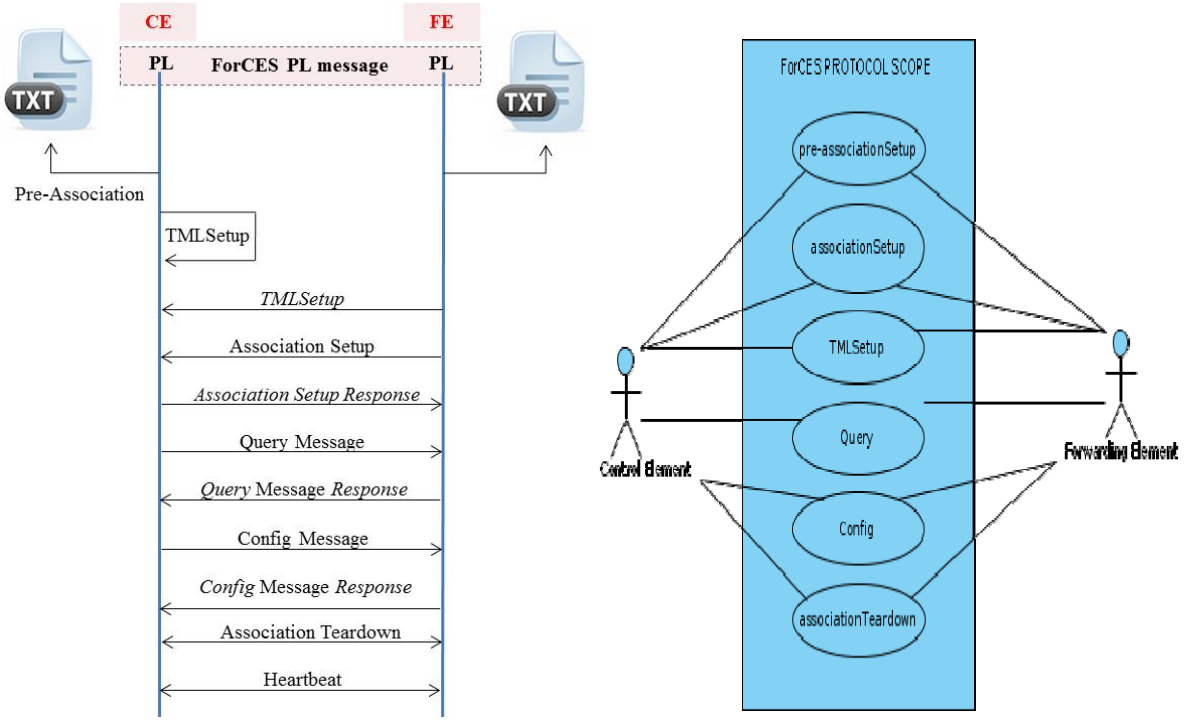

Figure 4. Use Case diagram and ForCES Protocol Messages

4.1 The CE and the FE

The CE is the element of control plane and the FE is the element of the data plane, both the CE and FE element internally contain two layers, the layer of the specific messages of the ForCES Protocol called Protocol Layer (PL) and the layer of Encapsulation and Transport of the ForCES Protocol Messages called Transport Mapping Layer (TML). Which in its turn interconnected among themselves, the Figure 5 shows their interaction. The control plane is the responsible for taking decisions and manage of logical way, what it happens with to some tasks that require the data plane, where its main function it is based on packet forwarding. But for some these logical operations, it is the CE of the control plane who determines through of the Logical Functional Blocks (LFBs) as it should work.

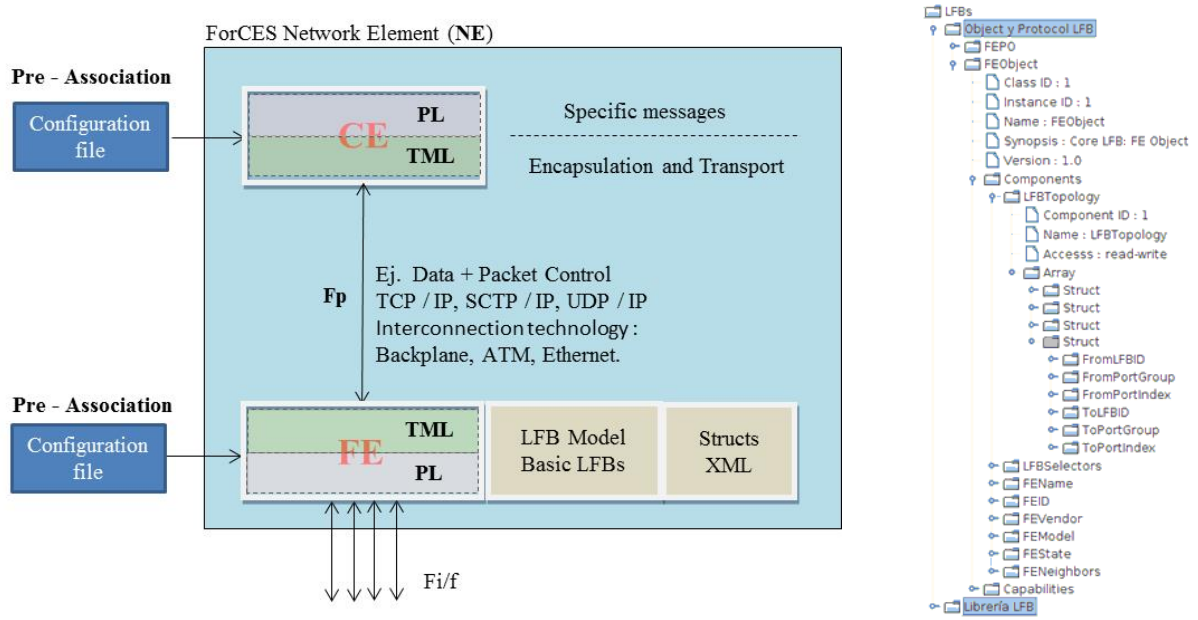

Figure 5. Interaction of the Protocol between CE and FE

4.1.1 The CE

The Control Element (CE) is the control element based on software, this is the logical part of the system and within this element are developed algorithms that allow taking the decisions of operation of the router. Specifically, in this article, these operations are limited only to the operations the establishment of the ForCES Protocol and the topological organization of the LFBs in the FE, as indicated by the LFB library [14], which contains two organization examples for the functionalities: IPv4 and ARP. It also performs the basic LFB control operations of the FE as they are: the LFB Protocol Object called (FEPO) and the FE Object called (FEO).

The CE uses a user interface in the form of a window or also called console, which has a text box to display the exchange of messages in a binary format and the events that occur during the operation of the Protocol, also has another text box where are written the command lines that you want to run.

4.1.2 The FE

The FE is a network element dedicated to packet forwarding and it is made up of software and hardware. In the hardware is the network cards and to the level of software is made the communication through protocol ForCES with the CE. In the FE is find also the PL and TML layers, the LFBs structures based on XML, which are loaded by libraries to give the functionalities that must have the FE. In this case and for this article will be found in the FE, three XML libraries [15]: two that contain the basic LFBs called LFB Protocol Object (FEPO) and the FE Object (FEO) and an example library called the LFB library contain the entire structure of the IPv4 functionality.

Within the design of this console, the management and reconfiguration of the LFBs are done through a folder tree, which is made available to the CE when the pre-association phase has been executed and finalized.

The FE window or console is designed to display three elements: ForCES PL messages in 32-bit line format, LFB folder tree libraries, and LFBs topology. In the line of instruction of the FE operates the command Topology which once executed takes the data that has been stored in the XML structure and draws them.

4.2 The LFBs

In ForCES is not possible to define all operations of the data plane of the router. Instead, ForCES defines an XML language that allows the construction of a generic model of the Forwarding Element, in which the routing process is divided into several stages. According to this model, an FE is formed by Functional Logic Blocks (LFB) interconnected with each other. Each LFB models a part of the process of forwarding an IP datagram. Therefore, the so-called ForCES protocol will be a control protocol that will allow the CE to modify the attributes of each LFB of an FE.

It is also possible to change the topology of the LFBs [16], changing the way in which they are interconnected, so the ForCES protocol can be used both to access the state of the LFBs and to configure their behavior. Because of this, each LFB has status attributes and configuration attributes. These modifications are made through the query and config messages, respectively.

The FEs are controlled by the CEs through the LFBs, which in turn are managed by a structure defined by the RFC5812 specification [14], based on markup language or labels called Extensible Markup Language (XML), which is a format based on text and specifically designed to store and transmit data.

The data to consult or transmit in the protocol are the attributes that form the functionalities of each of the LFBs and that are organized structurally making use of the XML language.

An LFB is created from a ClassID, an InstanceID and a variety of components and within these may have more components and within each component a collection of attributes, it also has an input port and an output port, in some cases a group of ports is used when an

LFB connects with several LFBs and is referred to as an input or output branch.

To make a connection between two LFBs, the input and output must have the same port that is identified with a 32-bit number. To address a component, the "Path" command was used following a few instructions are given by RFC5812 (page 32), where it is shown, how it is addressed one component within another. The graph below is a compilation of several graphs found in RFC5812 (pages 30 to 32), which illustrates how it is an LFB and the parts that make it up. This representation of an LFB is only a graphic way of explaining it, because of in reality its description it is made essentially through of parameters written in XML language and that are somewhat complex to understand.

The LFBs [16] can have multiple inputs and outputs, which it can be grouped together to form input/output groups. In general, the outputs of an LFB are grouped when all of them transmit the same type of packets, but must it processed by different LFBs (for example: an IPv4 datagram can send to another FE or forwarded by another FE interface that received it), while the exceptions in the processing of the packages have their own outputs. Therefore, LFBs with multiple outputs must choose one based on the contents of the packet or its associated metadata. LFB architecture is shown in figure 6.

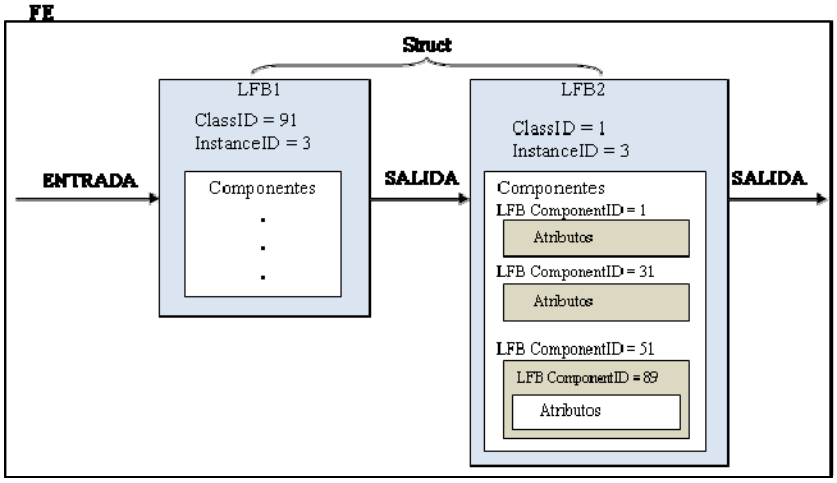

Figure 6. LFB architecture

The inputs/outputs of an LFB are always attached to other LFBs, so that an LFB receives a packet through some of its inputs, processes it and forwards it to one of its outputs. The only exceptions to this flow behavior are LFB Dropper that has no outputs but serves to discard packets and the LFBs of the port that represent the physical ports of the FE and therefore when they receive a packet, send it through the line, while their departure packages have just been received by the FE. In fact, even communication within the router (CE-FE or FE-FE) is modeled with LFBs, which allows ForCES to be independent and inter-FE communication of the interconnection technology used.

From the graphic representation can be understood and designed how to connect different types of LFBs in order to group them to perform a specific function.

The LFBs that are available maintain an XML structure and whenever an additional

function in the FE is desired an XML library must be added with all the description of these functionalities, in the case of the FE of this article it has two base LFB libraries and one example. These LFBs are not known by the CE, any changes or additions to LFBs libraries are only known by the FE and for the EC to know them uses the Query message of the ForCES protocol by means of the Query command to know the state of the variables of the libraries LFBs.

The values that are inside each of the attributes of the LFBs components are by default zero (0), that is, the content of these attributes depends on the functional design to be implemented.

4.3 The PL

Encapsulation of ForCES messages is carried out in the Protocol Layer (PL) and then it is passed to the Transport Mapping Layer (TML), to be transported. It is shown in figure 7.

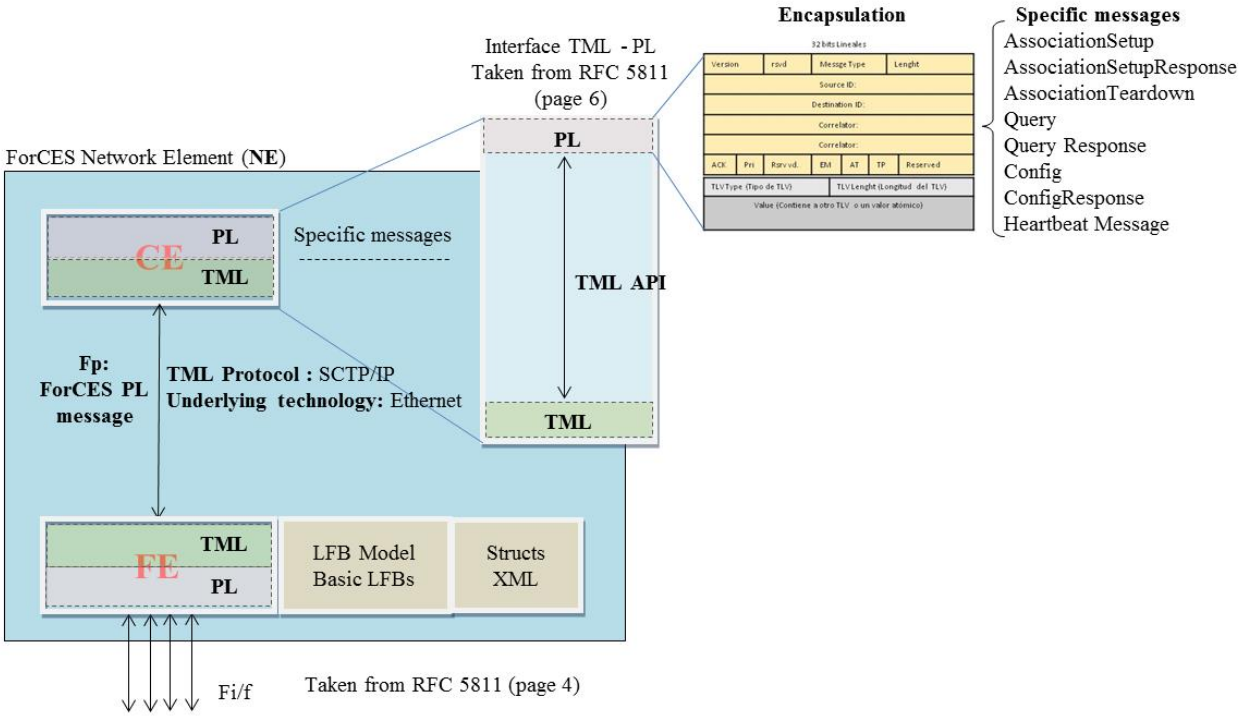

Figure 7. The PL

This process of encapsulating the ForCES messages is based on a header and a body of the message.

4.4 Construction of the ForCES PL Messages

ForCES Protocol Layer (ForCES PL) messages, it is composed of a “Header” and a “Body”. They are shown in figure 8.

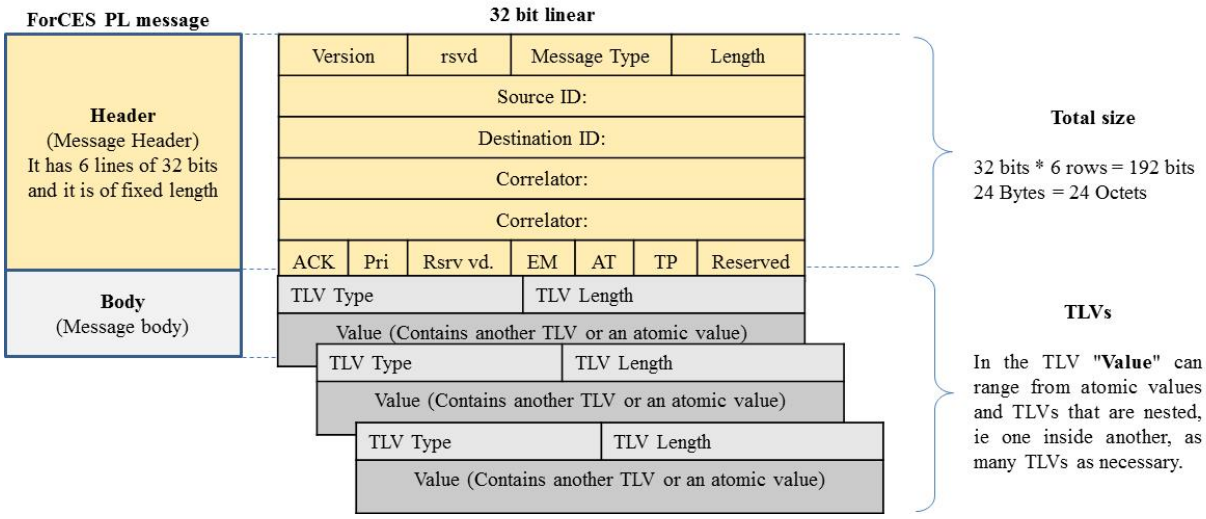

Figure 8. Encapsulating a ForCES message

4.4.1 Header

This it is the header of the message RFC 5810 (page 35), it is part of the composition of the Protocol Data Units (PDU) of ForCES protocol. Which it is used for the exchange between paired units, each of protocol messages. As it is shown in figure 9, it is used to control the complete behavior of the protocol in its functions of establishment and break of the connection, flow control, error control, etc.

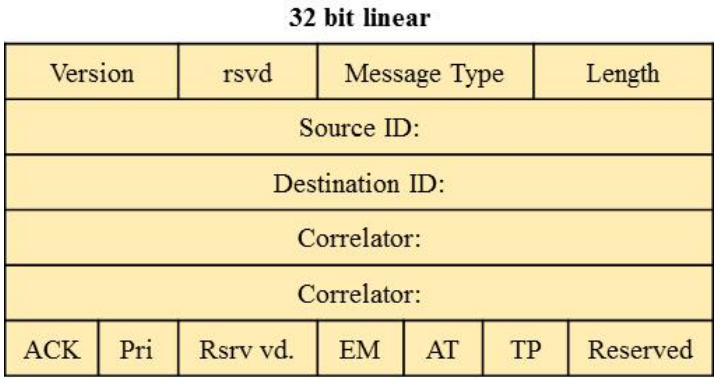

Figure 9. Header of the message

The header of the message contains six 32-bit lines, the size of this packet is of fixed length unlike the body that must be recalculated its length depending on the amount of data that is to be sent. Each of the bits in this header is modified according to specification RFC5810 (page 35). The length of the header is given in D-Words i.e. 4 bytes, as each 32-bit line is equal to 4 D-Words, this means that the length indicates the number of lines in the header of the message.

An example of how the bits are formatted it can be seen in the figure 10; in this case the header of the "Query" message was taken.

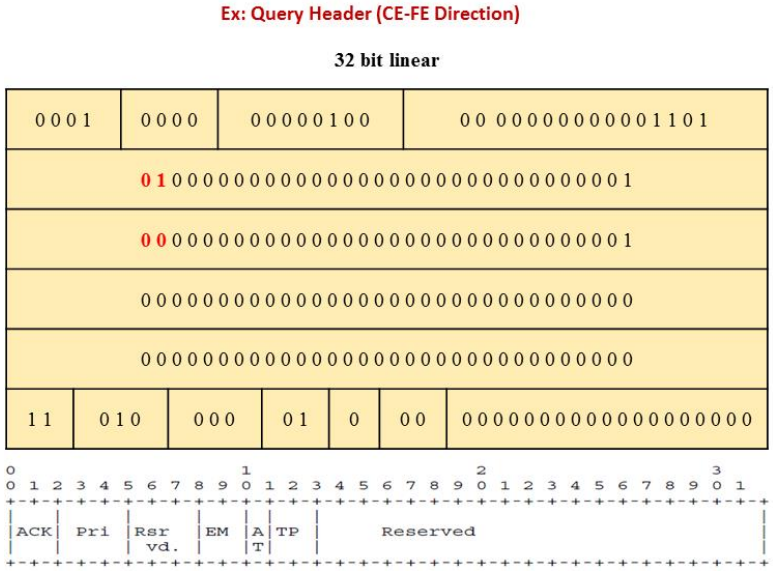

Figure 10. Query message header example

4.4.2 Body

Each message body is made up of one or more high-level TLVs. A high-level TLV may contain one or more sub-TLVs, these sub-TLVs are described in this article as an OPER-TLV, because they describe an operation to be performed. For the case of a config command the OPER-TLV = SET and for the Query command, the OPER-TLV = GET, as will be seen later when the Config and Query messages are constructed. The body of the message is shown in figure 11.

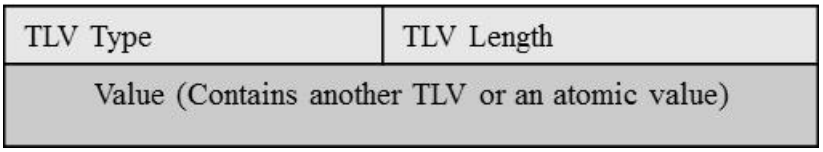

Figure 11. Body of the message

4.4.2.1 Structure of a TLV

TLVs (Longitude and Type Values) are information blocks that carry values including the type of information it contains and the length of the information, which for this article is of variable length. Some of its advantages are the coding of a sequence of data of easy interpretation, also can be added tags to the messages without making the message is incompatible, allows to nest structures TLVs one inside another so that the data of a TLV can be in turn another TLV. It is also easy to know if the message has arrived completely without the need for an extra charge. Nested TLVs are shown in figure 12.

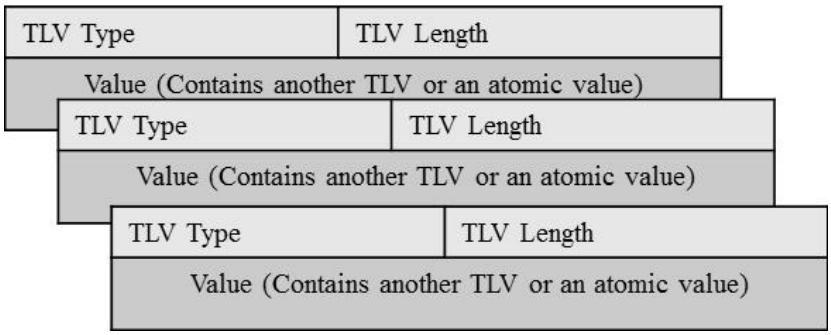

Figure 12. Nested TLVs

For the construction of PL ForCES messages, different types of TLVs are nested among them, including the OPER-TLV, FULLDATA-TLV, PATH-DATA-TLV.

There are different types of TLVs, this is the LFBselect which is a Basic LFB for the functionality of the ForCES protocol and has, in turn, a LFBCLASSID and an LFBInstance to address a particular LFB and within it has another TLV of the type OPERATION, which says whether it is a GET or a SET, plus all association IDs and all operation types of ForCES messages that are generally composed in this way RFC5810 (page 44).

Figure 13 shows several LFBs entities.

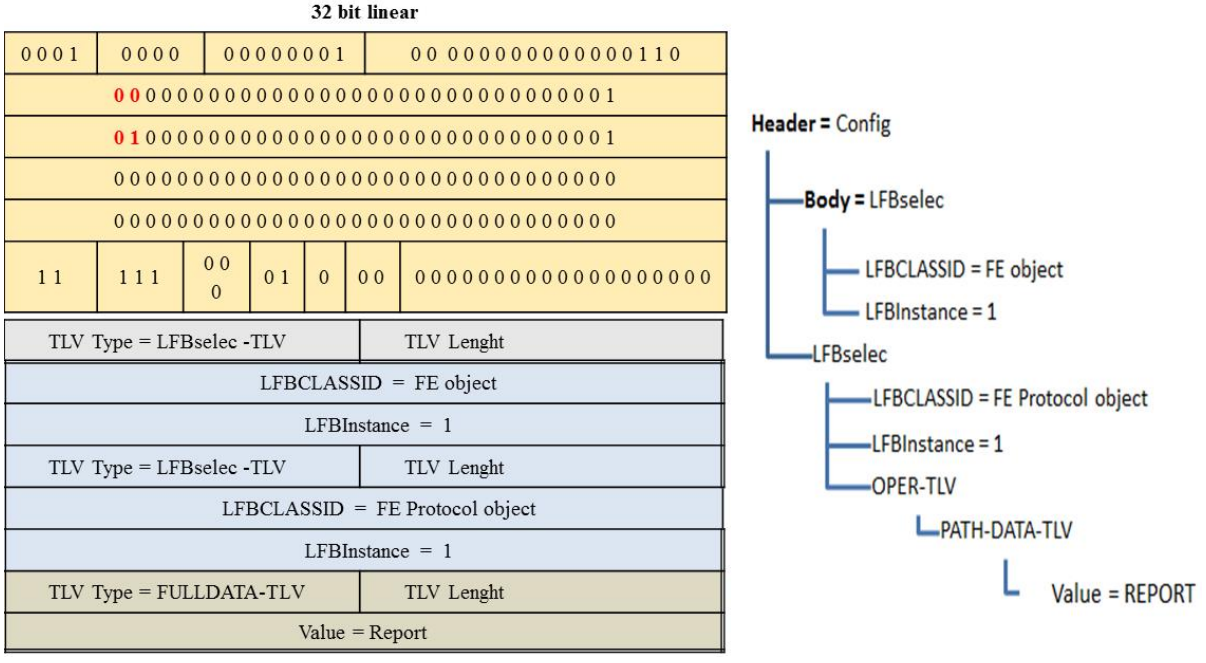

Figure 13. Addressing LFBs entities

There is a table which indicates which are each of the LFBs that each message uses described in the RFC5810 (page 45) and also the definition of how an OPER-TLV described in RFC5810 is configured (page 46), in which the first line of the OPER-TLV is the PATH, which it is where it is pointing. This PATH is defined by a FLAGS, IDCount, ID and then comes a field for the DATA or OPER-TLV information.

4.5 The TML

The Transport Mapping Layer (TML) [17] is responsible for transporting the messages that are encapsulated in the PL and corresponding to the ForCES protocol, once the message has been constructed, the PL passes this message to the TML, using a transport protocol (Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Stream Control Transmission Protocol (SCTP), Datagram Congestion Control Protocol (DCCP)) on any of the subjacent interconnection technologies (Ethernet, Backplane, Asynchronous Transfer Mode (ATM)), of which the international agency ForCES have selected the transport protocol of messages SCTP as it is stated in RFC5810 and RFC5811 and as interconnection technology the Ethernet protocol. This article it is showing its implementation on network cards located in different PCs chassis.

The Application Programming Interface (API) of the SCTP protocol [17], it is implemented through the operating systems: Linux or Solaris, because it is only found in these two. This work was developed in Ubuntu version 9.10, using the Java JDK 7, but since it is had not yet released a full version that included this API, it was necessary to use a Snapshot Release, called build b83 of February 12, 2010.

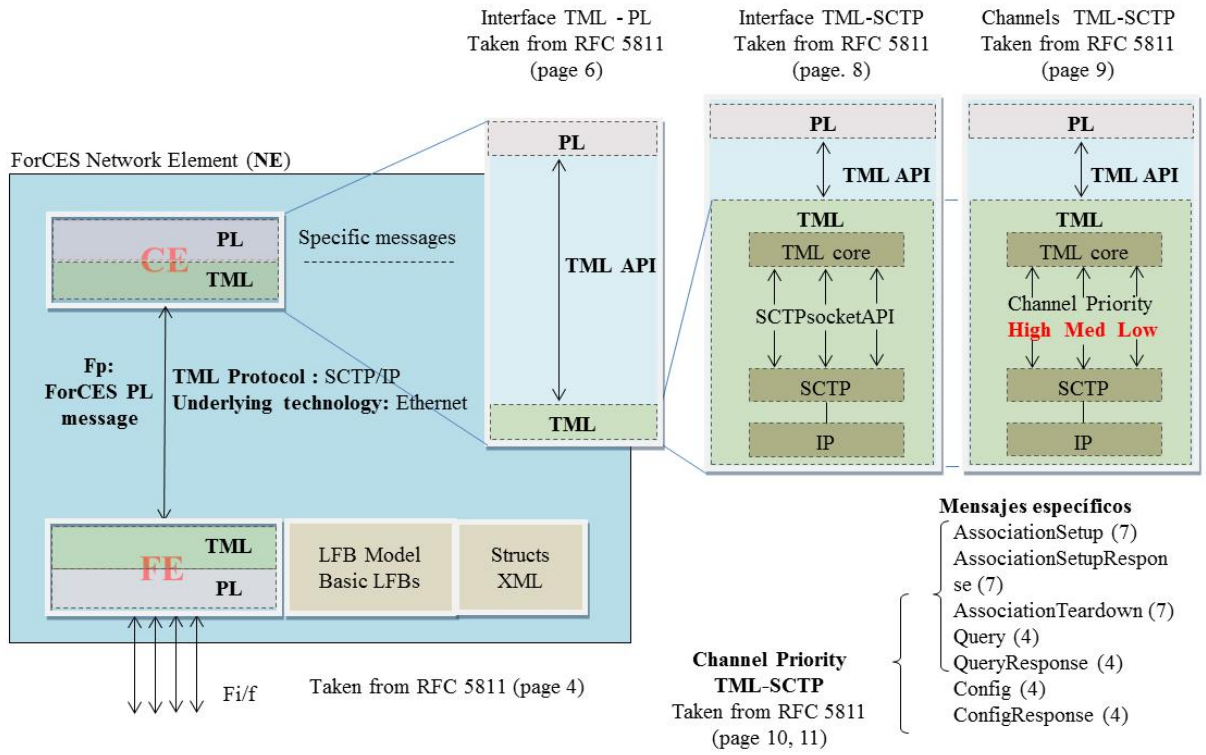

Figure 14. The TML

As it is shown in figure 14, both elements CE and FE are composed of two layers, the PL and TML. The TML communicates with the PL through an interface that transfers the already encapsulated messages for the TML to carry them. The TML internally contains an SCTP API [16] that is responsible for establishing the data connection between the CE and the FE. In the CE, the application is stays listening within three threads, each corresponding to the three

ports required by the specification and by which they are sent the PL messages depending on the priority.

In the program that was developed, we used methods to serialize these messages and send them as datagrams once the connection is established. These datagrams contain in their header a field in line six, which is dedicated to the flags of the message indicating the type of priority that allows deciding by which channel the information travels. The establishment of each of these channels is assigned to the port number according to its priority: port 6704 assigned to high priority messages, 6705 medium priority and 6706 low priority.

The order of establishment according to the protocol ForCES is in second place after the pre-association is carried out which has the information of identification and IP address of the CEs and FEs that are wished to connect. The program recognizes this information after executing the "TMLSetup" command that performs the aforementioned process.

4.6 SCTP Protocol

SCTP Protocol [16] is an end-to-end transport protocol that is equivalent to TCP, UDP, or DCCP in many respects. SCTP has a little of each of these protocols, it is oriented to the connection and the exchange of datagrams. Therefore, in the implementation of a client-server model of this application, the server is the CE element and the client is the FE element, which uses a socket application.

The establishment of the channel was achieved using the SCTP API over Linux OS, which is why this implementation was developed on this platform.

5. Programming the client-server application

In this section, it is shown how it's programmed the application client-server using platform Java, some of the code fragments that it shows reflect the internal logical algorithm necessary for its functioning.

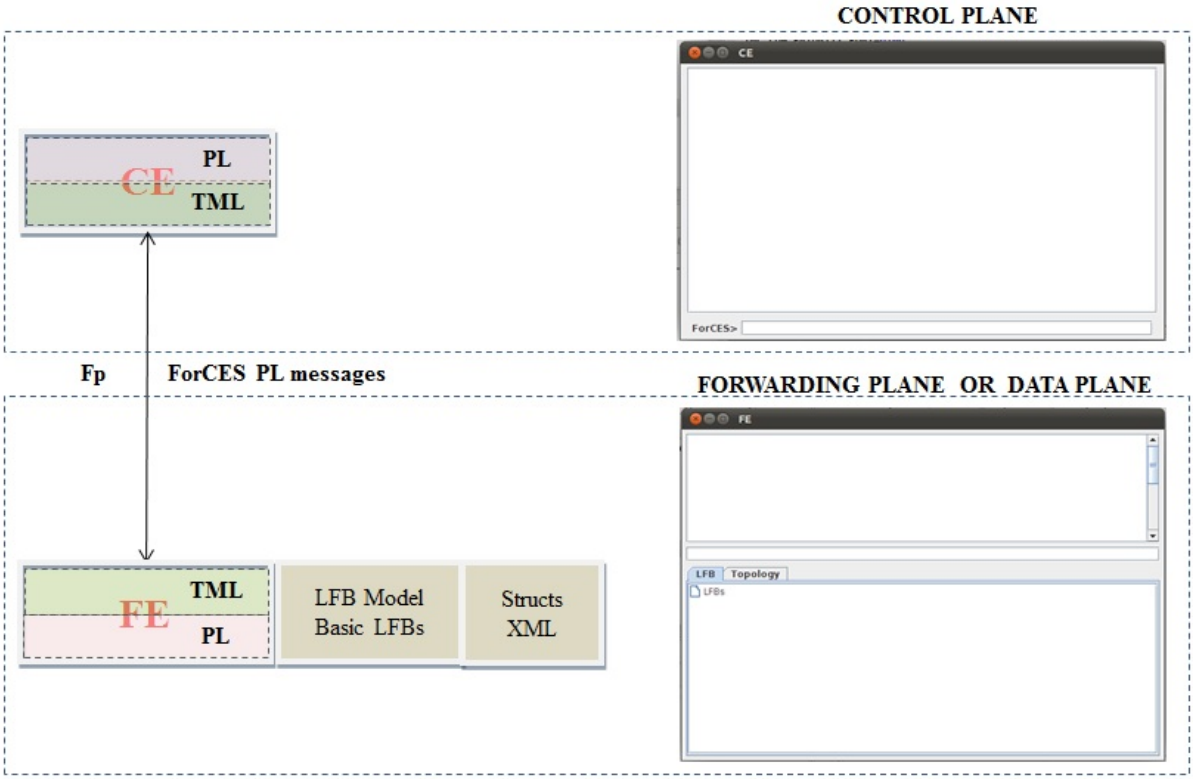

Figure 15. Application client-server

In figure 15, it shows the consoles correspondents to each CE and FE elements, each one with one field for write the commands and sent the ForCES PL messages and in the FE with two additional windows: one for seen the LFBs through of the file explorer and the another for configuring its topology.

More information about this implementation, it is found at [18] and [19].

```
public void handleTopology() {
    mxGraph mxGraph = new mxGraph();
    mxGraph.getStylesheet().getDefaultVertexStyle().put(mxConstants.STYLE_ROUNDED, true);
    mxGraph.getStylesheet().getDefaultVertexStyle().put(mxConstants.STYLE_WHITE_SPACE, "wrap");
    fe.updateTopology(mxGraph);
    mxOrganicLayout organicLayout = new mxOrganicLayout(mxGraph);
    organicLayout.execute(mxGraph.getDefaultParent());
    mxGraphComponent graphComponent = new mxGraphComponent(mxGraph);
    System.out.println("--- handleTopology ---");
    topologyPanel.setViewportView(graphComponent);

}
```

Figure 16. Handle topology method

```
public mxGraph updateTopology(LFBInstance lfbObjectLFBInstance, mxGraph mxGraph) {
    ComponentInstance lfbTopology = LFBHandler.getComponentByID(1, lfbObjectLFBInstance.getComponents());
    ArrayInstance arrayLFBLinks = (ArrayInstance)lfbTopology.getValue();
    List<Object> lfbLinks = arrayLFBLinks.getItems().subList(1, arrayLFBLinks.getItems().size());
    for (Object lfbLinkObject : lfbLinks) {
        StructInstance lfbLink = (StructInstance)lfbLinkObject;

        ComponentInstance fromLFBID = LFBHandler.getComponentByID(1, lfbLink.getComponents());
        StructInstance fromLFBSelector = (StructInstance)fromLFBID.getValue();
        String fromLFBClassID = Binary.parseLong(getLFBClassID(fromLFBSelector))+"";
        String fromLFBInstanceID = Binary.parseLong(getLFBInstanceID(fromLFBSelector))+"";
        AtomicInstance fromPortIndexAtomic = (AtomicInstance)LFBHandler.getComponentByID(3, lfbLink.getComponents()).getValue();
        String fromPortIndex = Binary.parseLong((String)fromPortIndexAtomic.getValue())+"";

        ComponentInstance toLFBID = LFBHandler.getComponentByID(4, lfbLink.getComponents());
        StructInstance toLFBSelector = (StructInstance)toLFBID.getValue();
        String toLFBClassID = Binary.parseLong(getLFBClassID(toLFBSelector))+"";
        String toLFBInstanceID = Binary.parseLong(getLFBInstanceID(toLFBSelector))+"";
        AtomicInstance toPortIndexAtomic = (AtomicInstance)LFBHandler.getComponentByID(6, lfbLink.getComponents()).getValue();
        String toPortIndex = Binary.parseLong((String)toPortIndexAtomic.getValue())+"";

        if (fromLFBClassID == null || fromLFBInstanceID == null || fromPortIndex == null) {
            System.out.println("Inconsistencia en origen");
            continue;
        }
        if (toLFBClassID == null || toLFBInstanceID == null || toPortIndex == null) {
            System.out.println("Inconsistencia en detino");
            continue;
        }
        if (!fromPortIndex.equals(toPortIndex)) {
            System.out.println("Inconsistencia en Puertos");
            continue;
        }
        String fromLFBName = getLFBName(fromLFBClassID+","+fromLFBInstanceID);
        String toLFBName = getLFBName(toLFBClassID+","+toLFBInstanceID);
        addTopologyLink(fromLFBName+":"+fromLFBClassID+","+fromLFBInstanceID, fromPortIndex, toLFBName+":"+toLFBClassID+","+toLFBInstanceID
    }
    return mxGraph;
}
```

Figure 17. Handle topology method

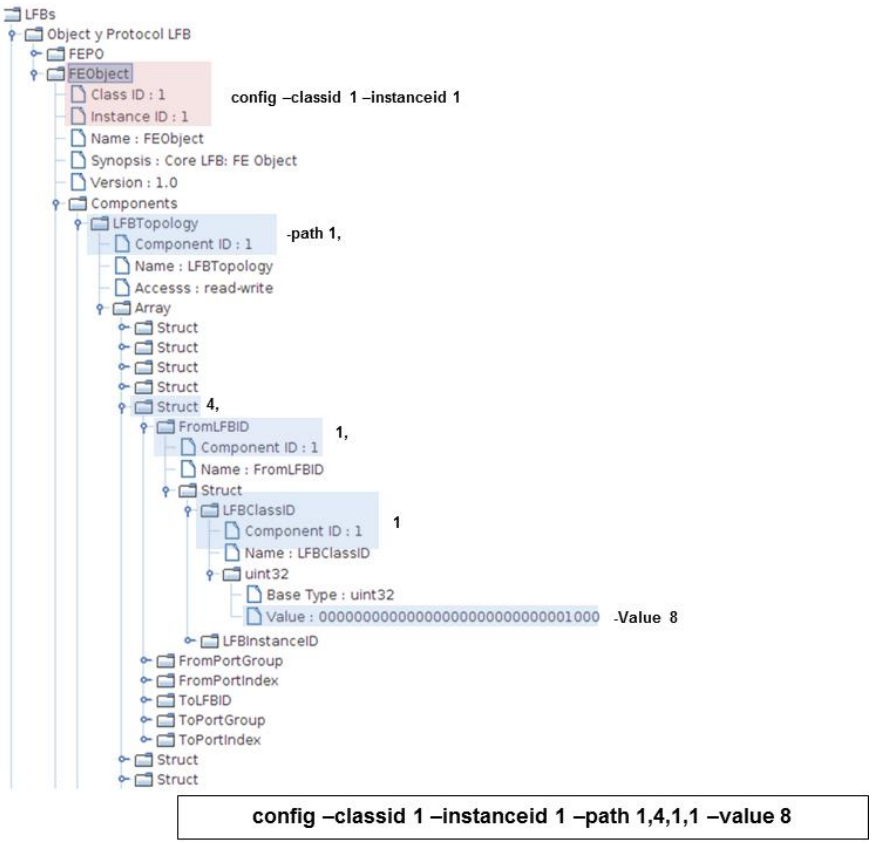

Figure 18. Example of command line

6. Interoperability tests

In this section, it is shown several scenarios with different uses of the commands, the configuration of LFBs through topology organization, loads of LFBs through the file explorer, loads of examples XML, where each scenario has a message different and where it is sent according to the sequence diagram of the ForCES PL messages.

6.1 Test scenarios

The draft-ietf-forces-interoperability-04 [15] contains the test scenarios necessary to evaluate the protocol's performance, these scenarios consist of the connection establishment and the ForCES messages of the PL layer that are exchanged between the CE element and the element FE. This draft allows checking the interoperability between these two elements by doing the tests one by one sequentially including an example scenario.

Each scenario is composed of one of the messages of the protocol, which is operated manually through the commands that are placed inside the consoles programmed in Java. To send the message one by one, the CE console is opened and the message is sent to the FE or from the FE console and the message is sent to the CE, depending on the direction of the message.

6.1.1 Scenario: Association Setup

The second message of the ForCES protocol used for the establishment of the connection between the CE and the FE called "Association setup" shows the way in which the message is sent through the application java (see figure 19).

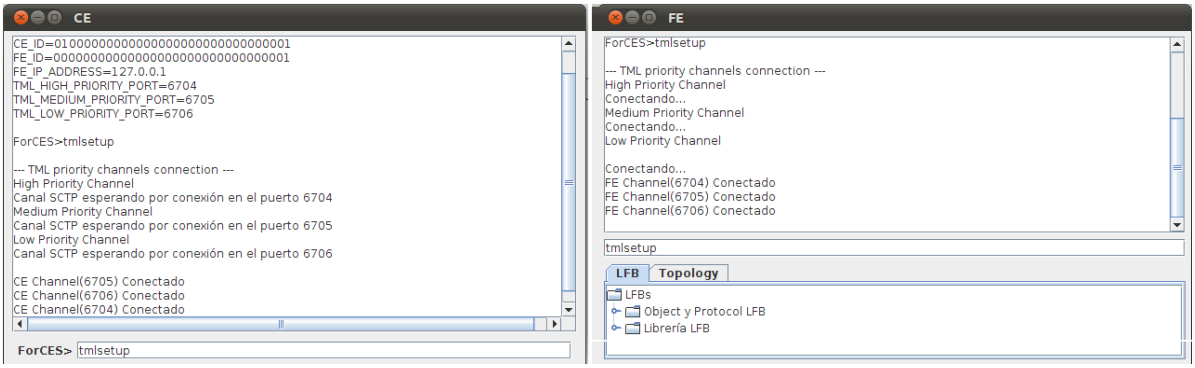

Figure 19. Association Setup message

The association is done by means of the "Association Setup" message that is sent from the FE to the CE only using the “associationsetup” command and the CE sends in response an "Association Setup Response" message, from this moment both CE and FE elements are associated.

In order for the ForCES protocol to work, each message must be sent in the sequence indicated by the RFC and as shown in the message exchange diagram.

6.1.2 Simple Config Command

The Config command consists of four elements: A ClassID, an InstanceID, a Path and a

Value, to send data from the CE to the FE. As shown in figure 20, a simple Config statement is sent that configures two LFBs to form a struct.

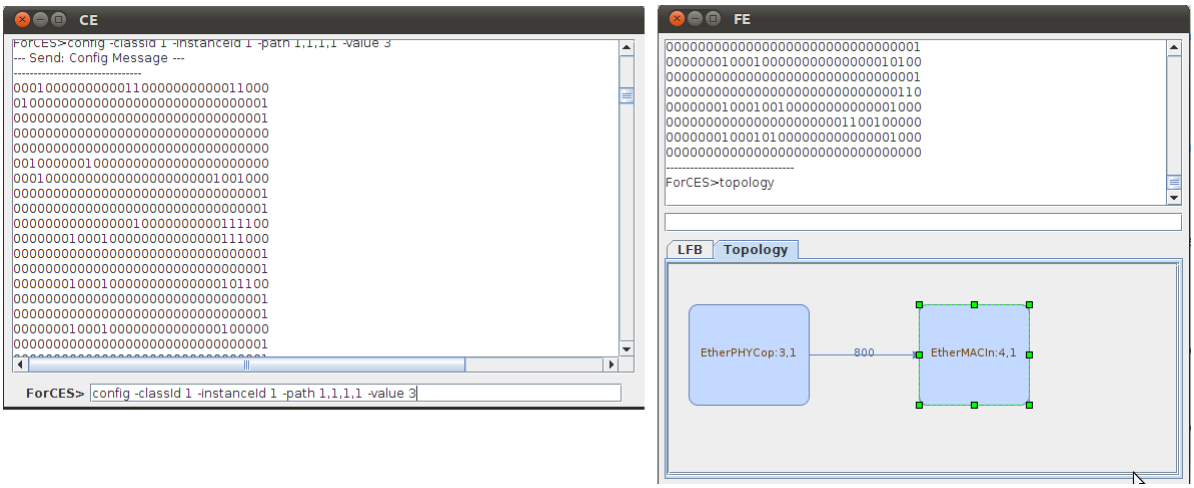

Figure 20. Simple Config Command

Once the values of the LFBs folder tree are mapped, the next step is to verify that the LFBs connection design has been done correctly by typing the Topology command in the topology tab of the FE console and in this way they observe how the configured LFBs are drawn.

6.1.3 Test example of the functionality IPv4 Forwarding

The draft-ietf-forces-lfb-lib-05 [20] containing libraries LFBs IPv4 Forwarding and ARP processing, and for a future proposal will be used the last draft, converted in the RFC6956 [21], which will be used to implement the following work [19]. For the case of this example, the LFBs IPv4 Forwarding library was used, following in the specification the figure on page 12 where the interconnection of all the functional logical blocks of the IP v4 Forwarding for an Ethernet port is shown graphically (see figure 21 and 22).

In the CE console, the statement **config -file "forces_script_IPv4Release.txt"** is used to send all necessary instructions with the information that creates the topology and fill the LFB tree of the CE.

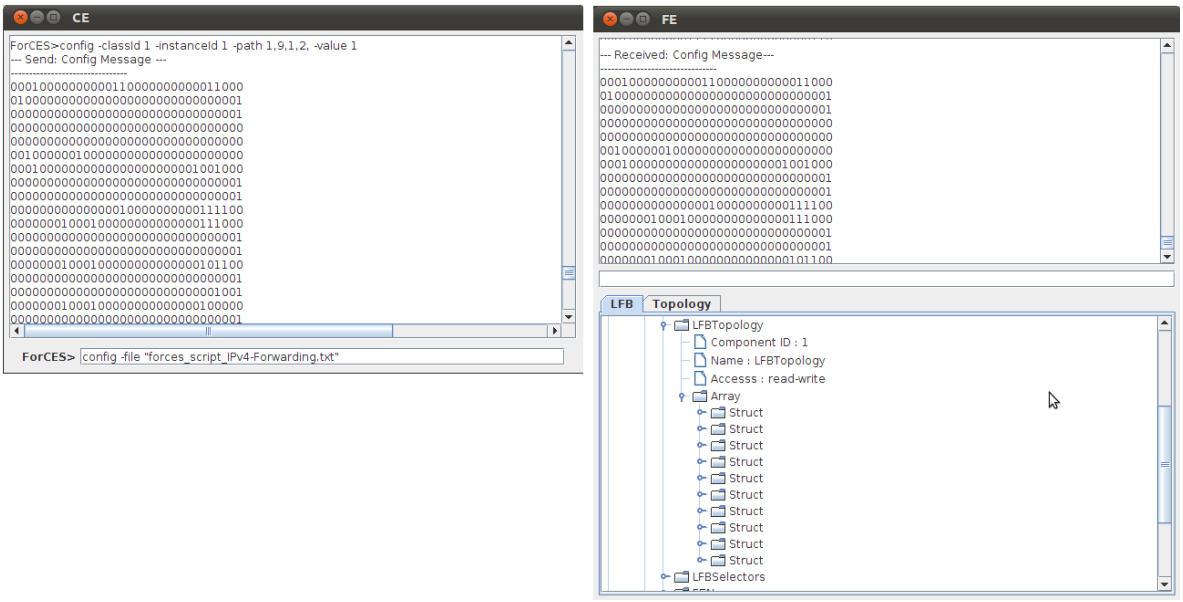

Figure 21. Configuring IPv4-Forwarding functionality

After the information in the LFBs folder tree is verified in the 9 structures required for this function, the topology tab is drawn to draw the connection of the LFBs involved (see figure 22).

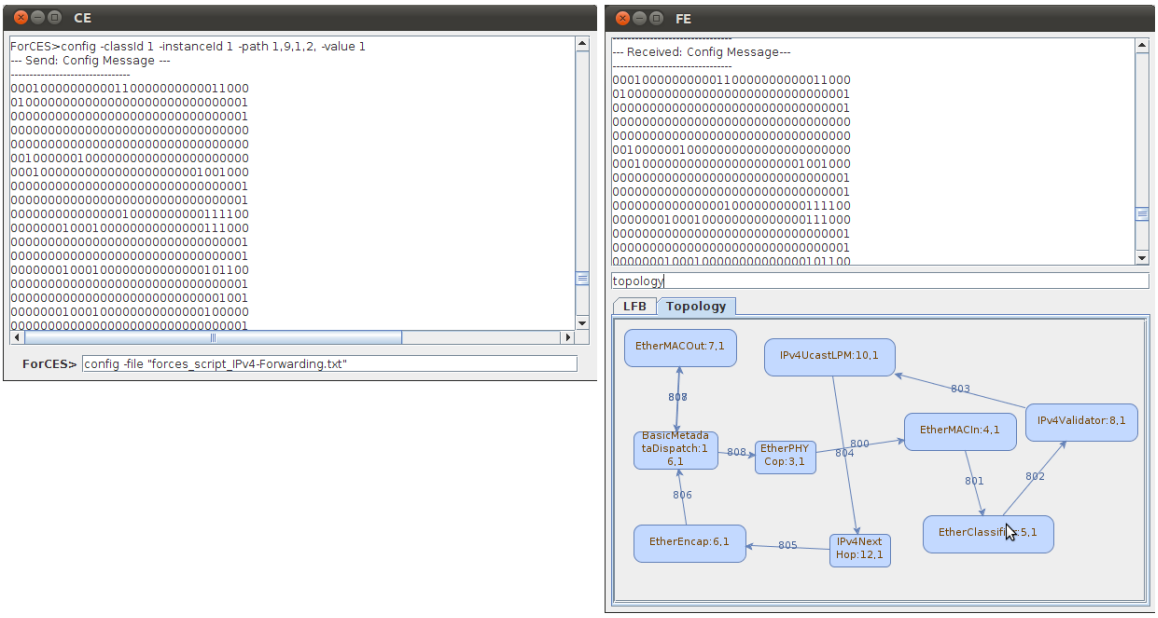

Figure 22. Topology of the LFBs involved

7. Testbed

In this section, it is proposing a test based on the previous scenarios, where each ForCES PL message is sent one per one and step to step with the target of verifying the packets format and its size. To make it, each packet it is captured with a protocol analyzer and then it is compared with the initial message.

On page 10 and 12 of the draft-ietf-forces-interoperability-04, it is shown the assembly of an experiment which consists of having an only CE element associated with several FEs elements through a Local Area Network (LAN), with a switch and a protocol analyzer. In this article, the testbed unlike the one proposed by the ForCES group consists of an only CE and an only FE associated with a LAN, with a switch and a protocol analyzer. In figure 23, a PC represents the CE and the other PC represents the FE, in the third PC it is found the protocol analyzer software called Wireshark, which is put to listen to the connection between the two PCs.

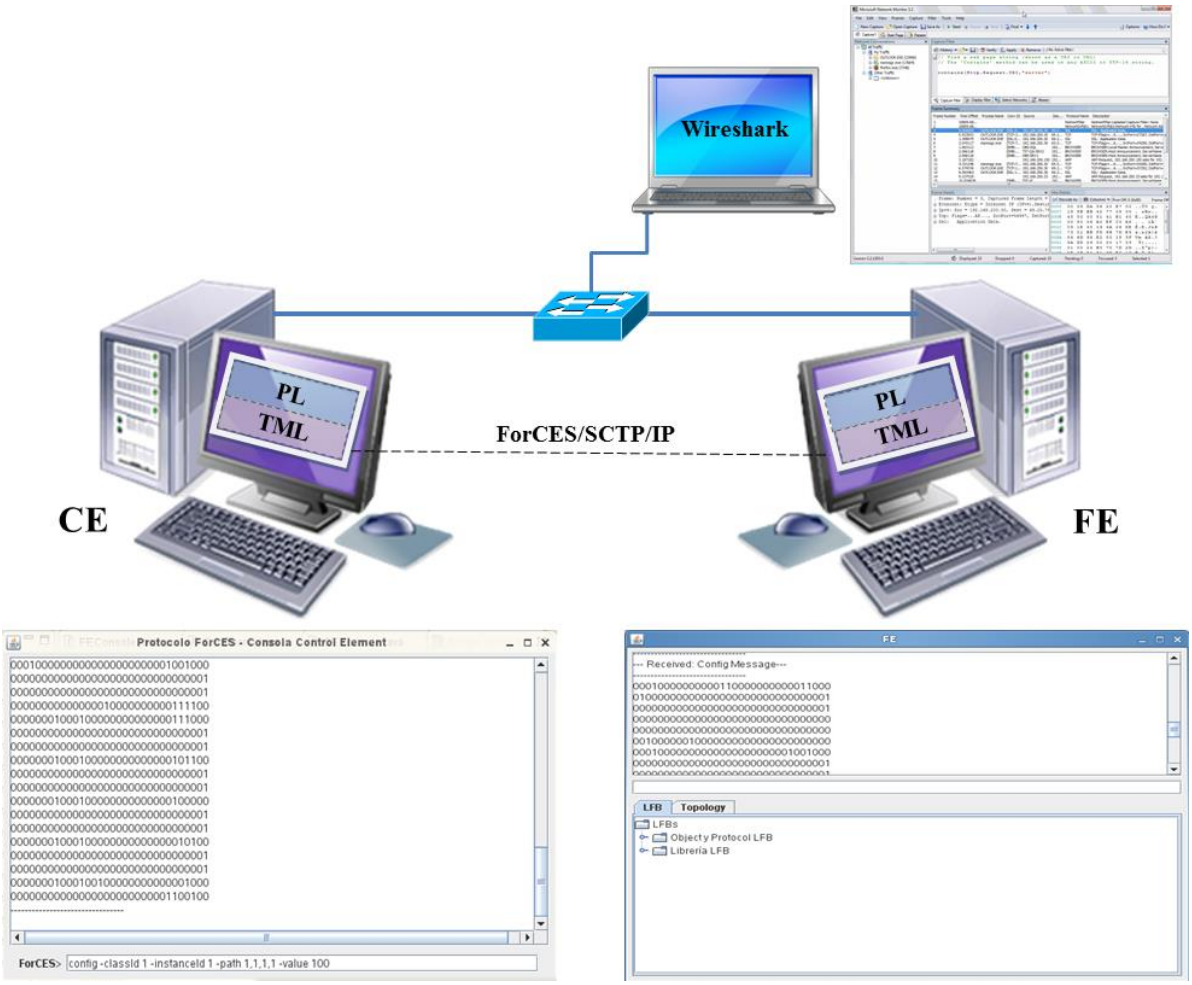

Figure 23. TestBed ForCES

For see the messages of these tests, we have carried out the capture with the protocol analyzer using the IP address of local host (127.0.0.1) on our own equipment. The Figure 24 shows the screen capture of the protocol analyzed during the test.

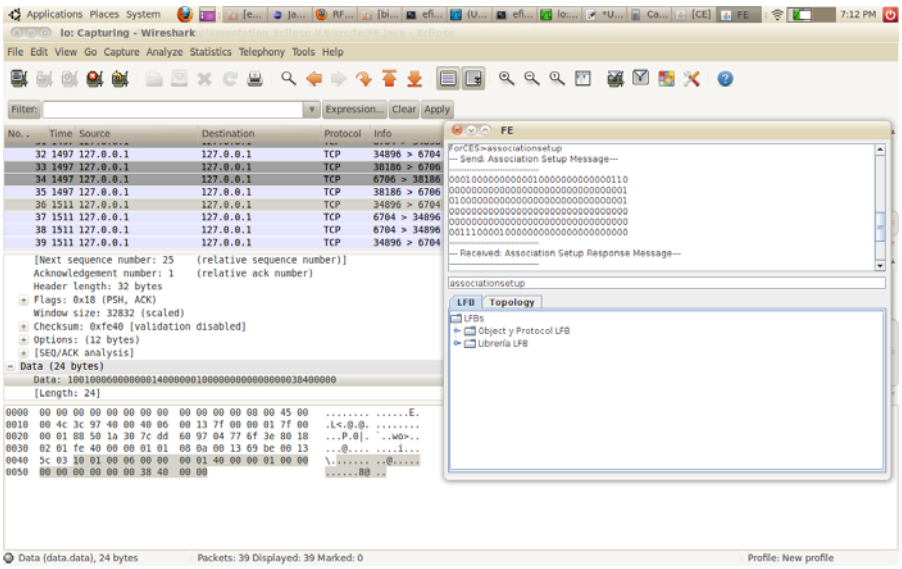

Figure 24. Captures with Wireshark of the “Association Setup” messages

In the previous graphs, we can see the captures of the message "config", where we compare what is captured with the protocol analyzer and the window CE from where the "config" command is sent, the capture is done under the encapsulation of the SCTCP protocol. The data that is compared are those that are sent through the protocol ForCES.

TCP message length given during our test is shown in figure 25. The maximum values are peaks of 134 Bytes, while the minimum value has been 54 Bytes, which belongs to a reset message. The average amount of Bytes transferred has been 80 Bytes.

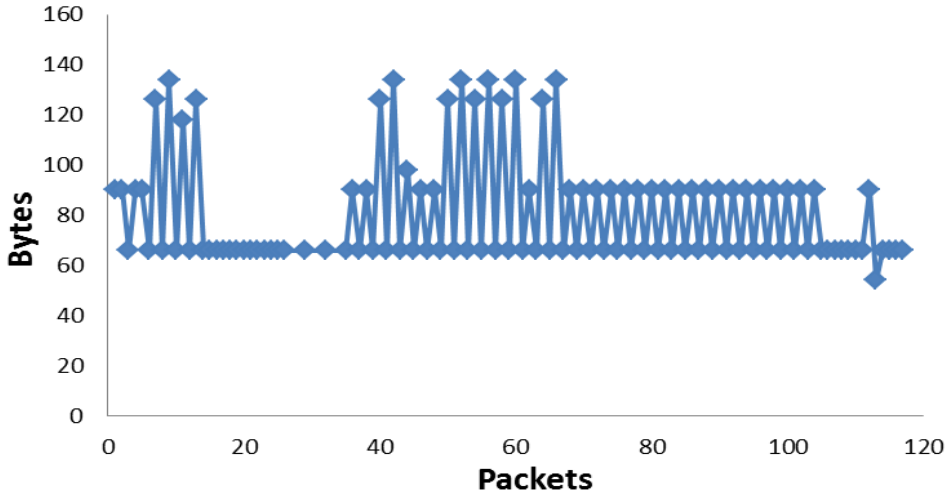

Figure 25. TCP message length along the communication.

Figure 26 shows a number of messages along the test bench. The time is measured in milliseconds. The maximum number of messages was at 2220 ms, with 16 messages.

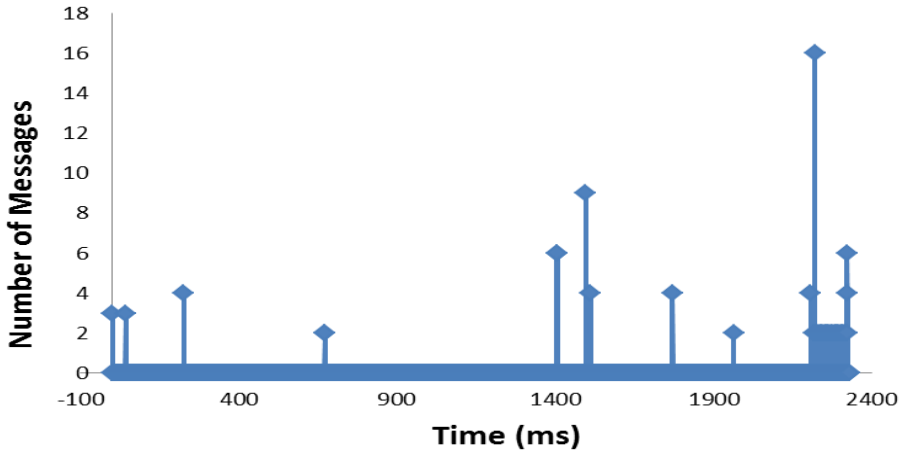

Figure 26. Number of messages transmitted along the time in our test bench.

8. Conclusion and future work

With the development of this article, the ForCES protocol can be used under the theoretical basis of the ForCES architecture, taking advantage of the open, flexible and reprogrammable architecture to perform simulation tests on the Java user interface and improvements to the open source java program.

The people who will benefit directly are those who belong to the ForCES development group and students and researchers in the academic world, who want to continue working and improving the functionality of the interface and protocol, as it provides a good Starting point for future developments in this area.

With the results obtained, it can provide support for future research; it will also provide the different research groups with a valuable tool, allowing them to work observing behaviors of different types of topologies and through them, construct different functionalities of the elements of network contained in a router with reprogrammable ForCES architecture.

This implementation allows proposing new LFB test topologies in the CEs that structure functions of a typical router, although it is still in the process of investigation and does not assure a clear management on the available attributes in the FEs.

Testing was accomplished by following the evaluation scheme proposed by the draft-ietf-forces-interoperability-04, through scenario-based commands and then capturing the data frame with the Wireshark protocol analyzer.

Making a protocol that conforms to the flexibility and reprogramming policy required by ForCES within its specification implies a high degree of complexity, which involves a considerable effort in the software development part, since it is necessary to interpret a lot of information from the RFCs and Draft that are not very clear when programming to maintain the ForCES standard.

For a future work, we will do more tests of interoperability with other scenarios in order

to see different network behavior and measure its performance, the operation time and the quality of service (QoS). We will also show new models of data based on different network topologies with different LFBs configurations.

References

[1] IETF ForCES Working Group. Forwarding and Control Element Separation (ForCES). Available at IETF website: https://datatracker.ietf.org/wg/ForCES/

[2] A. Doria, J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification," RFC 5810, March, 2010. Available at IETF website: https://tools.ietf.org/html/rfc5810. DOI: https://doi.org/10.17487/RFC5810

[3] W. Weiming, L. Dong, B. Zhuge, "Analysis and implementation of an open programmable router based on forwarding and control element separation," Journal of Computer Science and Technology, vol. 23, no. 5, pp. 769–779, Sep. 2008. DOI: https://doi.org/10.1007/s11390-008-9181-4

[4] M. Urueña Pascual and D. Larrabeiti López, "Contribución al diseño de arquitecturas distribuidas de nodos de red programables," Universidad Carlos III de Madrid, 2005. DOI: https://doi.org/10016/11415

[5] R.N. Takourout, S. Pierre, L. Marchand, "Separation of the control plane and forwarding plane in next-generation routers," Journal of Computer Science, 2006, vol. 2, no 11, p. 815-823. DOI: https://doi.org/10.3844/jcssp.2006.815.823

[6] W. Weiming, M. Gao, L. Dong, and J. Xi, "An Improved Algorithm for Forces LFBs Topology in J2EE", International Journal of Web Services Practices, Vol. 3, No.1-2 (2008), pp. 89-93. http://netcom.zjgsu.edu.cn/publications/IJWSP2008-Weiming%20Wang.pdf, http://openforces.zjsu.edu.cn.

[7] W. Weiming, M. Gao, L. Dong, and J. Xi, "A Technology for LFBs Topology Layout in ForCES Architecture", Next Generation Web Services Practices, International Conference on, vol. 00, pp. 131-134, 2007. DOI: https://doi.org/10.1109/NWESP.2007.9

[8] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka and T. Turletti, "A survey of software-defined networking: Past, present and future of programmable networks," IEEE Communications Surveys and Tutorials, vol. 16, no. 3, pp. 1617–1634, 2014. DOI: https://doi.org/10.1109/SURV.2014.012214.00180

[9] N. Feamster, J. Rexford and E. Zegura, "The road to SDN: an intellectual history of programmable networks," ACM SIGCOMM Computer Communication Review, vol. 44, no. 2, pp. 87–98, Apr. 2014. DOI: https://doi.org/10.1145/2602204.2602219

[10] J. M. Jimenez, O. Romero, A. Rego, A. Dilendra, J. Lloret, Study of multimedia delivery over software defined networks, Network Protocols and Algorithms 7 (4), 37-62. 2016. https://doi.org/10.5296/npa.v7i4.8794

[11] D. Medhi, K. Ramasamy, "Network Routing: Algorithms, Protocols and Architectures", Elsevier/Morgan Kaufmann Publishers, San Francisco, 2007. Pages 788. ISBN: 9780080474977. Available at Elsevier books website: https://www.elsevier.com/books/network-routing/medhi/978-0-12-088588-6

[12] S. Sendra, P.A. Fernández, M.A. Quilez, J. Lloret, Study and performance of interior gateway IP routing protocols, Network Protocols and Algorithms 2 (4), 88-117. 2011. doi: https://doi.org/10.5296/npa.v2i4.54
[13] N.A. Alrajeh, S. Khan, J. Lloret, J. Loo, Secure routing protocol using cross-layer design and energy harvesting in wireless sensor networks, International Journal of Distributed Sensor Networks 9 (1), 374796. 2013. doi: https://doi.org/10.1155/2013/374796
[14] J. Hadi Salim, J. Halpern, "Forwarding and Control Element Separation (ForCES) Element Model Forwarding," RFC 5812, March, 2010. Available at IETF website: https://tools.ietf.org/html/rfc5812. DOI: https://doi.org/10.17487/RFC5812
[15] E. Haleplidis, K. Ogawa, X. Wang, C. Li, "ForCES Interoperability Draft," draft-ietf-forces-interoperability-04, September 7, 2009. Available at IETF website: https://tools.ietf.org/html/draft-ietf-forces-interoperability-04
[16] W. Wang, E. Haleplidis, K. Ogawa, C. Li, J. Halpern, "ForCES Logical Function Block (LFB) Library," RFC6956, June 10, 2013. Available at IETF website: https://tools.ietf.org/html/rfc6956. DOI: https://doi.org/10.17487/RFC6956
[17] J. Hadi Salim, K. Ogawa, "SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol," RFC 5811, March, 2010. Available at IETF website: https://tools.ietf.org/html/rfc5811. DOI: https://doi.org/10.17487/RFC5811
[18] P. L. González Ramírez, “Diseño e Implementación del Protocolo ForCES,” Pontificia Universidad Javeriana, Master’s Thesis. 2012. DOI: https://doi.org/10554/20769
[19] S. C. Martínez Cordero, “Diseño e implementación de un prototipo de la entidad elemento de control de la arquitectura ForCES,” Pontificia Universidad Javeriana, Master’s Thesis. 2012. DOI: https://doi.org/10554/12716
[20] W. Wang, E. Haleplidis, K. Ogawa, C. Li, and J. Halpern, “ForCES Logical Function Block (LFB) Library,” draft-ietf-forces-lfb-lib-05. 2013. Available at IETF website: https://tools.ietf.org/html/draft-ietf-forces-lfb-lib-05
[21] Wang, W., Haleplidis, E., Ogawa, K., Li, C., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Logical Function Block (LFB) Library", RFC 6956, June 2013. DOI: https://doi.org/10.17487/RFC6956.

Copyright Disclaimer