

Design and implementation of a prototype of the entity Control Element (CE) of the Architecture ForCES

Susan Martínez Cordero

Dept. of Electronics, Pontificia Universidad Javeriana

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

E-mail: susan.martinez@javeriana.edu.co

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

Universidad Politécnica de Valencia

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

E-mail: jlloret@ocom.upv.es

Luis Carlos Trujillo Arboleda

Dept. of Electronics, Pontificia Universidad Javeriana

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

E-mail: trujillo.luis@javeriana.edu.co

Received: October 1, 2017 Accepted: December 20, 2017 Published: December 31, 2017

DOI: 10.5296/npa.v9i3-4.12433

URL: <http://dx.doi.org/10.5296/npa.v9i3-4.12433>

Abstract

This paper presents the designed an implementation of a prototype with the Forwarding and Control Element Separation (ForCES) Architecture. That is to say, which allows each of the elements to be improved separately and then interconnected through the ForCES protocol, even in remote locations. The Control Element (CE) is the logical entity that is part of the control plane and is responsible for managing Forwarding Elements (FE) of the data plane. The ForCES architecture allows you to see these two elements (the CE and the FE through the ForCES protocol) as a single Network Element (NE), even if they are located in remote sites each. To demonstrate this principle, a network testbed scenario was implemented, based on two Local Area Networks (LAN). The LAN 1, for the CEs and the LAN 2 for the FEs, once communicated through the ForCES protocol, the different LFBs configurations of the ARP, SNMP, RIP protocols were used to demonstrate their operation.

Keywords: Forwarding and Control Element Separation (ForCES) Architecture, ForCES protocol, programmable networks and systems, flexible architectures, open architecture, logical functional block, testbed ForCES, software defined networking.

1. Introduction

With the increase in traffic demand, according to a report made by Cisco Systems [1], and with the provision of new voice, data and video services, among others, data networks, both private and internet, require equipment that supports this demand, that are reliable in the transmission of information, and that present a scalable architecture and greater processing capacity. Currently the teams that provide these services are proprietary, have a closed architecture, a high cost, and are not scalable; therefore, if the network requires a new service and the equipment does not support it, it is necessary to change the device to an updated one, leading to a change of devices every time an improvement is made in the network, increasing costs and creating a dependency with the manufacturers.

To make an improvement in the devices and services provided in the communications networks, the standardization bodies have been interested in the creation of interfaces and network elements with an open architecture, programmable, scalable and at a low cost. The ForCES architecture, created and standardized by the Internet Engineering Task Force (IETF) network working group ForCES [2], allows the development of a network element based on the physical separation of the elements of the control plane and data plane [3]. The Control Element is designed under the regulations and structure given in the documents established by the EITF for ForCES [4], both architecture and the handling of the protocol used for communication between the CE and FE. To carry out tests, a basic Forwarding Element was implemented, composed internally of logical entities called Logical Functional Blocks (LFBs), which are characterized by some classes and instances, each LFB class [5], consists of one or several components, which are operational parameters that should be visible to the CEs, these can be flagged, arguments of unique or complex parameters and tables that the CE can read and / or write through the ForCES protocol [6].

The prototype of CE is composed of six main modules: ForCES Module, IP Module, Virtual Interface Module, Routing Module, Management Module and High Availability Module.

The ForCES module is responsible for establishing the connection between the CE-FE and allowing the Control Element (CE) functional manipulation, management and configuration of the Forwarding Elements (FE) through the LFB and also allows the interaction and management of the control elements (CE), when an CE fails in the network element, this functional block triggers another CE as backup.

The IP module manages the addressing of the physical interfaces of the CE. Manage refers to configuring, consulting, enabling and disabling the interfaces that are part of FE0.

The Routing module allows us to configure the RIP routing protocol and consult the routing tables.

The Management module, through the Net-SNMP, is responsible for displaying some parameters that allow monitoring the network and the router.

The Virtual Interface module, Virtual Server Interface Module (MIVS) is an application that allows to reflect the interfaces that are in the FE as if they were in the CE and in this way to be able to configure the RIP routing protocol from the CE and then transport that information through virtual tunnels to the physical interfaces in the FE and then to the outside.

In the FE there are the LFBs and a module Virtual Client Interface Module (MIVC), this module allows a replication of the physical interfaces that the FE has in logical interfaces to reflect them towards the MIVS that is in the CE, so that the CE sees these remote interfaces as its own.

The High Availability module is responsible for maintaining operability in the NE, in case the main CE fails, the FE enters the pre-association status and starts the search for another CE (backup) to associate again and continue operating.

This paper shows the operation of a prototype of the CE entity in a real environment of a simple data network, making use of the ForCES protocol [7], [8]. This allows to check the functioning of the ForCES Architecture in the conditions established by the specifications defined and created by the Internet Engineering Task Force (IETF) through the work group Forwarding and Control Element Separation (ForCES).

The rest of this paper is organized as follows. In Section 2, we discuss existing related works. Section 3 shows the router architecture ForCES and its functions, and then in the Section 4 the design of the prototype based on the RFCs and the contributions of the authors. The implementation is explained in Section 5. In the Section 6 shows the Configuration and Tests. Finally, Section 7 shows the conclusion and future works.

2. Related works

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

The ForCES architecture has been widely accepted by several academic research groups such as the University of Zhejiang Gongshang, the Institute of Communications Networks and Engineering in China, Ben Gurion University in Israel, and Stanford University among others, as can be seen in the documents [4], [9], [10], [11], [12], [13], [14]. This because of that the components of a network element (NE), control elements (CE) in the plane of control and elements of forwarding (FE) in the data plane, are can it studied separately and individually (improving its functions and services), to then integrate it and to form a single flexible and scalable device.

Hagsand et al. [15] Initially, it performs an analysis in the decentralized modular design, which would improve the scalability, flexibility and reliability of the future routers, then shows the design and implementation of a distributed router, based on the physical separation of the different functional modules of the operation planes: control and data (ForCES architecture) and communication between them by a protocol designed with the support of NetLink messages.

For the development of this work, the use of the ForCES protocol designed and implemented by the author was fundamental [7], which was used and modified to give continuity to the current proposal. The use of this application, already developed, allowed to advance and continue to implement the other modules of the ForCES architecture, and reach a closer development of the proposal elaborated by the ForCES working group.

In [16] the design and implementation of a control element (CE) is shown, through an exclusive and independent platform for the resources of the control plane, which can significantly improve its scalability, control capacity and efficiency. Also, a discussion is presented on the hierarchical structure of the reconfigurable network, the software architecture of the control element, its main components, the middleware of the protocol and the experiments. The results of the experiment show the feasibility of the design of the control element.

Some of these works have followed the ForCES architecture and its ForCES protocol, but others use this principle of similar way to reformulate new solutions, such as the Software Defined Networks (SDN), which has become a new way to make dynamic topologies. Tarnaras et al. [17] and Haleplidis et al. [18] they propose through SDN an alternative of programmable networks based on the separation of the planes of control and forwarding. Providing a dynamic and optimized way, capable of dealing with network traffic in constant growth. He also talks about OpenFlow's approach to creating the topology map by exchanging frames called the Link Layer Discovery Protocol (LLDP) between the CE and the FE. It is including provides a comprehensive tutorial on ForCES by summarizing numerous

standards documents and thus making the technology easily understood by the wider research community. Examples of the latter include the use of ForCES for network function virtualization (NFV) proofs-of-concepts. This work also surveys recent independent interoperable implementations that showcase the full spectrum of ForCES applications in the era of NFV and SDN. Recent investigations are focused on using SDN for multimedia delivery [19].

3. ForCES Architecture

In this section, it is shown how the ForCES architecture works.

A Network Element (NE), be it a router or a switch, is basically composed of numerous separate logical entities that interact with each other to provide a certain functionality such as, for example, routing. These entities are known as Control Elements (CE), which are part of the Control Plane and Forwarding Elements (FE) in the Data Plane, which are interconnected and communicated with each other by means of a Communication protocol, however, to external entities, appear as an integrated network element. ForCES provides a standard set of mechanisms for the connection of these components offering greater scalability and allowing control and forward plans to evolve independently, thus promoting rapid innovation [20].

The ForCES architecture is composed of one or more control elements (CE), one or more forwarding element (FE) and a communication protocol that allows interaction between these elements, as shown in Figure 1.

The CE is responsible for operations such as signaling and implementation of administration and routing protocols. Based on the information acquired through the information processing of the basic functions, the EC determines the behavior of the packets transmitted by the FE, using the interconnection protocol. For example, the CE can control an FE by manipulating the forwarding tables and the state of its interfaces. The FEs operates on the plane of forwarding and are responsible for the processing and handling of the packages. Some functions that FEs develops include forwarding, firewall, Network Address Translation (NAT), encapsulation, de-encapsulation, and encryption, among others.

To the ForCES protocol [7], once the message has been constructed, this message is send 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 [6] and as interconnection technology the Ethernet protocol. This article it is showing its implementation on network cards located in different PCs. In Figure 1, the ForCES architecture is observed.

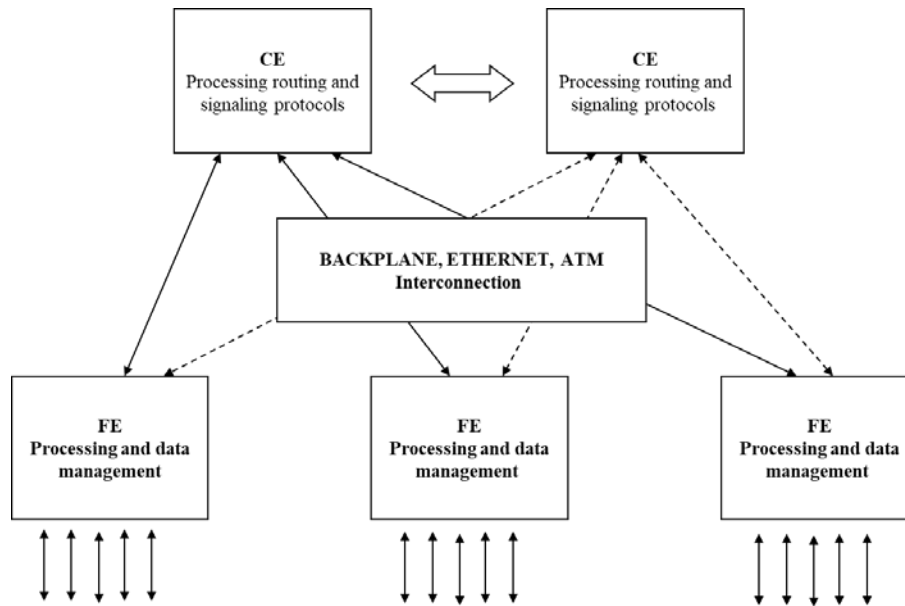


Figure 1. ForCES architecture. (based on RFC3654)

A fundamental part that is part of the FE and that is controlled by the CE and is specifically of the ForCES protocol, are the Functional Logical Blocks (LFBs).

According to RFC6956 [5], a Functional Logical Block is a block that resides in a forwarding element (FE) and is controlled by the control element (CE) operated on the ForCES protocol. These blocks are categorized by the LFBs classes, an LFB Instance, represents an existing class and each class is represented by an ID. Communication between different LFBs is done through metadata, which are defined in the RFC model of a FE (RFC5812) as a redirect packet that communicates the status of a packet from one LFB to another LFB. The metadata is sent between FEs and CEs, but is not sent through the network. An LFB class is made up of one or several components, which are operational parameters that must be visible to the CEs, these can be the flags, arguments of unique or complex parameters and tables that the CE can read and / or write through of the ForCES protocol.

The representation of how the instances of the LFBs are logically interconnected and placed along the data path in a FE is known as the LFB topology.

In Figure 2, it can be seen that the LFBs have inputs, outputs and components that can be required and manipulated by the CE by means of the reference point (Fp) defined in RFC3746 and the termination point (Ft) of the ForCES protocol. Internally in the FE are the LFBs, which are interconnected with the inputs and outputs of each of them, the P input, indicates the data packet and the M input indicates the metadata associated with a packet. The Fp point between the CE and the FE establishes a bidirectional communication. The communication from the CE to the FE is for configuration, control and packet entry, while the communication from the FE to the CE is to redirect packets to the control plane, monitoring information report, information counting and error reporting. The result of the interaction by the CE is the manipulation of the components of the instances of the LFBs.

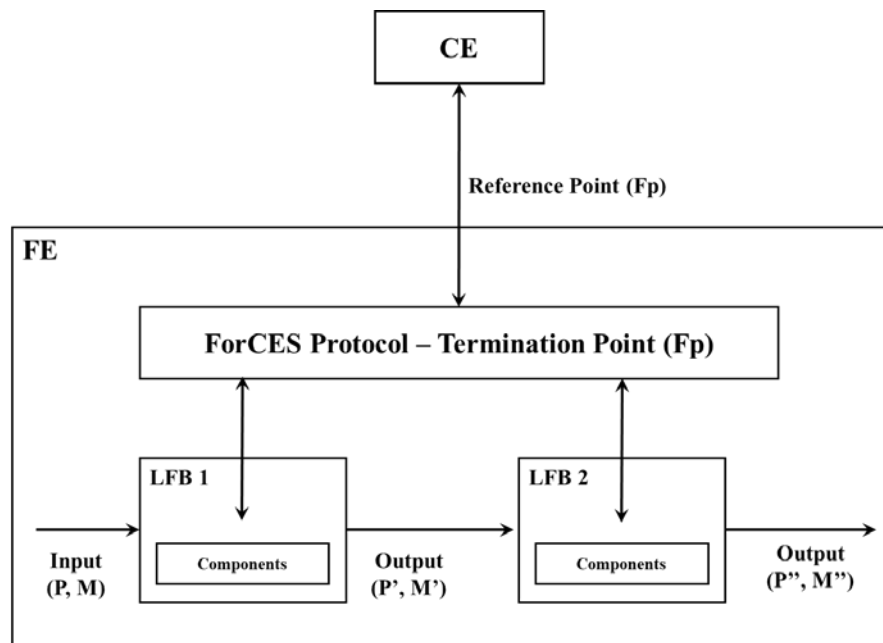


Figure 2. Diagram of an LFB (Taken from RFC5812, page 16)

4. Design of the prototype

In this section is present the analysis of the RFCs and general design of the prototype [21].

4.1 Control Element (CE)

The control plane is the logical part of a Network Element (NE). Internally the control plane is formed by control elements called CEs, based on software, which control and manage the operation of a router.

This development is implemented in the Linux platform with Operating System Ubuntu and the Graphical User Interface (GUI) was implemented in Eclipse, a Java-based development platform.

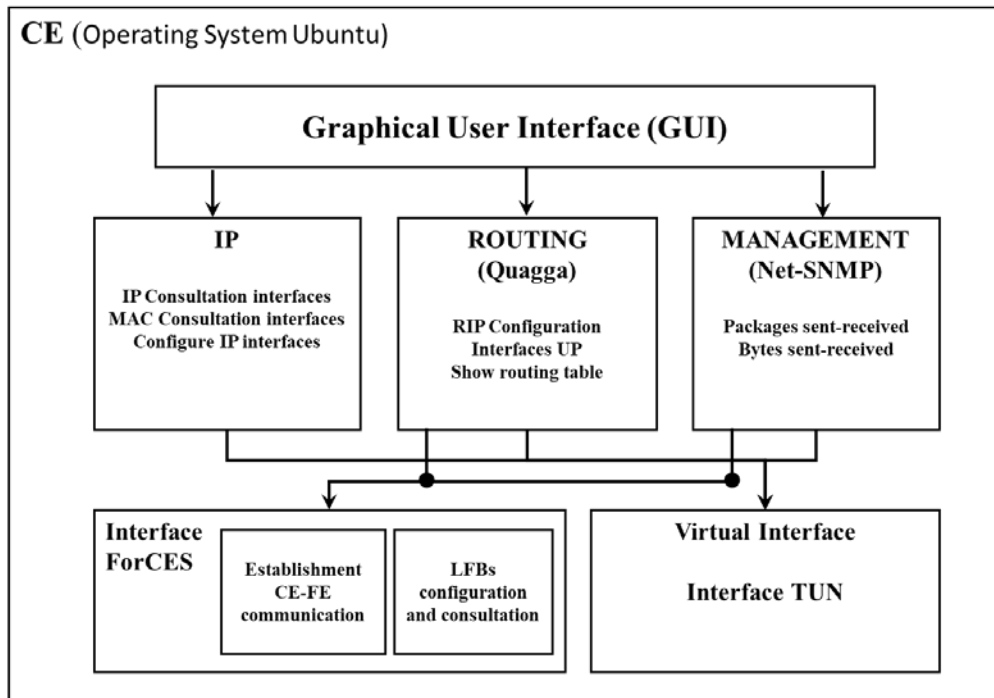


Figure 3. Structure of the Control Element

Figure 3 shows the block diagram of the Control Element designed and the different modules that are part of it. Next, each module of the prototype is explained.

4.1.1 ForCES Module

The ForCES module is responsible for allowing the interconnection and communication between the CE and FE, by means of the ForCES protocol. Initially performs the Pre-Association Phase, in which there is a discovery of the components and capabilities of each element. In each FE there are the entities known as Functional Logical Blocks or LFBs, explained above, each LFB Class has a series of components and each component has capabilities, that information is what the CE learns from the FE. Once this discovery is made, the Post-Association Phase enters into which the communication between the CE and FE is established through the ForCES protocol, which can be seen in detail [7].

Once the communication is established, the CE can consult the associated FE or FEs, the components and the capacities of each component, the query is done through the Query message, where the FE responds to the CE with the message Query Response.

The CE also has the ability to configure some components of the FE LFBs and establish topologies with the union of several LFBs to fulfill a specific function, for example: routing, ARP, SNMP and Forwarding, by means of the ForCES configuration message, Config.

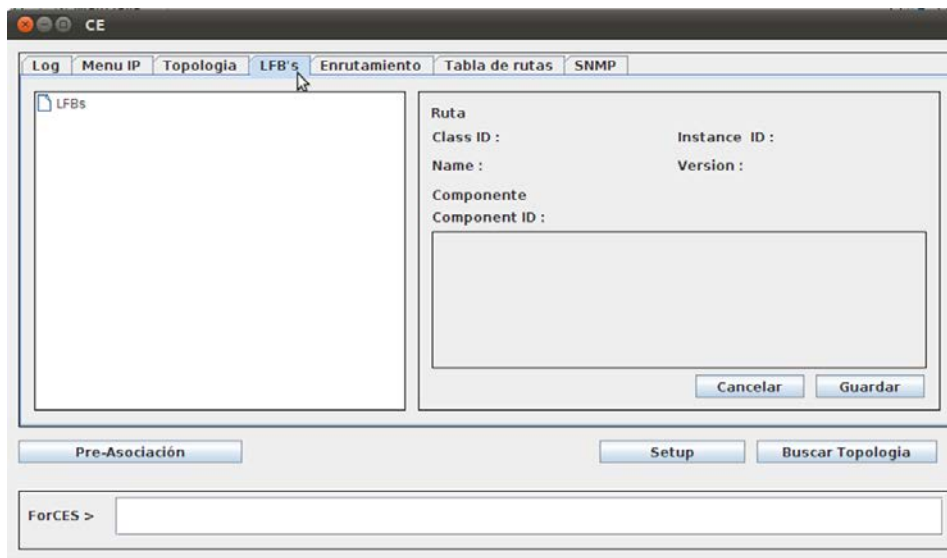


Figure 4. Graphical User Interface (GUI) of the ForCES Module in the CE

In Figure 4, the graphic interface of the ForCES Module in the CE is observed. When making the association, the CE consults the different parameters of the LFBs that make up the FE, among which are ClassID, InstanceID and ComponentID.

4.1.2 IP Module

It makes a communication between the CE and FE, which allows to know the physical interfaces, logical addresses and physical addresses that the FE has and are consulted by the CE, and in turn, the CE is in the ability to have managed over those interfaces as if they were local, although they are really remote.

To establish communication with this module, a protocol called Background is created and it uses the message with the same name. This protocol allows the CE to know the name of the FE interfaces, the physical address and the logical address of each interface. It also implements a Java interface called serializable, which adds Java own methods so that the class can be transportable and it is possible to send the messages through the network interface and the data arrives intact.

The columns, virtual port and virtually that are part of the virtualization module that will be explained in the next section, are in the IP menu because they are closely related to each FE interface captured in this module. In Figure 5, the graphic interface of this module is observed.

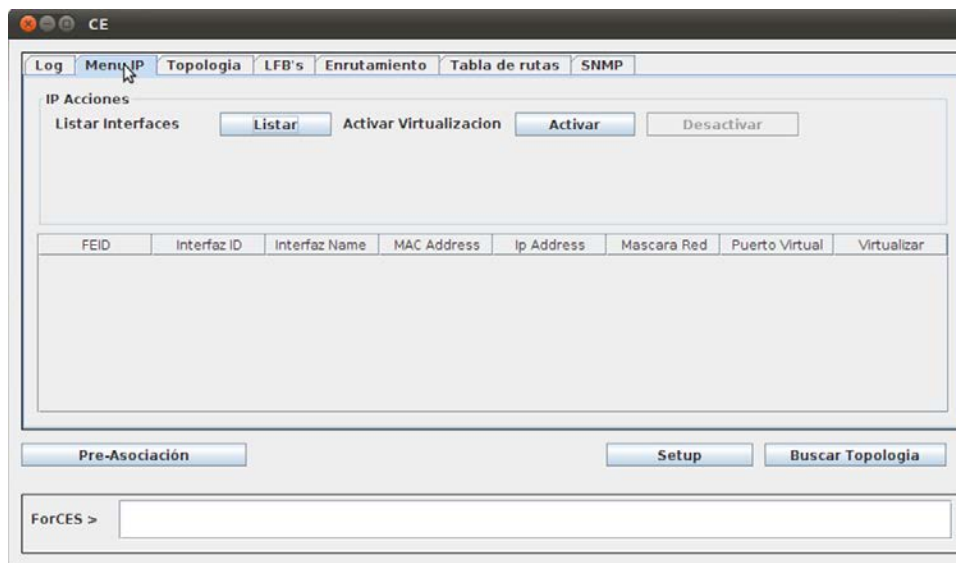


Figure 5. Graphical Interface of the IP Module

4.1.3 Virtual Interface Module

The Virtual Interface module uses the Linux application called Etherpuppet, which creates virtual interfaces (TUN/TAP) from one machine to another through the Ethernet interface through TCP ports.

This application allows everything seen by the real interfaces to be seen also in the virtual interfaces and what is sent by the virtual interfaces, be sent by the real interfaces, allowing to reflect the interfaces that are in the FE as if they were local interfaces of the CE and in this way, the CE can configure the IP addresses of the interfaces, enable/disable them and also configure the RIP routing protocol and then transport that information through virtual tunnels to the physical interfaces in the FE and then outward.

The source code of the Etherpuppet application can be downloaded from the page www.secdev.org/projects/etherpuppet/, the file is called **etherpuppet.c v0.3**.

The Figure 6 shows the graphical interface for the virtualization of the interfaces.

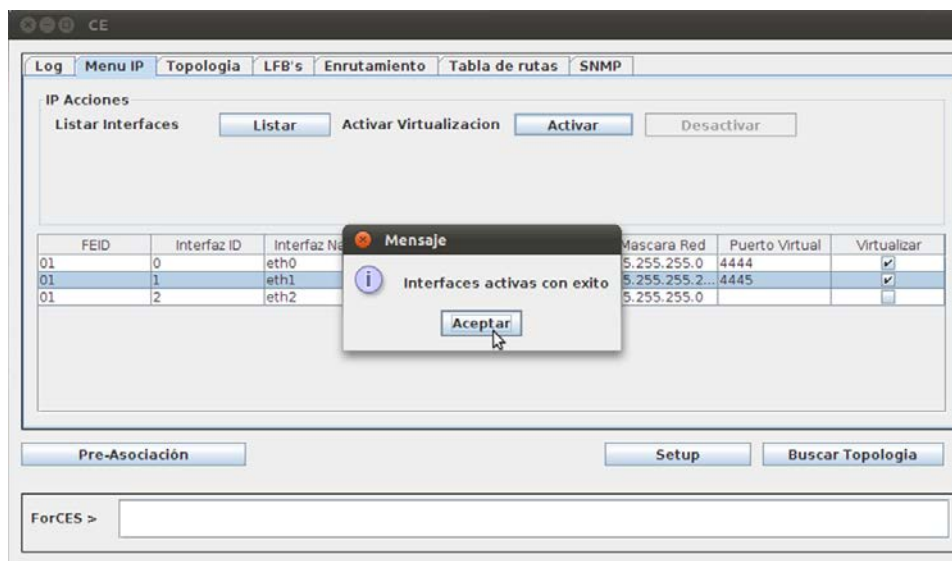


Figure 6. Virtualization of the interfaces

4.1.4 Routing Module

This module uses the **Quagga** application, which is a routing software package for TCP/IP networks that provides implementations of RIPv1, RIPv2, OSPFv2, OSPFv3 [22] and BGP for Unix platforms, including Linux. **Quagga** is a derivation of the Zebra GNU application. The core of **Quagga** is primarily the Zebra daemon, which acts as an abstraction layer to the Linux kernel and presents the Zserv API. There are Zserv clients, which implement a routing protocol and update routing tables for the Zebra daemon. These Zserv are:

- Ospf6d (Implements OSFPv2)
- Ripd (Implements Ripv1 and Ripv2)
- Ospf6d (Implements OSPFv3 / IPv6)
- Ripngd (Implements Ripng / IPv6)
- Bgpd (Implements BGPv4)

In Figure 7, the graphical interface of the routing module is observed.

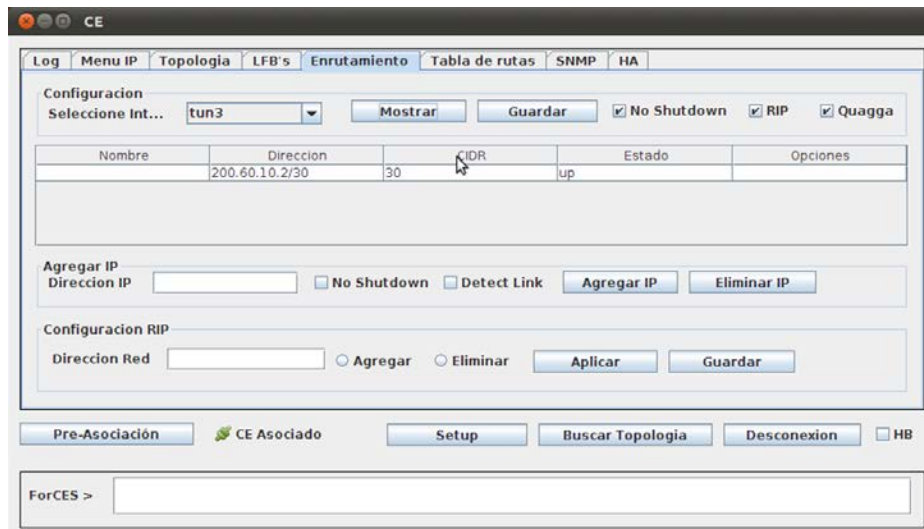


Figure 7. Routing Module

4.1.5 SNMP Module

For the development of this module, the Net-SNMP application was used [22], which is a set of applications used to implement the SNMP protocol.

The commands used are:

- Snmpget, snmpgetnext, snmpwalk, snmptable: Take information from managed devices.
- Snmpset: Manipulate information about device configuration.
- Snmptranslate: Translates numeric and textual OIDs of the objects of the MIB and visualizes the content and structure of the MIB.

To obtain the SNMP information in the interface table, the following commands were used in the SNMPInterface.java class:

- Snmpwalk -v2c localhost -c public 1.3.6.1.2.1.31.1.1.1.1
(list the interfaces)
- Snmpget -v2c localhost -c public 1.3.6.1.2.1.31.1.1.1.1.1
(list the name of the interface 1)
- Snmpget -v2c localhost -c public 1.3.6.1.2.1.31.1.1.1.1.2
(list the name of the interface 2)
- Snmpget -v2c localhost -c public 1.3.6.1.2.1.31.1.1.6.1
(bytes received by interface 1)
- Snmpget -v2c localhost -c public 1.3.6.1.2.1.31.1.1.10.1

(bytes transmitted through interface 1)

- `Snmptest -v2c localhost -c public 1.3.6.1.2.1.31.1.1.7.1`

(packets received by interface 1)

- `Snmptest -v2c localhost -c public 1.3.6.1.2.1.31.1.1.11.1`

(packets sent by interface 1)

In Figure 8, you can see the graphical interface of the SNMP module.

The screenshot shows a window titled 'CE' with a menu bar containing 'Log', 'Menu IP', 'Topología', 'LFB's', 'Enrutamiento', 'Tabla de rutas', and 'SNMP'. The main area displays a table with the following data:

Interfaz	Arbol	Bytes Enviados	Bytes Recibidos	Paquetes Enviados	Paquetes Recibidos
lo	iso.3.6.1.2.1.31.1.1....	261775	261775	3056	3056
eth0	iso.3.6.1.2.1.31.1.1....	545762	529454	4870	4189
wlan0	iso.3.6.1.2.1.31.1.1....	795990	3098537	6407	5282
tun3	iso.3.6.1.2.1.31.1.1....	29059	36265	410	311
tun4	iso.3.6.1.2.1.31.1.1....	7897	0	0	48

Below the table are buttons for 'Pre-Asociación', 'Setup', and 'Buscar Topología'. At the bottom, there is a text input field labeled 'ForCES >'.

Figure 8. SNMP Module Interface

4.1.6 High Availability Module

The ForCES protocol provides mechanisms for redundancy and failure of CEs, which is known as High Availability. The ForCES architecture allows FEs to take into account multiple CEs, but forces only one CE to be the master controller. This is known in the industry as 1 + N redundancy. The master CE controls the FEs by means of the ForCES protocol operating in the Fp interface. If the master CE fails, the backup CE assumes the operation of the NE.

The high availability parameterization in the FE is activated by configuring the LFB FE Protocol Object. The Heartbeat Interval FE, CE Heartbeat Dead Interval (CEHDI), and CE Heartbeat Policy help in the detection of connectivity problems between a FE and an EC. The CE Failover Policy defines the reaction on a detected fault.

The CE table of the FE Object Protocol LFB version 2 contains all the CEIDs that the FE can connect and associate as backupCEs.

The order of the CE IDs in the table defines the order of priority in which the FE will connect to the CEs. In the pre-association phase, the first CE ID (the lowest index in the table) in the table of the CEs, must be the first CE ID that the FE will take into account to connect and associate. If the connection and association of the FE with that first CE ID fails, it will try

to connect to the second CE ID and so on, and the cycle will return to the beginning of the list until there is a connection and association with it.

The FE must be associated with at least one CE. Following a successful association, the FEPO's CEID component identifies the associated CE master.

To avoid conflicts, the FE must respond to the messages of the master CE only, for example, the FE must ignore the messages that come from the backup CE. However, asynchronous events and heartbeats are sent to the associated CEs. The Heartbeat interval, the CEHB Policy and the FEHB Policy must be the same for all CEs.

Figure 9 shows a block diagram that facilitates the recovery of the connection with high availability. Once the FE has been associated with the master CE it moves to the post-association phase (Association Status). In this state, the master CE can update the list of CE backups. It is assumed that the master CE will communicate with the other CEs within the NE for the purpose of synchronization through the CE-CE interface, but this part is beyond the scope of the project.

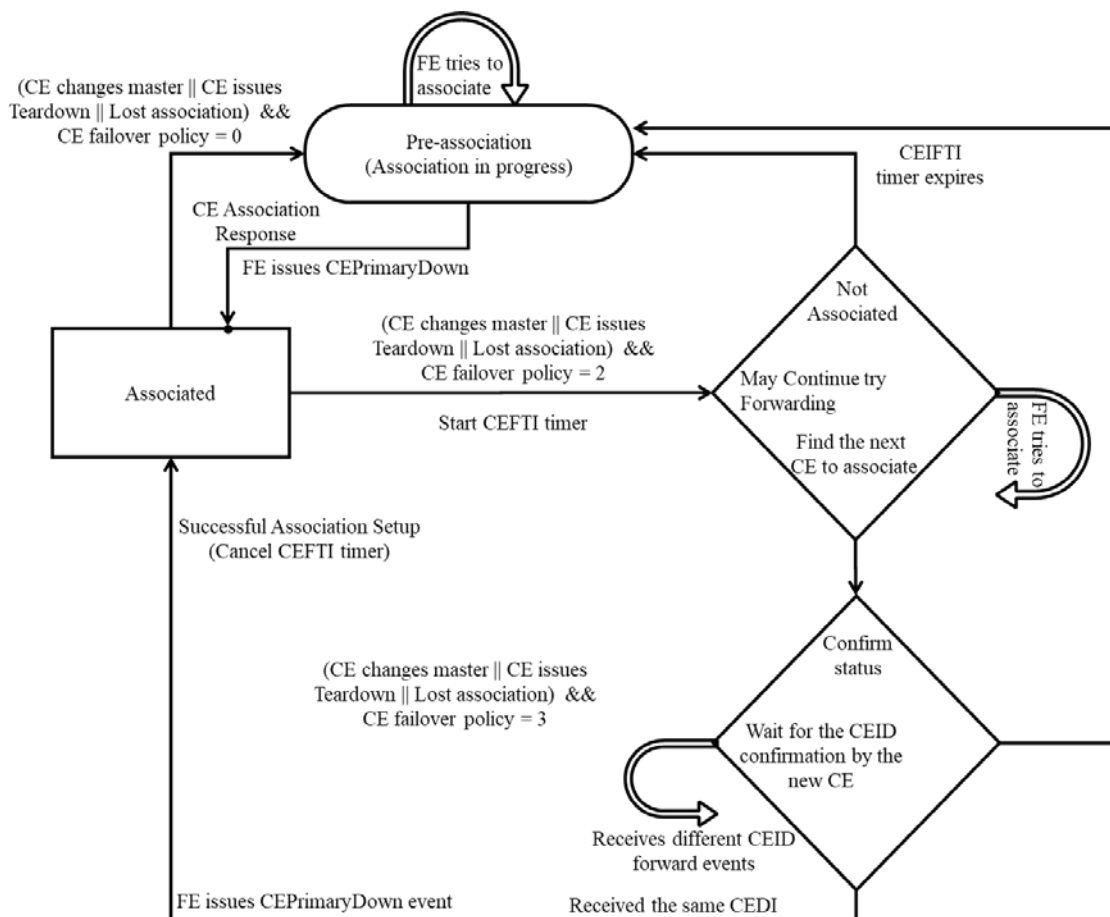


Figure 9. Block diagram of the CE considering High Availability (HA), taken from [4]

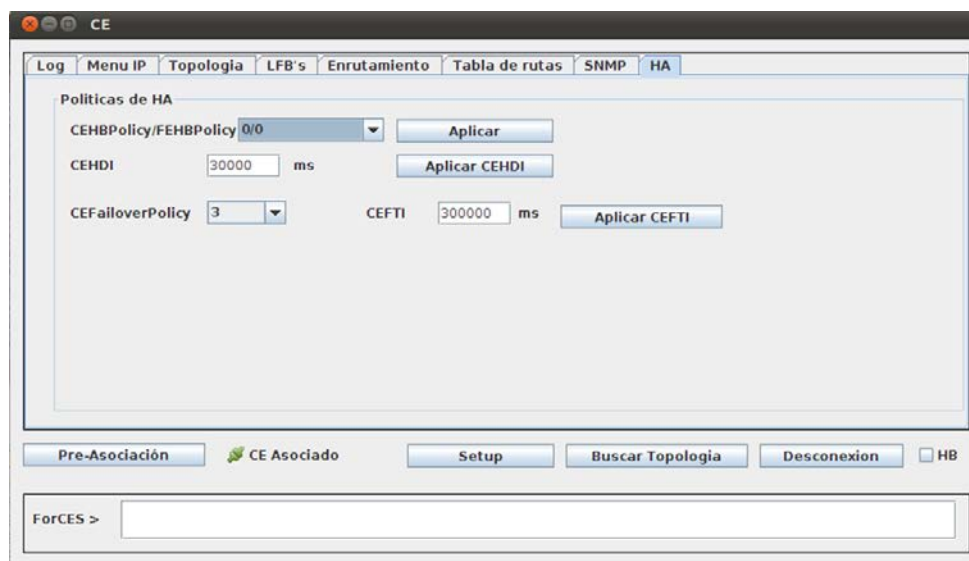


Figure 10. High Availability Module (HA)

4.2 Forwarding Element (FE)

To perform the tests that the designed CE prototype works, a test FE was developed, which contains the LFBs. These can be visualized in a FE graphical interface, it also has three Ethernet physical interfaces (Eth0, Eth1 and Eth2). The first, it is used for CE-FE interconnection (through the ForCES protocol), and the other two are used for interconnection with other NEs, in this case routers.

When using the ForCES architecture, which is a modular architecture since it separates the Control Plane and the Forwarding Plane and its elements (CEs-FEs), there are drawbacks when requiring to consult and configure the interfaces by the routing protocol and turn the sending and receiving of packages or perform a management of the interfaces by the SNMP protocol of the prototype. These two applications only monitor the interfaces of the equipment where it was installed. That is, in the CE to can send and receive routing information or SNMP packets from the FE [22], it is necessary for the CE to take control over the physical interfaces of the FE. Therefore, a virtual interface module is created that is will responsible for this process. As mentioned above, it is makes it possible to replicate the physical interfaces of the FE like logical interfaces.

Figure 11 shows the block diagram of the designed FE structure, which has an Interface ForCES module that contains the ForCES messages for the establishment of the connection with the CE. A module of LFBs, where the Extensible Markup Language (XML) that contain the different classes of the Functional Logical Blocks of the FE and the CE, and that can it consult and configure when there is an association between the FE-CE. The Virtual Client interface module, as explained above, allows a replication of the physical interfaces that the FE has in logical interfaces to reflect them towards the Virtual Interface module that is in the CE, so that the CE believes that these remote interfaces belong to it.

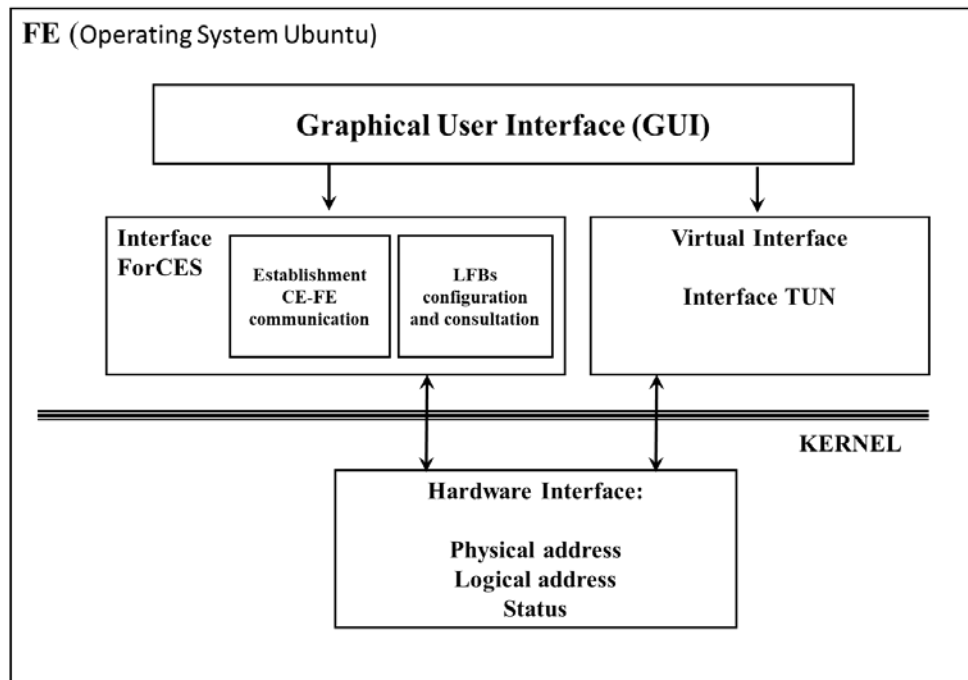


Figure 11 Structure of the Forwarding Element

This FE, is a CPU with Pentium IV processor, 512 RAM and 80 GB hard drive. It has three network cards (Eth0, Eth1, Eth2). The cards with the Eth0 and Eth1 interfaces are the interfaces that are going to be virtualized and the card with the Eth2 interface is the one that is connected to the CE through the ForCES protocol.

5. Development

This section shows the development and subsequent implementation of an application elaborated in Java, that allows a conventional personal computer (PC) to act as a CE entity or FE. More information about this implementation, it is found at [23].

In Figure 12, the topology that was used to perform the prototype test protocol is shown. Two PCs (one desktop and one laptop) were used as CE, one main (with address 192.168.5.3) and the other backup (with address 192.168.5.4). Another PC was configured as FE, this computer has three network cards as mentioned above, the Eth0 and Eth1 interfaces (to be virtualized by the CE) and the Eth2 interface (ForCES connection to the CE). Two CISCO routers were also configured that allow the interconnection between LAN A and LAN B.

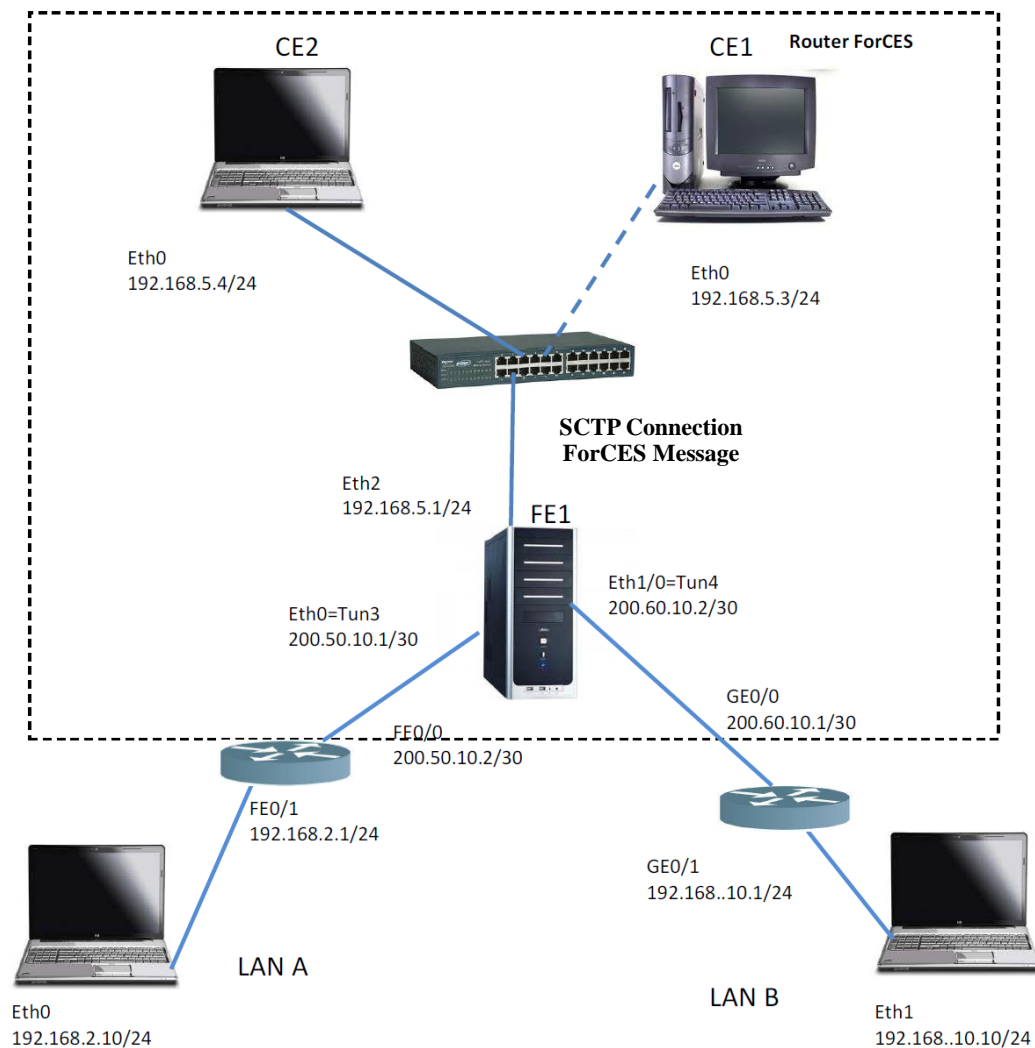


Figure 12. Topology for the test protocol

6. Configuration and Tests

In this section, the application developed in Java is configured, as CE on a PC and the other as FE, and connection tests are performed through the ForCES protocol.

The advantage of working with a versatile application and easy to install, is that it allows to activate the system of any computer with any configuration of red or topology, the roll of CE. In this way, create networks under the concept of a ForCES router.

6.1 Configurations

6.1.1 ForCES module

For the interconnection of the ForCES protocol, both in the CE and in the FE, the button with the Pre-Association command is pressed, which has the associated preassociation setup command. When performing the pre-association, the CE reads the CE Properties file and displays the CE console data, as shown in Figure 13.

Then the channel connection is made, which is executed by pressing the setup button associated with the tmlsetup command, which is executed first in the CE and then in the FE. The CE remains listening on the ports until the FE makes the connection, as shown in Figures 13 and 14.

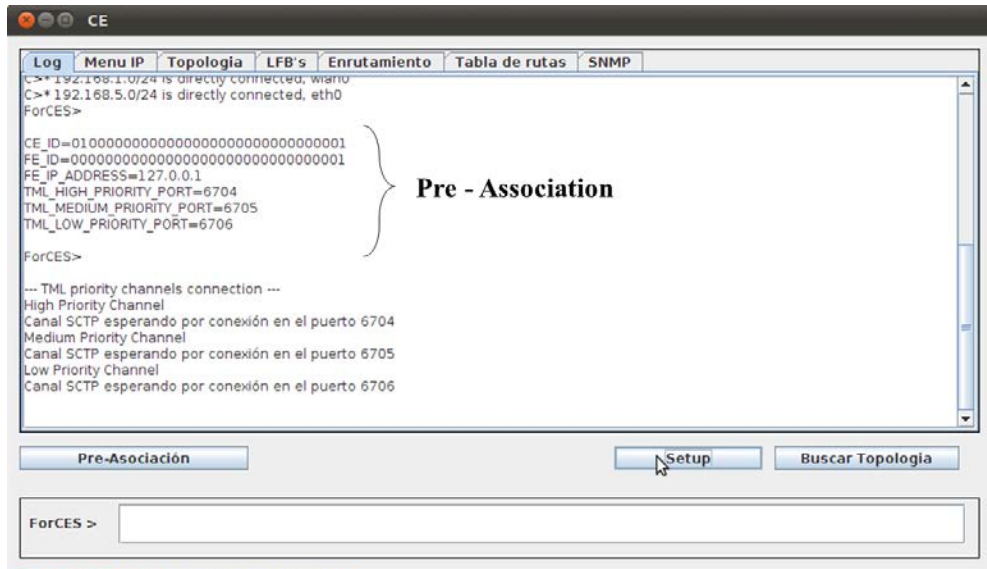


Figure 13. GUI of the CE

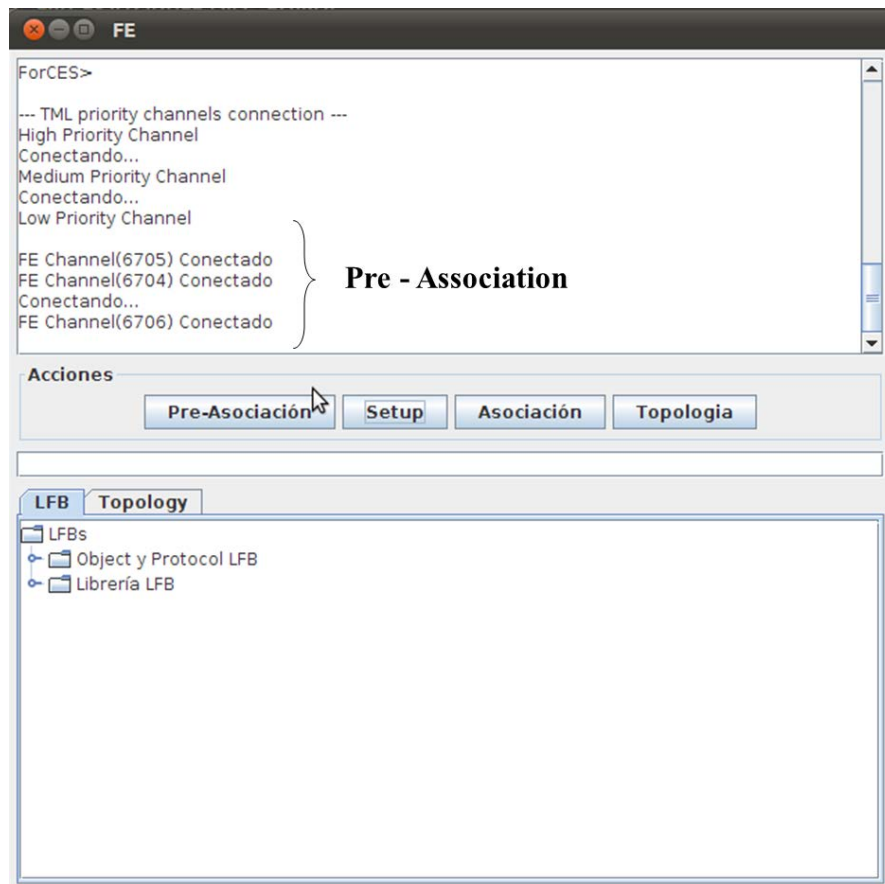


Figure 14. GUI of the FE

Finally, the FE is associated with the CE by pressing the Association button, associated with the association setup message. The CE responds with an association Setup Response, as shown in Figure 15.

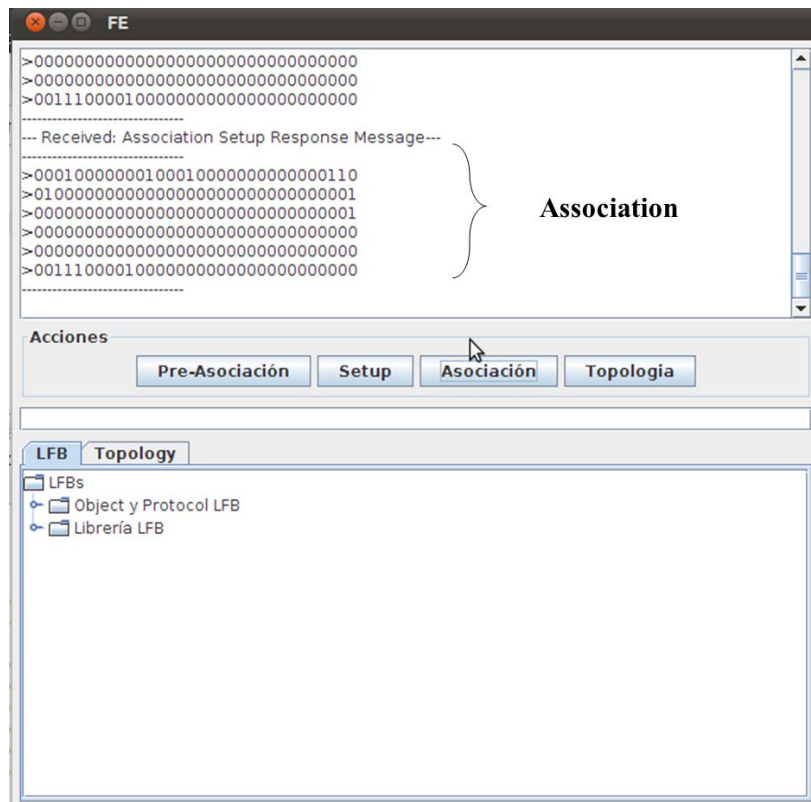
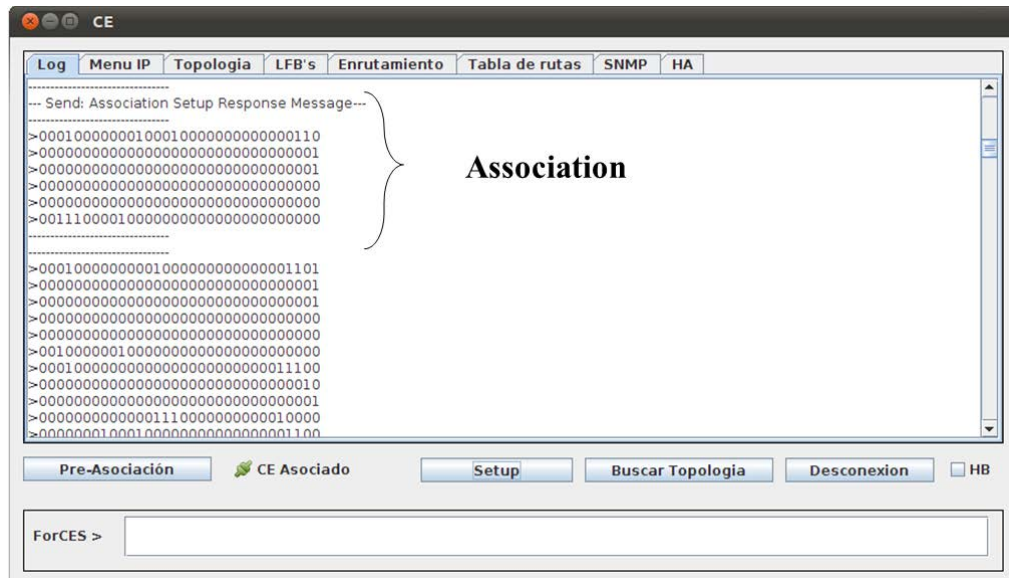


Figure 15. State of Association between CE and FE

6.1.2 IP Module

By pressing the List button, the CE by means of a Query message, consult the FE which interfaces are available. The FE responds with a Query Response message to the CE and in the third and fourth columns, Interface Name and MAC Address, the names of the interfaces and their physical addresses are respectively displayed, as shown in Figure 16.

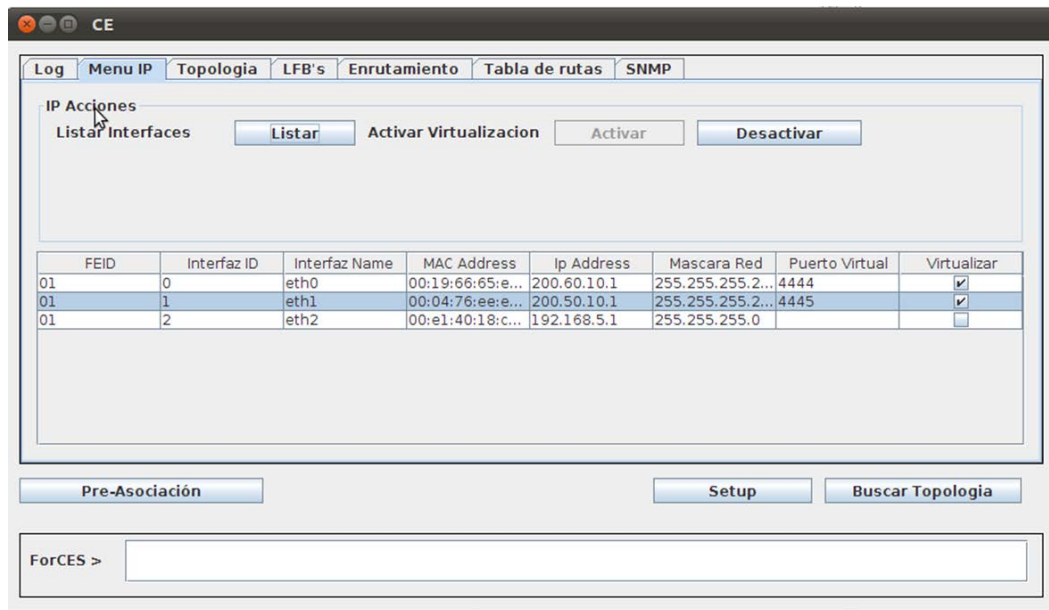


Figure 16. IP module

6.1.3 Virtual Interface Module

To virtualize the interfaces, in this case the interfaces eth0 and eth1 (eth2, not virtualized, since this interface is the connection from the FE to the CE), first each interface is configured with the IP address and the desired subnet mask, the next step is, in the Virtual Port column, enter port 4444 for an interface and 4445 for the other interface (in section 1.1.2, it explains why those ports). Then, in the Virtualize column, you click on each frame of each interface. Finally, the Activate button is pressed. A message "Successful active interfaces" should appear, indicating that the interfaces eth0 and eth1, the CE took them as their own. This is seen in Figure 17.

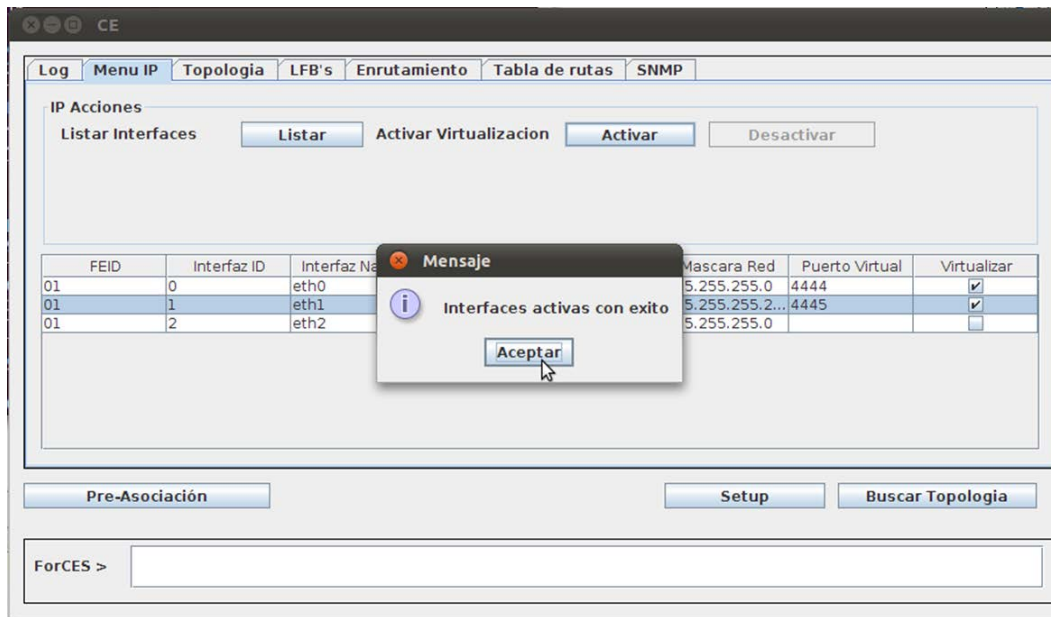


Figure 17. Virtualization of the interfaces

To verify that the Tun3 and Tun4 interfaces were configured in the CE, in the operating system console you type `root @susanpc:/home/susan# ifconfig -a`

In Figure 18, the logical addresses of the virtual interfaces configured in the GUI are observed.

```
eth0      Link encap:Ethernet direcciónHW 00:23:8b:e2:41:bd
          Direc. inet:192.168.5.3 Difus.:192.168.5.255 Másc:255.255.255.0
          Dirección inet6: fe80::223:8bff:fee2:41bd/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
          Paquetes RX:47 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:43 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:8070 (8.0 KB) TX bytes:6157 (6.1 KB)
          Interrupción:16

lo        Link encap:Bucle local
          Direc. inet:127.0.0.1 Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO MTU:16436 Métrica:1
          Paquetes RX:38 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:38 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:0
          Bytes RX:3392 (3.3 KB) TX bytes:3392 (3.3 KB)

tun3     Link encap:Ethernet direcciónHW 00:19:66:65:e5:59
          Direc. inet:192.168.10.1 Difus.:192.168.10.255 Másc:255.255.255.0
          DIFUSIÓN MULTICAST MTU:1500 Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:500
          Bytes RX:0 (0.0 B) TX bytes:0 (0.0 B)

tun4     Link encap:Ethernet direcciónHW 00:04:76:ee:ec:d6
          Direc. inet:200.50.10.1 Difus.:200.50.10.3 Másc:255.255.255.252
          DIFUSIÓN MULTICAST MTU:1500 Métrica:1
          Paquetes RX:1 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:500
          Bytes RX:54 (54.0 B) TX bytes:0 (0.0 B)

wlan0    Link encap:Ethernet direcciónHW 00:24:d2:98:9a:11
          Direc. inet:192.168.1.33 Difus.:192.168.1.255 Másc:255.255.255.0
          Dirección inet6: fe80::224:d2ff:fe98:9a11/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
          Paquetes RX:147 errores:0 perdidos:0 overruns:0 frame:0
```

Figure 18. IP addresses of the CE interfaces

6.1.4 Routing Module

When virtualizing the eth0 and eth1 interfaces (physical interfaces of the FE), the CE identifies them as their own, with the virtual names Tun3 and Tun4, respectively. In the routing module, there is a tab where you can select the interface to check the status (up or down), the logical address and the subnet mask, this is done by selecting the upper left label called "Selection of Int ...".

If any interface (Tun 3 or Tun4) is in the down state, it goes to the up state, selecting No shutdown and pressing the Save button.

To configure the routing protocol, in this case RIPv2, the RIP button is selected (upper right) and in "Network Address" (lower left) the neighboring networks are typed, <network IP address/CIDR> and press "Add" and "Save" to save the changes [24].

In case you want to delete a network address, enter the address <IP address of the network/CIDR>, select "Delete" and finally press "Save" [24].

The "Add IP / IP Address" label is used if you want to change the IP address of a Tun interface.

In Figure 19, the graphical interface of the Routing Module is observed, for this prototype the RIPv2 protocol is configured, although as it was previously mentioned in section 4, **Quagga** supports OSPFv2, v3 and BGP that could be configured in a future version of this prototype. The GUI of the routing module is shown in Figure 19.

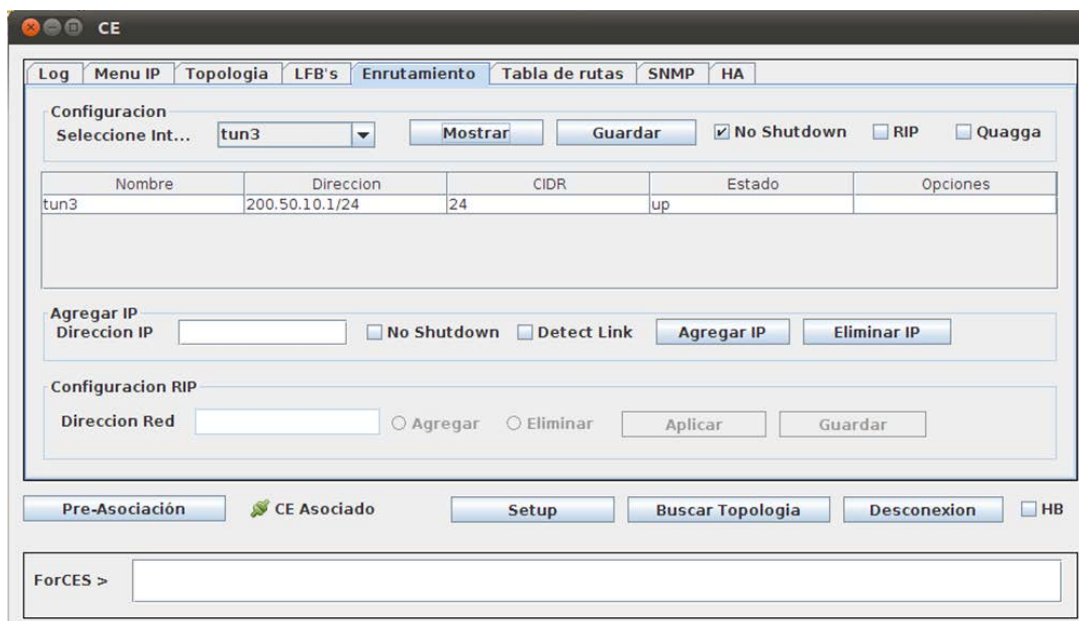


Figure 19. GUI of the routing module

In the upper left, you can see the TUN interfaces (which are the virtual interfaces). First the interface is selected and then the "Show" button is pressed, which displays in the table the name of the interface, the IP address, the subnet mask and the status (up / down).

To visualize the routing table, select the "Table of routes" tab, which is shown in the Figure 20.

Tipo	Descripcion	Direccion IP	Metrica	Proximo Salto	Interfaz
K	Kernel Route	169.254.0.0/16	-	-	eth0
R	RIP	192.168.2.0/24	[120/1]	200.50.10.2	tun3
C	Connected	192.168.5.0/24	-	-	eth0
R	RIP	192.168.10.0/24	[120/1]	200.60.10.1	tun4
C	Connected	200.50.10.0/24	-	-	tun3
C	Connected	200.60.10.0/30	-	-	tun4

Actualizar Tabla

Pre-Asociación CE Asociado HB

ForCES >

Figure 19. Routing table

6.1.5 Management Module

It is responsible for displaying some parameters such as: bytes sent, bytes received, packets sent and packets received by each of the interfaces, allowing monitoring of the network and the router. It is observed in Figure 21.

Interfaz	Arbol	Bytes Enviados	Bytes Recibidos	Paquetes Enviados	Paquetes Recibidos
lo	iso.3.6.1.2.1.31.1.1....	175735	175735	2185	2185
eth0	iso.3.6.1.2.1.31.1.1....	6432885	8043449	60298	64033
wlan0	iso.3.6.1.2.1.31.1.1....	0	0	0	0
tun3	iso.3.6.1.2.1.31.1.1....	960148	1800188	20613	10695
tun4	iso.3.6.1.2.1.31.1.1....	955974	1930495	22076	10713

Actualizar

Pre-Asociación CE Asociado HB

ForCES >

Figure 21. Management module

6.1.6 High Availability Module

The parameters to be configured in this module are the following:

- **CEHDI in 30000 ms:** This time, in milliseconds, is the time for the FE to determine that the main CE was disconnected (it does not receive heartbeat messages), and start a search for a backup CE.
- **CEFailoverPolicy in 3:** Indicates that the prototype has high availability with a restart.
- **CEFTI in 300000 ms:** If during this time, the FE fails to connect and associate again with a backup CE, the system stops working.

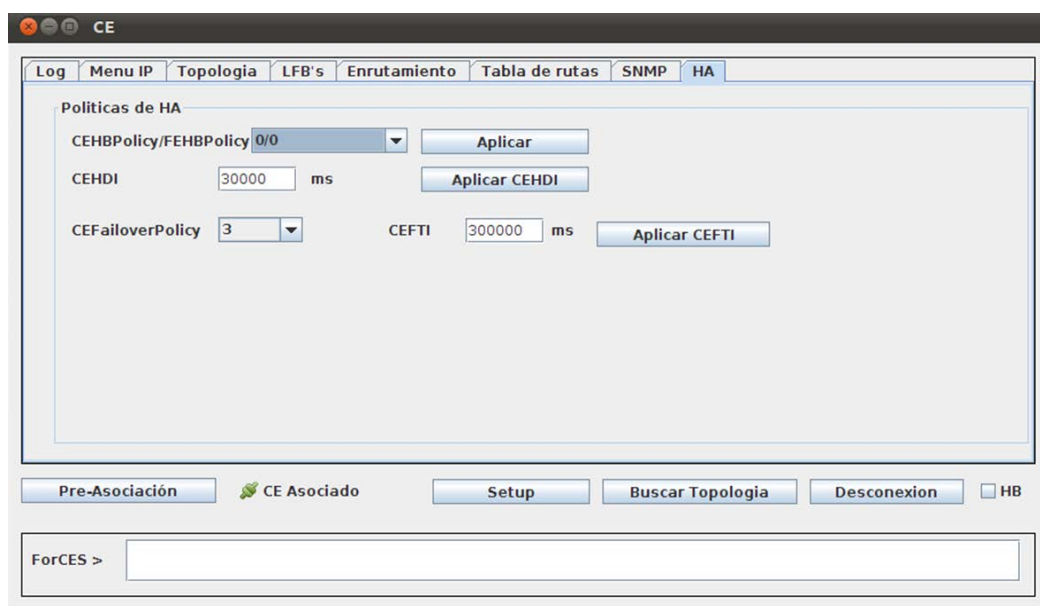


Figure 22. High Availability Module

In Figure 22, the HB Request message and the HB Response message are observed, when the High Availability policies in the CEHBPoly / FEHBPoly are set to 0/0.

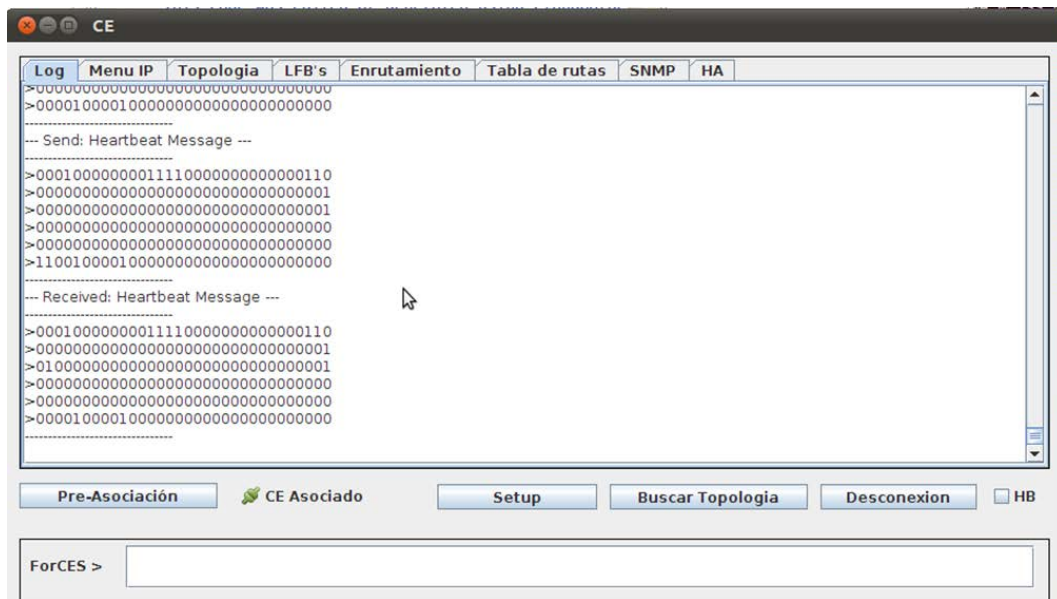


Figure 22. Heartbeat message sent and received from the CE-FE and FE-CE respectively

Once all the parameters have been configured, the interconnection tests are carried out.

6.2 Interconnection Tests

6.2.1 Verification interface ForCES

Ping from the operating system terminal of the CE to the address of the FE interface (eth2): root@susanpc:/home/susan# ping 192.168.5.1

and then a ping is made from the eth2 interface of the FE to the eth0 interface of the CE: root@susanpc:/home/susan# ping 192.168.5.3

If the pings respond, it means that the connection to initiate a pre-association by ForCES is possible.

6.2.2 Configuration of Interfaces and virtualization

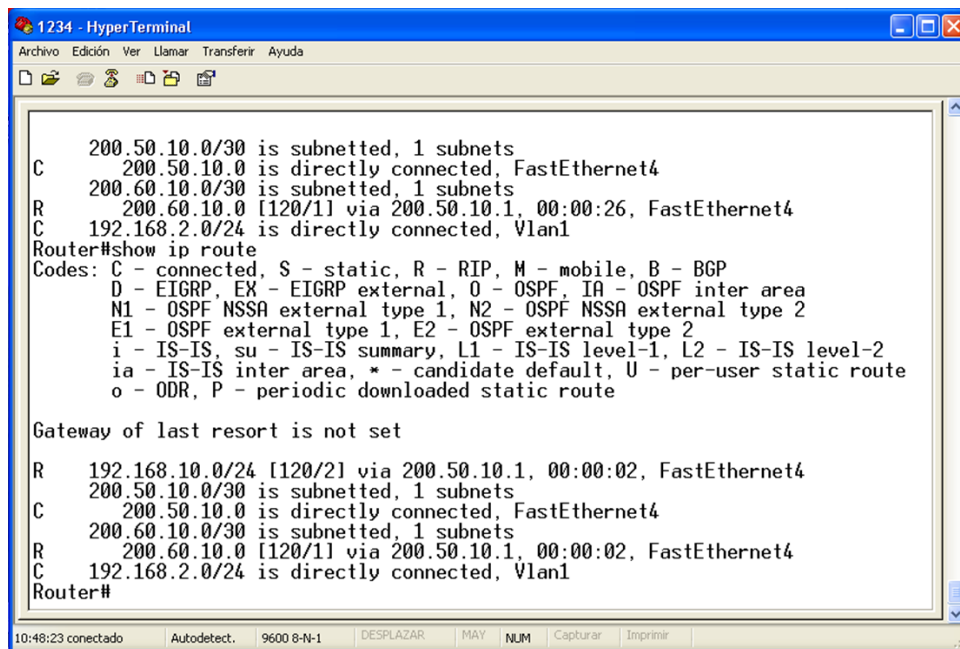
Once the CE and the FE are associated, we proceed to configure the IP addresses of the interfaces from the CE GUI and their virtualization.

6.2.3 Configuration of the RIPv2 Protocol

Once the interfaces are virtualized, we proceed to configure the routing protocol RIPv2.

Once RIP is configured, it is verified that both in the designed prototype and in the Cisco routers the routing tables are really learning routes by this protocol.

The routing table in Figure 19 shows that the prototype learns two networks per RIP, the 192.168.2.0/24 network with metric [120/1] through the Tun3 interface and that the next hop is address 200.50.10.2. The other network is 192.168.10.0/24 with metric [120/1] through the Tun4 interface and the next hop is address 200.60.10.1.



```

200.50.10.0/30 is subnetted, 1 subnets
C   200.50.10.0 is directly connected, FastEthernet4
200.60.10.0/30 is subnetted, 1 subnets
R   200.60.10.0 [120/1] via 200.50.10.1, 00:00:26, FastEthernet4
192.168.2.0/24 is directly connected, Vlan1
Router#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

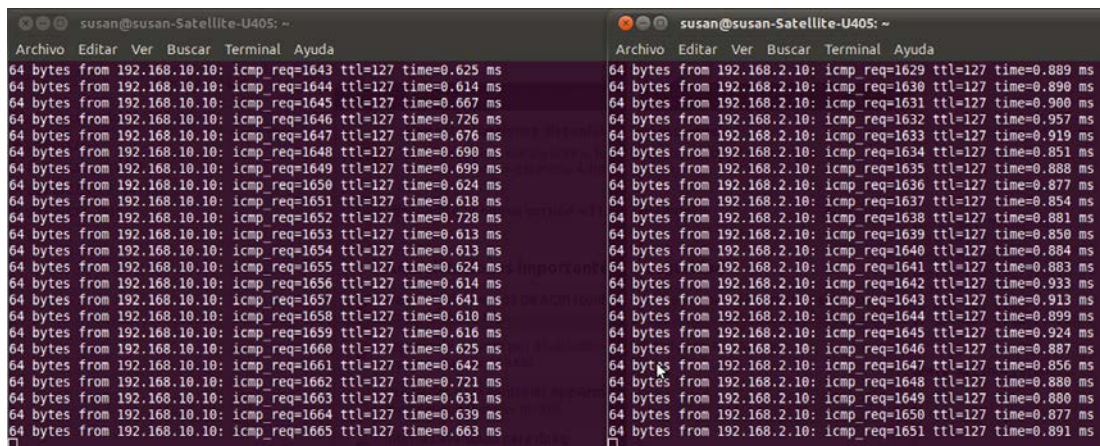
Gateway of last resort is not set

R   192.168.10.0/24 [120/2] via 200.50.10.1, 00:00:02, FastEthernet4
200.50.10.0/30 is subnetted, 1 subnets
C   200.50.10.0 is directly connected, FastEthernet4
200.60.10.0/30 is subnetted, 1 subnets
R   200.60.10.0 [120/1] via 200.50.10.1, 00:00:02, FastEthernet4
192.168.2.0/24 is directly connected, Vlan1
Router#
    
```

Figure 24. routing table of the Cisco R1 router

In Figure 24, you can see the routing table of the Cisco R1 router. This learns the network 192.168.10.0/24, with metric [120/2] by means of the 200.50.10.1 and learns the network 200.60.10.0/30, with metric [120/2] by means of the 200.50.10.1

In Figure 25, the pings are verified from the prototype designed to the PCs of the networks 192.168.2.0 and 192.168.10.0 see also [25].



```

susan@susan-Satellite-U405: ~
Archivo Editar Ver Buscar Terminal Ayuda
64 bytes from 192.168.10.10: icmp_req=1643 ttl=127 time=0.625 ms
64 bytes from 192.168.10.10: icmp_req=1644 ttl=127 time=0.614 ms
64 bytes from 192.168.10.10: icmp_req=1645 ttl=127 time=0.667 ms
64 bytes from 192.168.10.10: icmp_req=1646 ttl=127 time=0.726 ms
64 bytes from 192.168.10.10: icmp_req=1647 ttl=127 time=0.676 ms
64 bytes from 192.168.10.10: icmp_req=1648 ttl=127 time=0.690 ms
64 bytes from 192.168.10.10: icmp_req=1649 ttl=127 time=0.699 ms
64 bytes from 192.168.10.10: icmp_req=1650 ttl=127 time=0.624 ms
64 bytes from 192.168.10.10: icmp_req=1651 ttl=127 time=0.618 ms
64 bytes from 192.168.10.10: icmp_req=1652 ttl=127 time=0.728 ms
64 bytes from 192.168.10.10: icmp_req=1653 ttl=127 time=0.613 ms
64 bytes from 192.168.10.10: icmp_req=1654 ttl=127 time=0.613 ms
64 bytes from 192.168.10.10: icmp_req=1655 ttl=127 time=0.624 ms
64 bytes from 192.168.10.10: icmp_req=1656 ttl=127 time=0.614 ms
64 bytes from 192.168.10.10: icmp_req=1657 ttl=127 time=0.641 ms
64 bytes from 192.168.10.10: icmp_req=1658 ttl=127 time=0.610 ms
64 bytes from 192.168.10.10: icmp_req=1659 ttl=127 time=0.616 ms
64 bytes from 192.168.10.10: icmp_req=1660 ttl=127 time=0.625 ms
64 bytes from 192.168.10.10: icmp_req=1661 ttl=127 time=0.642 ms
64 bytes from 192.168.10.10: icmp_req=1662 ttl=127 time=0.721 ms
64 bytes from 192.168.10.10: icmp_req=1663 ttl=127 time=0.631 ms
64 bytes from 192.168.10.10: icmp_req=1664 ttl=127 time=0.639 ms
64 bytes from 192.168.10.10: icmp_req=1665 ttl=127 time=0.663 ms

susan@susan-Satellite-U405: ~
Archivo Editar Ver Buscar Terminal Ayuda
64 bytes from 192.168.2.10: icmp_req=1629 ttl=127 time=0.889 ms
64 bytes from 192.168.2.10: icmp_req=1630 ttl=127 time=0.896 ms
64 bytes from 192.168.2.10: icmp_req=1631 ttl=127 time=0.906 ms
64 bytes from 192.168.2.10: icmp_req=1632 ttl=127 time=0.957 ms
64 bytes from 192.168.2.10: icmp_req=1633 ttl=127 time=0.919 ms
64 bytes from 192.168.2.10: icmp_req=1634 ttl=127 time=0.851 ms
64 bytes from 192.168.2.10: icmp_req=1635 ttl=127 time=0.888 ms
64 bytes from 192.168.2.10: icmp_req=1636 ttl=127 time=0.877 ms
64 bytes from 192.168.2.10: icmp_req=1637 ttl=127 time=0.854 ms
64 bytes from 192.168.2.10: icmp_req=1638 ttl=127 time=0.881 ms
64 bytes from 192.168.2.10: icmp_req=1639 ttl=127 time=0.850 ms
64 bytes from 192.168.2.10: icmp_req=1640 ttl=127 time=0.884 ms
64 bytes from 192.168.2.10: icmp_req=1641 ttl=127 time=0.883 ms
64 bytes from 192.168.2.10: icmp_req=1642 ttl=127 time=0.933 ms
64 bytes from 192.168.2.10: icmp_req=1643 ttl=127 time=0.913 ms
64 bytes from 192.168.2.10: icmp_req=1644 ttl=127 time=0.899 ms
64 bytes from 192.168.2.10: icmp_req=1645 ttl=127 time=0.924 ms
64 bytes from 192.168.2.10: icmp_req=1646 ttl=127 time=0.887 ms
64 bytes from 192.168.2.10: icmp_req=1647 ttl=127 time=0.856 ms
64 bytes from 192.168.2.10: icmp_req=1648 ttl=127 time=0.880 ms
64 bytes from 192.168.2.10: icmp_req=1649 ttl=127 time=0.880 ms
64 bytes from 192.168.2.10: icmp_req=1650 ttl=127 time=0.877 ms
64 bytes from 192.168.2.10: icmp_req=1651 ttl=127 time=0.891 ms
    
```

Figure 25. Ping test from prototype to neighboring networks

6.2.4 Configuration of the Management Module

To verify the operation of the Net-SNMP management protocol implemented in the CE prototype, the MIB Browser software was used in host of each configured LAN network (192.168.2.0/24 and 192.168.10.0/24). These tests are shown in Figure 26.

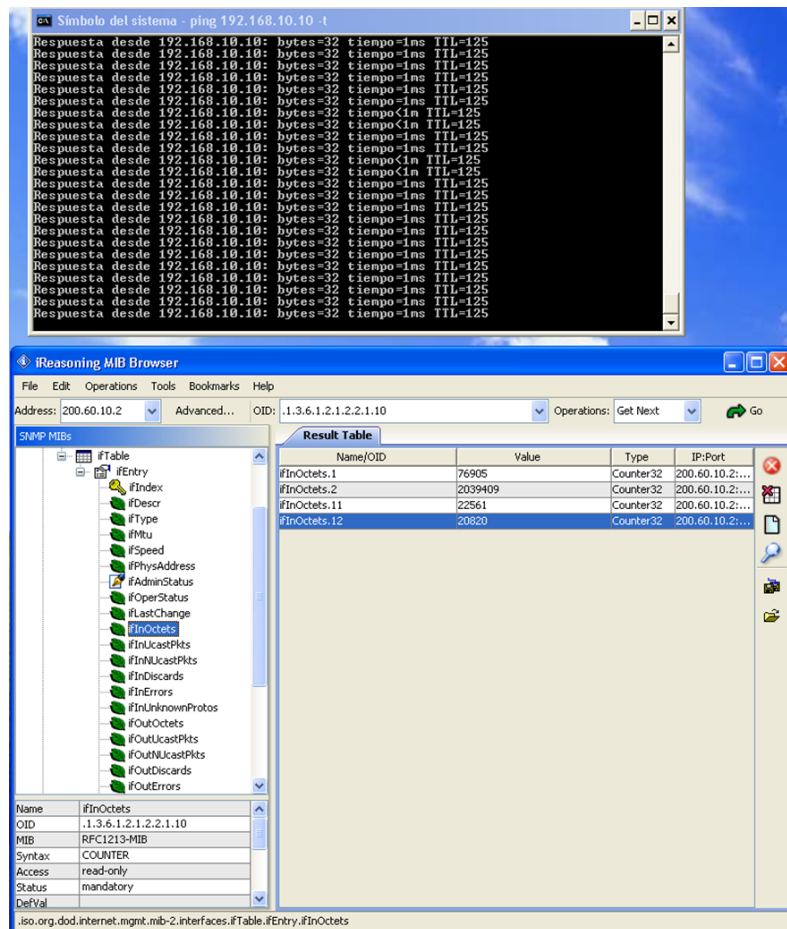


Figure 26. MIB Browser in the LAN 192.168.10.0/24

7. Conclusion and future work

With the development of this work, an application of the CE entity (based on the ForCES architecture) is made available to researchers, which can be used to test real networks connected in a LAN network or even connected through a Wide Area Network (WAN). Providing new information on the study of this type of architectures and then compare it with SDN systems.

Looking forward to the possibility of developing tests similar to those performed by the two research groups that have designed ForCES modular routers. The research group of the University of Zhejiang Gongshang of China, who carried out the interconnection of several control elements and several elements of forwarding remotely (in different cities) through the ForCES protocol. The research group of the Royal Institute of Technology of Stockholm, who designed a modular router with ForCES architecture, but with an interconnection protocol between the CE and the FE developed by them.

Conventional routers are devices that have a closed architecture, are not scalable or reconfigurable since the control element and the forwarding element are part of a single structure and it is not possible to make improvements separately in each one. If the device

does not support certain services such as Quality of Service (QoS) or Voice over IP (VoIP), for example, then it is necessary to make a change of it. The prototype designed, when working with the ForCES architecture, which is an open architecture, makes this a modular, scalable and low cost device, since it allows to make improvements in each element that composes it separately (control element and the element of forwarding).

The implementation of the virtual interfaces between the elements of control and forwarding, allows the ForCES architecture to see these two elements as a single network element, even if they are located at remote sites each.

The system of high availability developed, allows the prototype to be a little more robust, because if the connection is lost between the main control element and the element of forwarding, which are part of the network element, there is another control element like backup, in this way the device continues to function unless the fault is present in the FE.

For future work, we will do more tests, configuring the application with OSPFv2, v3 and BGP in different scenarios and topologies to see the behavior of the network and measure its performance, the time of operation and the Quality of Service (QoS).

References

- [1] Cisco White Paper, "Cisco Visual Networking Index: Forecast and Methodology, 2016–2021", September 15, 2017. Available Cisco White Paper at website: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html
- [2] IETF ForCES Working Group. Available at IETF website: <https://datatracker.ietf.org/wg/forces/about/>
- [3] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004. Available at IETF website: <https://tools.ietf.org/html/rfc3746>. DOI: <https://doi.org/10.17487/RFC3746>
- [4] Ogawa, K., Wang, W., Haleplidis, E., and J. Hadi Salim, "High Availability within a Forwarding and Control Element Separation (ForCES) Network Element", RFC 7121, February 2014. Available at IETF website: <https://tools.ietf.org/html/rfc7121>. DOI: <https://doi.org/10.17487/RFC7121>
- [5] 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. Available at IETF website: <https://tools.ietf.org/html/rfc6956>. DOI: <https://doi.org/10.17487/RFC6956>
- [6] 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>
- [7] P. L. González Ramírez, "Diseño e Implementación del Protocolo ForCES," Pontificia Universidad Javeriana, Master's Thesis, 2012. DOI: <https://doi.org/10554/20769>

- [8] P. L. Gonzalez Ramirez, J. Lloret, S. Martínez Cordero, and L. C. Trujillo Arboleda, "Design and Implementation of ForCES Protocol," *Network, Protocol and Algorithms*, vol. 9, Issue 1–2, pp 1-27. June 30 - 2017. DOI: <https://doi.org/10.5296/npa.v9i1-2.10943>
- [9] W. M. Wang, L. G. Dong, and B. Zhuge, "Analysis and implementation of an open programmable router based on forwarding and control element separation," *J. Comput. Sci. Technol.*, vol. 23, no. 5, pp. 769–779, Sep. 2008. DOI: <https://doi.org/10.1007/s11390-008-9181-4>
- [10] W. Wang, L. Dong, B. Zhuge, M. Gao, F. Jia, R. Jin, J. Yu, X. Wu, "Design and Implementation of an Open Programmable Router Compliant to IETF ForCES Specifications," in *Sixth International Conference on Networking (ICN'07)*, 2007, pp. 82–82. DOI: <https://doi.org/10.1109/ICN.2007.35>
- [11]. K. Ogawa, W. Wang, E. Haleplidis, and J. Hadi Salim, "ForCES Intra-NE High Availability." draft-ietf-forces-ceha-03. Available at IETF website: <https://tools.ietf.org/html/draft-ietf-forces-ceha-03>
- [12] Q. Li, L. Dong and M. Gao, "Research on High-Availability Based on Architecture of ForCES," 2009 Asia-Pacific Conference on Information Processing, Shenzhen, 2009, pp. 537-540. DOI: <https://doi.org/10.1109/APCIP.2009.268>
- [13] X. Wu and L. Dong, "Research and Design of the High Availability Mechanism in the ForCES Router." Available at website: <http://netcom.zjgsu.edu.cn/publications/English-paper/AICCSA2009-Xiaochun Wu.pdf>
- [14] Weiping Yu, Bin Zhuge and Weiming Wang, "Inter-FE Topology Discovery and maintenance technology on ForCES routers," *The 19th Annual Wireless and Optical Communications Conference (WOCC 2010)*, Shanghai, 2010, pp. 1-4. DOI: <https://doi.org/10.1109/WOCC.2010.5510651>
- [15] O. Hagsand, M. Hidell and P. Sjodin, "Design and implementation of a distributed router," *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology*, Athens, 2005, pp. 227-232. DOI: <https://doi.org/10.1109/ISSPIT.2005.1577100>
- [16] S. Yan, C. Li, M. Gao, W. Wang, L. Dong, and B. Zhuge, "A Network Controller Supported Open Reconfigurable Technology,". In: Leung V., Chen M., Wan J., Zhang Y. (eds), *Book: "Testbeds and Research Infrastructure: Development of Networks and Communities"*. vol 137. Springer, Cham, 2014, pp. 395–405. ISBN: 978-3-319-13325-6. DOI: https://doi.org/10.1007/978-3-319-13326-3_38
- [17] G. Tarnaras, E. Haleplidis and S. Denazis, "SDN and ForCES based optimal network topology discovery," *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, London, 2015, pp. 1-6. DOI: <https://doi.org/10.1109/NETSOFT.2015.7116181>
- [18] E. Haleplidis et al., "Network Programmability with ForCES," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1423-1440, thirdquarter 2015. DOI: <https://doi.org/10.1109/COMST.2015.2439033>
- [19] 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>

- [20] Khosravi, H., Ed., and T. Anderson, Ed., "Requirements for Separation of IP Control and Forwarding", RFC 3654. Available at IETF website: <https://tools.ietf.org/html/rfc3654>. DOI: <https://doi.org/10.17487/RFC3654>
- [21] J. Hadi Salim; J. Halpern, "Forwarding and Control Element Separation (ForCES) Element Model Forwarding," RFC5812, March, 2010. Available at IETF website: <https://tools.ietf.org/html/rfc5812>. DOI: <https://doi.org/10.17487/RFC5812>
- [22] Baker, F. and R. Coltun, "OSPF Version 2 Management Information Base", RFC 1850, November 1995. Available at IETF website: <https://tools.ietf.org/html/rfc1850>. DOI: <https://doi.org/10.17487/RFC1850>
- [23] 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>
- [24] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, June 1995. Available at IETF website: <https://tools.ietf.org/html/rfc1812>. DOI: <https://doi.org/10.17487/RFC1812>
- [25] F. Luo, L. Dong and F. Jia, "Method and implementation of building ForCES protocol dissector based on wireshark," 2010 2nd IEEE International Conference on Information Management and Engineering, Chengdu, 2010, pp. 291-294. DOI: <https://doi.org/10.1109/ICIME.2010.5478081>

Copyright Disclaimer

Copyright reserved by the author(s).

This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).