UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería Técnica de Telecomunicación,
especialidad en Telemática

Proyecto Fin de Carrera

LABORATORIO VIRTUAL UC3M: DESARROLLO DE UN ENTORNO NO PRESENCIAL PARA LA REALIZACIÓN DE PRÁCTICAS DE CONFIGURACIÓN DE RED Y SU AUTOCORRECCIÓN

Alumno: María Celeste Durán González
Tutor: Carlos Jesús Bernardos Cano

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería Técnica de Telecomunicación,
especialidad en Telemática

Proyecto Fin de Carrera

LABORATORIO VIRTUAL UC3M: DESARROLLO DE UN ENTORNO NO PRESENCIAL PARA LA REALIZACIÓN DE PRÁCTICAS DE CONFIGURACIÓN DE RED Y SU AUTOCORRECCIÓN

Alumno: María Celeste Durán González
Tutor: Carlos Jesús Bernardos Cano

Este trabajo se encuentra bajo la licencia *Creative Commons Reconocimiento-NoComercial-CompartirIgual 2.5 España*.
This work is licensed under the *Creative Commons Attribution-NonCommercial-ShareAlike 2.5 Spain License*.

http://creativecommons.org/licenses/by-nc-sa/2.5/es/legalcode.es

Resumen

Desde hace unos años, en el Departamento de Ingeniería Telemática se imparten una serie de asignaturas cuya parte práctica requiere la configuración de distintos tipos de redes. Estas prácticas se realizan en los laboratorios de Telemática del Departamento y para llevarlas a cabo se utilizan los recursos que actualmente hay en él. Estos recursos son, principalmente, los ordenadores de las aulas y un conjunto de routers Linksys (modelos WRT54GSv4 y WRT54GLv1) que los alumnos utilizan para realizar las configuraciones que se proponen en las prácticas, y a los cuales tienen acceso únicamente durante el tiempo programado para la realización de éstas, limitando su tiempo de aprendizaje.

Actualmente, nos encontramos en un periodo de cambio educativo por la llegada de la adaptación al Espacio Europeo de Educación Superior (EEES), en el cual se prima el trabajo del alumno y el "autoaprendizaje" de éste a través de las prácticas. Para adaptar las asignaturas que se estaban impartiendo era necesario que los alumnos pudieran disponer de todas las herramientas posibles (los equipos y routers que se utilizan para las prácticas) a cualquier hora y que, al mismo tiempo, no se pusieran en riesgo los recursos de los que dispone el Departamento.

Este proyecto se centra en el desarrollo de un entorno virtual, configurable y fácilmente adaptable a las distintas asignaturas de redes de comunicaciones, gracias a una herramienta de generación de ejercicios. Este entorno permite realizar, de manera no presencial ejercicios de configuración de routers, similares a los propuestos en las sesiones de prácticas realizadas en los laboratorios del Departamento. Adicionalmente se proporciona una herramienta que permite la autocorrección de los ejercicios realizados con el entorno virtual, que no sólo ayuda al autoaprendizaje del alumno, sino que además, ayuda al profesorado, liberándole de una carga importante de trabajo, más aún en el entorno del EEES, dónde la evaluación continua juega un papel muy importante.

proyecto, entorno virtual, autocorrección

A mis padres, Francisco y Rosalía y a mi hermano, Fran, sin los cuales no podría haber llegado hasta aquí.

A mis amigo/as y compañero/as, Inés, Nacho, Humberto, Rico, Miguel, Chemi, Rober, Virgi, Pablo y David que me han hecho el camino más agradable y fácil.

A Goyo y a Rafa por darme animos y consejos cuando lo he necesitado.

A Raúl, por haberme ayudado y apoyado como nadie.

A mi tutor Carlos Jesús que me ha ayudado en el desarrollo de este trabajo y ha tenido una enorme paciencia.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
INTRODUCCIÓN

1.1. INTRODUCCIÓN

Este documento presenta el trabajo realizado como Proyecto Fin de Carrera que se centra en el desarrollo de un entorno virtual, configurable y auto-corregible que permite realizar de manera no presencial ejercicios de configuración de redes y routers similares a los propuestos en las sesiones de prácticas de las asignaturas de redes de comunicaciones. En la actualidad se están cursando varias de estas asignaturas en la Universidad Carlos III de Madrid que incluyen el uso de este entorno como parte de su programa, con el ánimo de ofrecer más herramientas de aprendizaje a los alumnos. Dado el contenido de este trabajo, que aporta una herramienta novedosa, orientada a la docencia, fácil de manejar y la cual está siendo utilizada actualmente por los alumnos de la Universidad, se han desarrollado dos proyectos de innovación docente que tienen este trabajo como base de su contenido. Uno de estos proyectos está relacionado con el entorno virtual y el otro proyecto de innovación docente está relacionado con la herramienta de autocorrección.

Nos encontramos en una sociedad en la que las redes forman parte de nuestra vida. En cualquier sitio encontramos una red, de un tipo u otro, y lo bien o mal que funcione dicha red, o lo bien o mal que se puedan aprovechar los recursos de los que se dispongan, dependen en gran medida del diseño que se haya hecho de dicha red, por lo que nos podemos hacer una idea de lo importante que puede ser conocer las opciones de configuración existentes y los parámetros de red configurables. El diseño de redes de ordenadores puede llegar a ser un campo muy complicado por la gran cantidad de parámetros configurables que podemos modificar y por el gran número de nodos que podemos llegar a tener en un mismo escenario de red, por lo tanto, se considera que el diseño de redes un campo muy importante dentro de la enseñanza en el ámbito de las telecomunicaciones.

El entorno virtual del que hemos partido para crear nuestra herramienta ha sido VNUML (Virtual Network User Mode Linux) que es una herramienta de virtualización de propósito general basada en UML (User Mode Linux).

Las motivaciones que han llevado al desarrollo de este proyecto son diversas. Una de estas motivaciones es que el departamento dispone de un número de routers determinado, y esto provoca que los ejercicios de configuración que se proponen en las prácticas tengan una dificultad limitada, ya que hay que repartir los recursos entre todos los alumnos. Sin embargo, utilizando un entorno virtual en el cual se pueden simular estos routers y los equipos del laboratorio, el número de posibles configuraciones aumenta considerablemente, debido a que la única limitación es la capacidad del equipo

donde se simularán los escenarios configurables. Otra motivación que ha llevado al desarrollo de este proyecto ha sido proporcionar una herramienta al alumno que le permitiera preparar las prácticas y repetirlas las veces que deseen con el objetivo de aclarar dudas, todo ello sin necesidad de tener los routers físicamente y sin tener limitaciones de tiempo, como ocurre en las prácticas que se han desarrollado hasta ahora de este tipo en el Departamento. Además, cada vez que se cierra el entorno virtual se dejan las máquinas virtuales tal y como estaban antes de ser configuradas y se evitan problemas derivados de golpes o mal uso físico del router. Es importante aclarar que en ningún caso se pretende que la herramienta virtual sustituya por completo al entorno físico, pero sí que lo complemente. Los problemas que pueden surgir cuando alguien se enfrenta a una máquina real forman una parte muy importante del aprendizaje que, ningún caso, se quiere dejar de ofrecer en las prácticas docentes.

Al profesorado se le ofrece en general dos cosas: un entorno virtual con el que se pueden realizar ejercicios adicionales a los propuestos en el laboratorio y un entorno de autocorrección que permite liberarle de una gran carga de trabajo. Dentro de estos grandes bloques, además se facilitan herramientas para una sencilla definición de escenarios y plantillas de autocorrección.

1.2. ESTRUCTURA DE LA MEMORIA

La presente memoria del Proyecto de Fin de Carrera se encuentra dividida en las siguientes partes:

- **PARTE I: Introducción.** Introducción general del proyecto, presentación de los objetivos del proyecto y descripción de la estructura de la memoria.
- **PARTE II: Estado del arte y antecedentes.** Se presenta cual es la situación previa al desarrollo de este Proyecto. Se explican los antecedentes en el tipo de hardware de red, para las prácticas, que se ha utilizado en la Universidad y también se hace un recorrido sobre los distintos tipos de entornos de virtualización que existen y cuál ha sido la elegida para el desarrollo de este trabajo. Esta parte está dividida en varios capítulos.
    - *Capítulo 2. Antecedentes hardware: Routers.* En este capítulo se describen cuales han sido los routers que se han utilizado en las prácticas en los últimos años. Se explicarán los motivos por los cuales se utilizan y la experiencia docente que se ha tenido con ellos.
    - *Capítulo 3. Antecedentes software: Entornos de virtualización.* Se presentan las características de diferentes entornos de virtualización, sus ventajas y desventajas. En base a este análisis se explicarán los motivos de nuestra elección.
    - *Capítulo 4. Prácticas docentes.* Se explican las motivaciones que nos han llevado al desarrollo de este Proyecto Fin de Carrera.
- **PARTE III: Descripción del trabajo realizado.** En esta parte se presenta y describe el trabajo realizado en este Proyecto Fin de Carrera. Se divide en los siguientes capítulos:
    - *Capítulo 5. Desarrollo del entorno virtual* En este capítulo se describe como se adaptó el entorno real que tenemos en los laboratorios docentes al entorno virtual.
    - *Capítulo 6. Desarrollo del entorno de autocorrección* Se explica el proceso de diseño del entorno de corrección, los módulos por los cuales esta compuesta la herramienta de corrección y como interactúan entre ellos.

  - *Capítulo 7. Diseño y desarrollo de la herramienta de generación de plantillas: definición de escenarios y de pruebas de corrección* Se explica el proceso de diseño de la herramienta que permite, de forma fácil, definir los escenarios para las prácticas y las pruebas de corrección que se aplicarán sobre ellos.
  - *Capítulo 8. Validación y pruebas realizadas* Por una parte explicaremos las pruebas realizadas para garantizar la corrección de trabajo y por otro lado se explicará la experiencia real y el resultado del uso de estas herramientas en las prácticas realizadas por los alumnos

- **PARTE IV: Conclusiones y trabajos futuros.** En esta parte se presentan las conclusiones más importantes al trabajo realizado y también se introducen las futura lineas de investigación que surgen tras la realización de este proyecto.

- **PARTE V: Apéndices.** En esta parte se incluyen los siguientes apéndices:

  - *Apéndice A. DTDs y XMLs.*
  - *Apéndice B. Instalación software.*
  - *Apéndice C. Manual de Usuario.*
  - *Apéndice D. Presupuesto.*

Capítulo 2

ANTECEDENTES HARDWARE: ROUTERS

2.1. INTRODUCCIÓN

Los conocimientos que se adquieren al trabajar con máquinas reales ayudan a afianzar los conocimientos adquiridos teóricamente, ya que por un lado permite entender los conceptos relacionados con la redes de ordenadores y por otro lado permite el uso de máquinas reales que nos acercan a los equipos que podemos encontrarnos en cualquier organización en el mundo real.
Para llevar a cabo estos objetivos, en las prácticas docentes del Departamento de Ingeniería de Telemática de la Universidad Carlos III de Madrid (UC3M), se han utilizado dos tipos de routers diferentes:

- **TELDAT C y CBRA**. Son los primeros que se utilizaron en la Universidad. Se trata de unos routers de propósito general con amplio espectro de aplicación: entornos personales, PYME y corporativos. Gracias a su versatilidad, son adecuados para una gran variedad de escenarios IP: desde proporcionar acceso simultáneo a Internet a los usuarios de una red privada de área local hasta la adaptación a redes de teleproceso y soporte SNA, pasando por el soporte de terminales de puntos de venta (Datáfono). Cubre las necesidades de acceso por ADSL, RDSI y línea serie (conexión a un módem telefónico externo, Frame Relay, X.25, PPP, etc.). Estos equipos requieren una alta configurabilidad para poder adaptarse a las distintas necesidades y entornos, y por tanto, la cantidad de parámetros de configuración disponibles es muy extensa 2.1. Para proceder a la configuración de estos equipo se ofrecen distintas vías de configuración: a través de consola local (mediante el puerto serie) y a través de TELNET. Cuando se eligieron estos routers se tuvieron en cuenta el tipo de interfaces que ofrecía:

  - Interfaz Básico RDSI
  - Interfaz Ethernet
  - Interfaz puerto serie

  En esos momentos RDSI era una tecnología muy utilizada en las empresas. Además, sólo tenían un interfaz de Ethernet (que limitaban el tamaño de los escenarios a configurar) y no disponían de ningún interfaz que permitiera comunicación wifi. Por estos motivos se migró a otros routers que sí cubrieran estas necesidades.

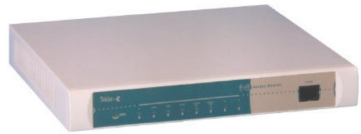


Figura 2.1: Router Teldat C

- **LINKSYS**. Son los routers que actualmente se están utilizando en la Universidad, concretamente los routers Linksys WRT54GSv4 y Linksys WRT54GLv1, ambos modelos son equivalentes. 2.2 Hay una serie de motivaciones que han llevado a elegir estos equipos para el Departamento:
  - La configuración de los routers Linksys se realiza a través de comandos, los cuales, tienen una sintaxis prácticamente idéntica a la que se utiliza en CISCO IOS, sistema que llevan equipos de gran rendimiento y que se suelen utilizar en empresas de cierta envergadura.
  - Son unos equipos bastante baratos que permiten acercarnos a los routers de gran envergadura de forma económica.
  - El tipo de interfaces que tienen estos routers son de tecnologías de red ampliamente utilizadas y versátiles,lo que aumenta la flexibilidad en las prácticas.
  - El número de interfaces de red es superior al de los routers anteriores (Teldat), lo que nos permite crear configuraciones de red más complejas y completas. Los routers Linksys poseen 5 interfaces Ethernet y uno wlan, mientras que los routers Teldat tienen un interfaz Ethernet, uno RDSI y uno serie.

Figura 2.2: Router Linksys WRT54GS

2.2. ROUTERS LINKSYS WRT54GSv4 y WRT54GLv1

El router Linksys WRT54GSv4 y WRT54GLv1 es un router orientado al entorno doméstico y de pequeña empresa, que dispone de 5 interfaces Ethernet y una interfaz inalámbrica (IEEE 802.11b/g), que satisface las necesidades que actualmente hay en la Universidad 2.3. Ambos modelos son equivalentes aunque en la actualidad sólo se comercializa el modelo WRT54GLv1. A partir de ahora, nos referiremos a estos routers como “routers Linksys”.

Figura 2.3: Interfaces del router Linksys WRT54GS y WRT54GLv1

Como se aprecia en la figura 2.3, las interfaces Ethernet son las bocas que se ven en la parte trasera del router. La primera boca (de izquierda a derecha), etiquetada como 'Internet' en el router, internamente recibe el nombre de eth0.0. A continuación hay 4 bocas más, etiquetadas del 1 al 4 en el router, que internamente reciben el nombre de eth0.1, eth0.2, eth0.3 y eth0.4 respectivamente. Realmente se dispone de una interfaz física Ethernet (eth0), conectada a un switch con capacidades de VLAN, lo que permite crear hasta 5 interfaces distintas. Por último, el router dispone de una interfaz inalámbrica que recibe, internamente, el nombre de wlan0.

El router ejecuta Linux como Sistema Operativo, lo cual permite instalar en él multitud de aplicaciones y utilidades diferentes, en concreto se ha instalado el demonio de encaminamiento Quagga [5], que permite configurar diversas características del router y experimentar aspectos de configuración, de forma muy similar a como se hace con routers comerciales de mayor envergadura. Existen 2 maneras complementarias de acceder al router:

- A través de TELNET, accediendo a una consola del router, que recibe el nombre de CLI (Command Line Interface). Mediante dicha consola se pueden configurar multitud de parámetros de red relacionados con los interfaces y los protocolos.
- A través de SSH, accediendo a una consola Linux. Dicha consola proporciona los comandos básicos que se pueden encontrar en un sistema Linux y que se emplea básicamente para la configuración de la interfaz inalámbrica.

Los estándares soportados por los routers Linksys WRT54GSv4 y WRT54GLv1 son: IEEE 802.3 (Ethernet), IEEE 802.3u (Fast Ethernet), IEEE 802.11b (WLAN, con velocidad máxima de transmisión a 11 Mbit/s) y IEEE 802.11g (WLAN, con velocidad máxima de transmisión a 54 Mbit/s).

Los protocolos de red y enrutamiento que pueden ser configurados a través de cualquiera de los modos de acceso al router son:

- Configuración de interfaces IP (Ethernet y wlan)
- Enrutamiento estático IP
- RIP (Router Information Protocol)
- OSPF (Open Shortest Path First)

- BGP (Border Gateway Protocol)

Como se puede apreciar, es posible hacer realizar configuraciones de todos los protocolos de encaminamiento que más se utilizan actualmente, por lo que estos routers se han convertido en una valiosa herramienta docente. Prueba de esto es que el departamento dispone de aproximadamente 170 routers Linksys, los cuales están siendo utilizados en, al menos, 7 asignaturas de diferentes titulaciones para la realización de prácticas, además de ser una herramienta para la realización de numerosos proyectos, trabajos fin de máster y proyectos de investigación.

2.3. FIRMWARE

Los routers Linksys están pensados para el uso doméstico, vienen de fabrica con una distribución de Linux que permite entre otras cosas ofrecer al usuario un interfaz web para la configuración de un conjunto limitado de parámetros de red. Esta utilidad que el router ofrece no es interesante para nuestro cometido, ya que lo que buscamos es que se realicen configuraciones de red a través de comandos, imitando así el tipo de configuración que utilizan los routers de alto rendimiento. Por esta y otras razones, se decidió personalizar el software que traían aplicando un firmware basado en una distribución Linux de código abierto (OpenWRT WhiteRussian RC6 en las primeras versiones del firmware instalado y Backfire 10.03 en las últimas versiones de firmware instalado en los routers) ligeramente modificada para que ésta se ajustase a los requisitos docentes que se deseaban:

- Se instalaron varios demonios de enrutamiento basados en Quagga suite para permitir la posibilidad de configurar distintos protocolos de routing con el objetivo de que la interfaz de configuración fuese lo más similar posible a la interfaz de configuración de CISCO IOS. Los protocolos de enrutamiento de los que hablamos son:
  - RIP (Router Information Protocol)
  - OSPF (Open Shortest Path First)
  - BGP (Border Gateway Protocol)
- Se proporcionó que las interfaces fueran accesible a través de SSH. De esta forma se permite la conexión a una consola de Linux en el router y se proporciona así un mecanismo flexible y potente para la configuración de la interfaz inalámbrica, dado que a través de Quagga suite no se puede modificar la configuración de WLAN.
- Se modificó lo necesario para asegurarse de que los routers no estuvieran expuestos a una mala configuración, ya sea accidental o intencionada. De esta forma los usuarios sólo tienen permisos para realizar configuraciones de red, sin permisos para que puedan cambiar otros parámetros de configuración.
- Otro cambio que se hizo fue el no permitir que el router almacenase la configuración que se hubiera realizado. Cada vez que el router arranca, se carga una configuración por defecto. De esta forma se garantiza que los routers se comporten de la misma manera después de ser reiniciados, evitando arrastrar errores de configuración anteriores.

Capítulo 3

ANTECEDENTES SOFTWARE: MAQUINAS VIRTUALES

3.1. INTRODUCCIÓN A LAS MÁQUINAS VIRTUALES Y LOS ENTORNOS DE VIRTUALIZACIÓN

Una máquina virtual o un entorno de virtualización es un software que permite tener un duplicado eficiente y aislado de una máquina física creando una capa de abstracción. Los usuarios y aplicaciones dentro de la máquina virtual se relacionan con la capa de abstracción, no con la maquina real. Podríamos decir que es como tener uno o varios ordenadores dentro de otro ordenador 3.1:.
Al sistema operativo de la máquina real se le denomina “host” o “hypervisor”, mientras que al sistema operativo de la máquina virtual se le denomina “guest”.

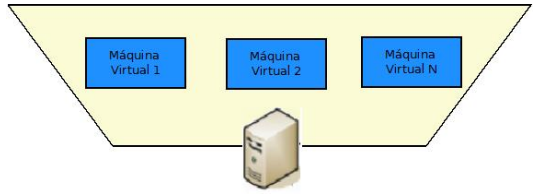


Figura 3.1: Concepto máquina virtual

En los últimos años se ha puesto muy de moda el uso de esta tecnología, pero las máquinas virtuales no son algo nuevo, existen desde principios de los años 70 y se han utilizado para cubrir múltiples necesidades. Uno de los usos más extendidos es para poder tener distintos sistemas operativos funcionando a la vez en un mismo equipo. Otra de las utilidades que ofrecen es poder ejecutar aplicaciones hechas para una plataforma sobre otra plataforma diferente, un ejemplo de esto lo vemos con la máquina virtual de Java (JVM). Además hace posible tener un entorno de trabajo controlado, evitando poner en riesgo nuestro equipo cuando hacemos ciertas operaciones que pueden ser comprometidas para nuestra máquina real, gracias a que los procesos que se ejecutan están limitados por los recursos y abstracciones proporcionados por las máquinas virtuales (estos procesos no pueden escaparse del “ordenador virtual”), lo cual nos puede resultar muy útil cuando tenemos que probar aplicaciones de docencia, aplicaciones en desarrollo o para ejecutar aquellas aplicaciones cuya pro-

cedencia es de sitios no confiables. Otra posible utilidad de las máquinas virtuales es utilizarlas como señuelos (honeypots), para evitar que sean atacadas máquinas con información relevante dentro de una organización. Una utilidad muy en auge en los últimos años es el uso de servidores virtuales, hay muchas empresas de hosting que están ofreciendo este tipo de servicio.

Hay un gran número de ventajas que se derivan de la idea que persiguen las máquina virtuales y de las posibles utilidades que se les pueden dar:

- Aislamiento: las máquinas virtuales son totalmente independientes, entre sí y entre el host y los guests. Por tanto un fallo en una aplicación o en una máquina virtual afectará únicamente a esa máquina virtual. El resto de máquinas virtuales y el host seguirán funcionando normalmente.
- Seguridad: cada máquina tiene un acceso privilegiado (root o administrador) independiente. Por tanto, un ataque de seguridad en una máquina virtual sólo afectará a esa máquina.
- Flexibilidad: podemos crear las máquinas virtuales con las características de CPU, memoria, disco y red que necesitemos, sin necesidad de adquirir un ordenador con esas características.
- Agilidad: la creación de una máquina virtual es un proceso muy rápido. Por tanto, si necesitamos una máquina nueva la podremos tener casi al instante, sin pasar por el proceso de compra, configuración, etc.
- Portabilidad: toda la configuración de una máquina virtual reside en uno o varios ficheros. Esto hace que sea muy fácil clonar o transportar la máquina virtual a otro lugar físico, simplemente copiando y moviendo dichos ficheros que encapsulan la máquina virtual.
- Recuperación rápida en caso de fallo: si se dispone de una copia de los ficheros de configuración de la máquina virtual, en caso de desastre la recuperación será muy rápida, simplemente habría que arrancar la máquina virtual con los ficheros de configuración guardados. No es necesario reinstalar, recuperar backups y/o aplicar otros procedimientos más largos que se utilizan en las máquinas físicas.

Existen diferentes tipos de máquinas virtuales o entornos de virtualización. A continuación mostramos una clasificación de éstos acompañada de algunos ejemplos, que pueden ayudarnos a entender esta división entre máquinas virtuales:

- **Virtualización completa.** No sólo emulan memoria, disco y otros dispositivos, sino también la CPU, lo que hace que sean especialmente lentos. Permiten que el sistema operativo de la máquina virtual y el sistema operativo del host trabajen en arquitecturas diferentes. Un ejemplo de este tipo de virtualización es QEMU [6], un emulador de procesadores basado en la traducción dinámica de binarios.
- **Virtualización.** Se emula memoria virtual, disco y dispositivos. No emulan CPU, por lo tanto, guest y host tienen que utilizar la misma arquitectura. VirtualBox [8] y VMWare [9] son ejemplos muy extendidos de este tipo de máquinas virtuales.
- **Máquinas virtuales cooperativas.** Este tipo de virtualización se da cuando no se emula hardware del host. Los sistemas operativos del host y el guest acceden en paralelo al hardware de la máquina. Un ejemplo de este tipo de virtualización es UML (User Mode Linux) [7], que es un núcleo de Linux ligeramente modificado para que pueda ejecutarse como un proceso de usuario de otro núcleo Linux.

- **Paravirtualización.** Es similar a la virtualización, pero exige una versión del guest ligeramente modificada que hace que el rendimiento sea mejor que en el caso de la virtualización. Xen [12] pertenece a este tipo de máquinas virtuales.

- **Virtualización nativa.** Es una emulación completa, pero realizada por la CPU con lo que el rendimiento es próximo al del sistema nativo. Esta es una técnica en desarrollo que es soporta por Xen.

3.2. ENTORNO VNUML

VNUML (Virtual Network User Mode Linux) [10] es un entorno de virtualización de propósito general desarrollado para la simulación de redes virtuales en equipos físicos. Está basada en software libre, por lo que cualquier usuario puede modificarlo para su uso y para ayudar a aumentar las funcionalidades proporcionadas por la herramienta. Este entorno está basado en UML, por lo que se encuadra dentro de la virtualización de tipo "máquinas virtuales cooperativas". El objetivo que persigue VNUML es ayudar a probar servicios de red y aplicaciones que impliquen la interactuación de varios nodos diferentes, interconectados entre ellos, pero dentro una misma máquina Linux.

VNUML se compone de dos módulos principales:

- **Lenguaje VNUML**. A través de este lenguaje, basado en XML [1] , se definen los escenarios de red sobre los que se desea trabajar. Cada escenario queda descrito en un único fichero XML. En estos ficheros se definen los nodos que forman la red y sus características (nombre del nodo, número de interfaces, direcciones IP, etc). Los escenarios definidos mediantes este lenguaje deben cumplir el DTD [2] que proporciona VNUML, que se puede consultar en el apéndice A y en el cual se especifican los parámetros necesarios para definir un escenario.

- **Interprete del lenguaje VNUML**[3]. Este componente se encarga de construir los escenarios definidos previamente y gestionar el entorno de virtualización ocultando toda la complejidad al usuario.

Figura 3.2: Interacción de los dos módulos principales de VNUML

[1] XML:eXternded Markup Languaje es un metalenguaje extensible de etiquetas que fue desarrollado por Word Wide Web Consortium y permite la organización y el etiquetado de documentos.

[2] DTD: Document Type Definition. La función de este fichero es la de describir el formato de datos, con el objetivo de usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD.

[3] Este intérprete, desarrollado en perl, se llama vnumlparser.pl. Es un comando que al ejecutarse es capaz de entender el XML que define el escenario y generar un entorno virtual de acuerdo con lo especificado en el fichero.

La interacción de estos dos módulos permite la creación de escenarios compuestos por un número determinado de nodos que simulan equipos GNU/Linux y se interconectan entre ellos formando distintas redes 3.2.

Hay tres modos o formas de utilizar VNUML en función de los privilegios que quieran usar 3.3:

1. **Sin privilegios de administrador sobre el entorno**. Únicamente se podrán ejecutar los escenarios y configurar los nodos a través del XML. Una vez lanzado el entorno, no se tiene acceso a las máquinas virtuales.

2. **Con algunos privilegios de administrador sobre en entorno**. Con este nivel de privilegios se permite tener acceso a la red para la gestión de las máquinas virtuales. El host "anfitrión" puede conectarse al resto de nodos que forman el escenario de red y a la vez, estos nodos, pueden utilizar al "anfitrión" como un router NAT [4] para salir a Internet.

3. **Con todos los privilegios de administrador en el entorno**. Permite realizar la gestión de la red y la configuración de los nodos a través de acceso directo a las máquinas virtuales.

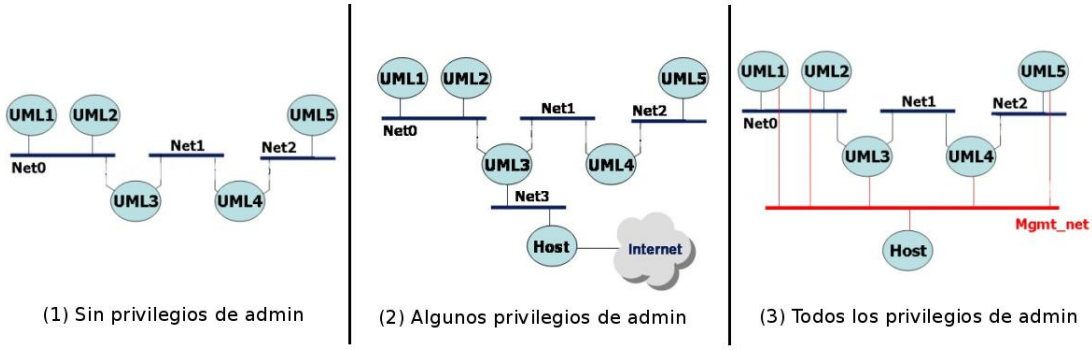


Figura 3.3: Formas de funcionamiento de VNUML según el nivel de privilegios

Un ciclo de uso típico de VNUML está formado por las siguientes fases:

1. Fase de diseño. En esta primera parte, el usuario debe diseñar el escenario que se va a simular. Hay varios aspectos importantes que se deben elegir: el número de máquinas virtuales, la topología de la red (número de interfaces por nodo y como se interconectan entre sí), que procesos ejecutará cada máquina virtual,etc.

2. Fase de implementación. Una vez diseñado el escenario con el que se desea trabajar, el usuario debe implementarlo escribiendo el fichero XML que lo describa, siguiendo las reglas semánticas definidas en el DTD de VNUML.

[4] NAT: Network Address Translation. Es un estándar desarrollado por la IETF para la utilización de una o más direcciones IP para conectar varias computadoras a una red (especialmente Internet). Cada computadora tiene una dirección IP distinta (que no es válida para la red a la que quieren conectarse) y el equipo que hace de NAT proporciona una IP válida de forma temporal para la máquina que desea conectarse.

3. Fase de ejecución. En esta fase el usuario podrá lanzar el escenario y hacer las pruebas que considere necesarias sobre éste.

4. Fase destrucción del entorno. Una vez se haya terminado de trabajar con la herramienta, se cerrará el entorno, destruyendo las máquinas virtuales creadas y las configuración y pruebas realizadas sobre éstas.

Capítulo 4

PRACTICAS DOCENTES

4.1. INTRODUCCIÓN

En muchas de las asignaturas que se imparten en el Departamento de Ingeniería Telemática las clases magistrales son necesarias, pero insuficientes para tener un completo aprendizaje, el cual se consigue a través del apoyo que ofrecen las clases prácticas para la compresión de las materias. En una clase magistral de este tipo de asignaturas se suelen explicar los conceptos, los protocolos de red o comunicación y las características de estos. Además son clases útiles para realizar ejercicios teóricos que ayuden a fijar ideas que, en su gran mayoría, son muy abstractas. Por otro lado, en las clases prácticas podemos usar los conceptos aprendidos y aplicarlos en casos propuestos que se asemejan a posibles situaciones de la vida real, por ejemplo en un entorno empresarial.

En nuestro caso, hemos puesto especial interés en las prácticas que se realizan en los Laboratorios de Ingeniería Telemática y en los que se utilizan como material docente los routers Linksys. En estas clases, los alumnos tienen la posibilidad de trabajar con máquinas reales, enfrentándose a problemas que no son planteados en las clases teóricas, como por ejemplo la interconexión de cables o el uso de comandos específicos para realizar las configuraciones propuestas. Gracias a estas sesiones los alumnos adquieren la capacidad de enfrentarse a un entorno físico real que les proporciona un valor añadido a los conocimientos que adquieran tras haber cursado la asignatura.

Principalmente se realizan dos tipos de prácticas que necesitan el uso de los routers Linksys: prácticas de configuración de red y prácticas de desarrollo de protocolos. En las prácticas de configuración de red se propone un escenario al alumno, el cuál tiene que montarlo utilizando el material que ofrece el Departamento (routers y PCs) y configurarlo según las especificaciones que se le indiquen en el enunciado. En las prácticas de desarrollo de protocolos se propone a los alumnos implementar un protocolo de red existente, el cual una vez desarrollado tiene que poder comunicarse con el resto de routers y equipos que utilizan el protocolo oficial.

Como se puede apreciar, según el tipo de prácticas que se realizan en el laboratorio, éstas no podrían existir sin una parte teórica previa, al igual que las clases magistrales por si solas son insuficientes y alejan al alumno de la aplicación al mundo real de los conceptos aprendidos. Combinando las dos técnicas, el aprendizaje del alumno será más fácil de asimilar y más completo.

4.2. METODOLOGÍA EMPLEADA EN LAS PRÁCTICAS

El ciclo típico de "vida" de una práctica impartida por el Departamento de Ingeniería Telemáticas en sus laboratorios es:

- Preparación de la práctica por parte del profesor. En esta fase, se plantean los ejercicios que se van a proponer a los alumnos. Deben ser ejercicios de una dificultad adecuada para el nivel de los alumnos, dependiendo del curso en el que se encuentren. Además hay que estimar el tiempo que se pueda tardar en resolver dicha práctica ya que el material que se ofrece se utilizará únicamente en el tiempo estimado para la realización de ésta. Por último, hay que decidir el peso en nota que tendrá dicha práctica.
- Preparación de la documentación necesaria para la realización de la práctica. El profesor facilitará a los alumnos los manuales del material empleado en la sesión.
- Realización del trabajo previo a la sesión por parte de los alumnos. En algunas ocasiones esta fase puede consistir simplemente en la lectura de los ejercicios propuestos y un esquema de cómo resolverlos. En otras ocasiones, se puede requerir una serie de ejercicios teóricos necesarios para la resolución de la práctica.
- Ejecución de la práctica en los laboratorios durante las sesiones.
- Entrega de la práctica y corrección. Esta fase puede consistir simplemente en una corrección "in-situ" del profesor, aplicando una batería de pruebas específicas sobre el escenario creado por los alumnos o puede implicar la entrega de una memoria explicando lo que se ha hecho y/o pidiendo ciertos resultados.

Como se puede observar la realización de estas prácticas implican un esfuerzo extra tanto del profesorado como de los alumnos. Por un lado los profesores tienen que preparar las prácticas y la documentación necesaria para realizarlas, además de tener que corregir los ejercicios propuestos. Por otro lado, los alumnos no sólo tienen que realizar las prácticas en las sesiones de los laboratorios, sino que deben tenerlas previamente preparadas para ejecutarlas en el tiempo establecido para la sesión.

Observando la metodología utilizada actualmente y conociendo la experiencia de la realización de estas prácticas, se observan dos problemas principales:

1. El tiempo de realización de las prácticas está determinado y es fijo, lo cual, limita la dificultad de los ejercicios que se pueden proponer, además de obligar al alumnado a restringir el tiempo de uso de los routers. Ciertas circunstancias pueden hacer que el tiempo establecido para realizar las prácticas sea escaso, no por mala planificación, sino por imprevistos que pueden surgir fuera del alcance del profesor. Lo ideal sería proveer de una herramienta que se pudiera utilizar el tiempo que se deseara para la realización de los ejercicios propuestos.
2. La carga de trabajo para el profesor es elevada. Sería deseable tener un método fiable de corrección de prácticas que ayudase al profesor en esta tarea.

Capítulo 5

DESARROLLO DEL ENTORNO VIRTUAL

5.1. INTRODUCCIÓN

El material docente del que dispone el Departamento debe ser usado por todos los alumnos que cursen las asignaturas cuyo programa incluya prácticas que requieran su uso. Por este motivo, este material debe ser cuidado, y el uso del mismo tiene que estar limitado a unas horas concretas, bajo cierta supervisión.

La motivación que nos ha llevado a crear este entorno es ofrecer a los alumnos un entorno virtual, al que puedan acceder en cualquier momento, que permitiese realizar las prácticas de las asignaturas de redes sin precisar del entorno físico real.

En este capítulo vamos a explicar como hemos adaptado el entorno del cual partíamos como base, VNUML, a nuestro entorno real, el laboratorio de prácticas. Primero vamos a detallar cuales son los objetivos que queremos lograr con el desarrollo de este entorno. Posteriormente, se explicarán los módulos de VNUML que hemos tenido que modificar para realizar esta adaptación, las capas de abstracción que hemos creado para simular el entorno real y la elección del modo de ejecución que hemos hecho.

5.2. OBJETIVOS

Los objetivos o requisitos que nos hemos fijado a la hora de diseñar un entorno virtual que simule el escenario físico de los Laboratorios de Ingeniería Telemática son los siguientes:

- Debe ser un entorno que simule perfectamente el material del Laboratorio real. La idea es trabajar con un entorno virtual pero simulando el entorno real que, por motivos de tiempo y/o seguridad, no siempre es accesible para el alumnado.
- El aprendizaje para utilizar el entorno virtual ha de ser mínimo, sino nulo.
- Queremos poder utilizarlo de forma remota, sin necesidad de estar físicamente en los Laboratorios.

- El entorno virtual tiene que poder utilizarse tanto para las prácticas de configuración de red como para las prácticas de desarrollo de protocolos.
- Un requisito es que los "PCs virtualizados" se han de poder intercambiar ficheros entre el entorno real y el virtual, para facilitar tareas como: la captura de tráfico o la prueba de prácticas de desarrollo.
- Cuando se realizan prácticas en el entorno real, es muy frecuente tener un router, ya configurado por el profesor, que se utiliza de forma auxiliar para probar la configuración global. Sería deseable que en el entorno virtual pudiésemos arrancar un escenario en el cual hubiese un equipo con una configuración de red determinada, definida para ese escenario y precargada.

5.3. DISEÑO E IMPLEMENTACIÓN DEL ENTORNO

5.3.1. Módulos VNUML

De los dos módulos principales que componen VNUML (el lenguaje y el intérprete de éste), el módulo que, por su naturaleza, es más configurable y podemos utilizar para adaptar nuestro entorno real al virtual es el del lenguaje VNUML.

Como se ha explicado en el capítulo 3, este lenguaje está compuesto por dos submódulos: el DTD propio de la herramienta y los XMLs que tienen que seguir las reglas definidas en el DTD para generar los escenarios. Es importante destacar que el intérprete del lenguaje sólo va a ser capaz de entender los parámetros que han sido definidos en el DTD propio de la herramienta, con lo que nos impide el poder modificarlo, por lo tanto, el submódulo configurable que tenemos para adaptar el entorno son los XMLs que definen los escenarios de red.

Dada la importancia que tiene para nosotros este submódulo, vamos a detenernos en explicar los parámetros que se definen en este fichero y como nos van a influir en la adaptación del entorno real al virtual. Los principales tags[1] que definen el XML y por lo tanto el escenario de red se pueden apreciar en la figura 5.1 y son los siguientes:

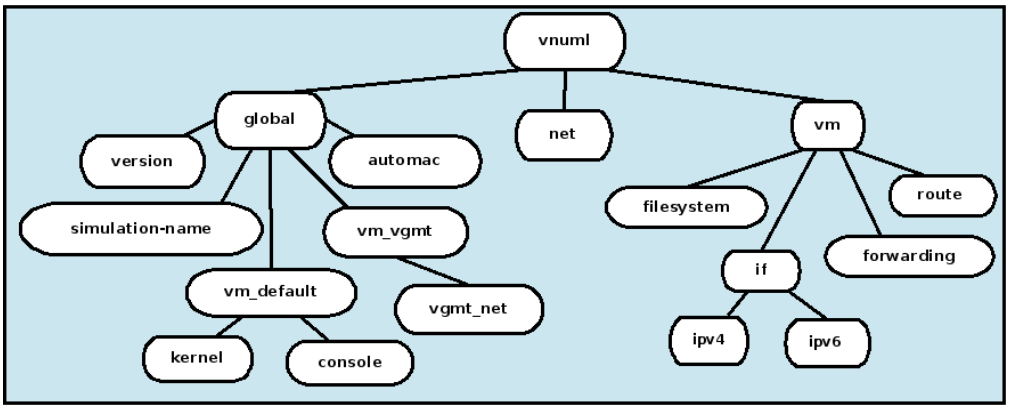


Figura 5.1: Elementos que componen el XML que describe el escenario de red

[1] tag: etiqueta o marca que delimita una región en un texto

- **vnuml**: es el nodo padre de todos. Los hijos de este nodo definen completamente el escenario de red.
- **global**: en este nodo se recogen todos los datos generales del escenario, así como aquellos datos que son aplicables de forma común para el resto de nodos que forman el XML. En la tabla 5.1, se describen cada uno de los elementos hijos de este nodo.

Elementos hijos del tag global	
Elemento	**Descripción**
version	Versión de vnuml que vamos a utilizar (1.0).
simulation-name	Nombre que vamos a darle a la simulación.
vm_default	Define los atributos comunes que se aplicarán a todas las máquinas virtuales. En concreto se definen el kernel y el tipo de terminal con el que accederemos a cada máquina virtual.
vm_vgmt	Sirve para configurar la red de mantenimiento que se genera si ejecutamos la simulación con todos los privilegios de administrador. Se configuran los siguiente valores: la red, la máscara de red, IP del host anfitrión, etc. Todos estos parámetros se explicarán más en detalle en la sección 5.2.3.
automac	Indica que las mac de las máquinas virtuales se van a generar de forma automática, sin ser especificadas por nosotros.

Tabla 5.1: Descripción de los elementos hijos del nodo global

- **vm**: es el nodo que define un objeto de tipo máquina virtual. Se debe declarar un objeto "vm" por cada una de las máquinas virtuales que van a formar el escenario de red. Los hijos de este nodo son los parámetros que sirven para definir los atributos configurables de cada una de las máquinas virtuales y las características de éstas (nombre de la máquina, número de interfaces de red, etc) sin llegar a bajo nivel (CPU, memoria). En la tabla 5.2, se describen cada uno de los elementos hijos de este nodo y sus atributos.

Elementos hijos del tag vm	
Elemento	**Descripción**
filesystem	Imagen que va a cargar la máquina virtual al arrancar. Hay que especificar la ruta completa
if	Define cada uno de los interfaces de red que tendrá la máquina virtual. Los atributos que definen el interfaz son un identificador numérico, el nombre de la red a la cual está conectado y la IP asignada al mismo.
route	Sirve para configurar una ruta específica en la máquina virtual. Sus atributos son: el tipo de protocolo (ipv4 o ipv6), el gateway[2], y la red destino.
forwarding	Se utiliza para especificar si el nodo tendrá capacidad para reenviar paquetes.

Tabla 5.2: Descripción de los elementos hijos del nodo vm

- **net**: es el nodo que define un objeto de tipo red, el cual vamos a utilizar para interconectar las máquinas virtuales entre sí. Cada máquina virtual debe estar conectada a al menos una red. La

[2] gateway: se refiere a la IP del router que pertenece a una red y da acceso al exterior de ésta.

persona que genera el XML es quien debe decidir que topología lógica desea dar a el escenario, teniendo en cuenta que para crear una topología que tenga sentido, el número de redes debe ser menor del número de nodos "vm" definidos menos uno (Nnet <Nvm-1). En la tabla 5.3, se describen cada uno de los atributos de este nodo.

Atributos del tag net	
Atributo	**Descripción**
name	Asigna un nombre a la red. Tiene que ser único, ya que sirve para identificar la red en todo el fichero.
mode	Sirve para definir el modo de red (bridge o swich).

Tabla 5.3: Descripción de los atributos del nodo net

Tras conocer bien los tags principales con los que se definen las simulaciones, vamos a identificar qué parte del entorno real se va a representar con cada tag y qué restricciones vamos a fijar para reflejar el entorno físico del laboratorio:

- Dentro del tag "global" vamos a definir los datos la práctica o simulación, de esta forma podremos tener un conjunto de escenario bien diferenciados entre sí. Las restricciones definidas sobre este nodo son las siguientes:
  - El nombre simulación corresponderá con el nombre de la práctica.
  - El tipo de terminal con el cual accederemos directamente a cada máquina virtual será de tipo Xterm[2], ya que es de los terminales menos pesados.
  - El kernel que vamos a utilizar para cada máquina virtual es linux-2.6.18.1-bb2-xt-4m
- Utilizaremos el tag "vm" para simular los equipos hardware que tenemos en el laboratorio. Las restricciones definidas sobre este nodo son:
  - El número de interfaces que se van a definir para cada máquina virtual puede ser mayor que uno.
  - El número de IPs que se pueden asignar a cada interfaz no puede ser nunca superior a uno[3].
  - Como máximo, aparecerá una ruta de encaminamiento configurada en la máquina virtual, que corresponderá con la IP del router por defecto que hay en los laboratorios. De esta forma, el alumno verá la tabla de rutas de la máquina virtual y la tabla de rutas de la máquina física del laboratorio idénticas para el interfaz de prácticas. No se desea que al arrancar la máquina virtual tenga más configuraciones de encaminamiento, ya que esta labor la deberá realizar el alumno para resolver el ejercicio propuesto en cada simulación.
- Con los tags "net" vamos a simular cada una de las redes que se configuran en el laboratorio para interconectar los equipos hardware del escenario, es decir, cada nodo de tipo "net" simulará un cable de red. Solo se ha definido una restricción para este nodo:
  - El modo de red que vamos a utilizar es siempre switch, ya que queremos operar con direcciones ip, no a bajo nivel.

[2] Xterm: emulador de terminal para el sistema de ventanas gráficas.

[3] Este requisito ha sido definido para que al arrancar el entorno sólo aparezca una IP por interfaz de red, tal y como ocurre en el escenario real. El entorno virtual nos permite configurar más de una IP en un mismo interfaz sin problemas.

5.3.2. Capa de Abstracción: Hosts y Routers

Como hemos podido observar en la sección anterior, los nodos de tipo “vm” nos dan libertad para crear todas las máquinas virtuales que deseemos, con lo que nuestro entorno de red puede ser bastante grande. También hemos visto que estos tags nos permiten crear nodos que representen equipos PC y nodos capaces de encaminar paquetes, gracias al tag “forwarding”. Éstos últimos pueden representar un router simple, en el cual pudiésemos configurar una tabla de encaminamiento, pero no pueden llegar a ofrecer las posibilidades y servicios que nos ofrecen los routers que se utilizan en el Laboratorio.

Otro problema que nos surge, si utilizamos directamente el entorno que nos ofrece VNUML, es que el alumno percibiría grandes diferencias entre el entorno real y el virtual ya que la interfaz y los comandos que podría utilizar en uno y otro no serían los mismos. Los routers de Laboratorio tienen instalado un software específico, que permite configuraciones de red complejas a través de una serie de comandos, sin embargo en el software que proporcionan las máquinas virtuales de VNUML estos comandos no existen.

Para solucionar estos problemas hemos creado una capa de abstracción entre el nodo “vm”, definido en el XML, y la propia máquina virtual que se ofrece al alumno al arrancar el entorno. Como se puede ver en la figura 5.2, esta capa de abstracción se coloca justo por encima del concepto de máquina virtual definido en el XML y esta compuesto por dos elementos:

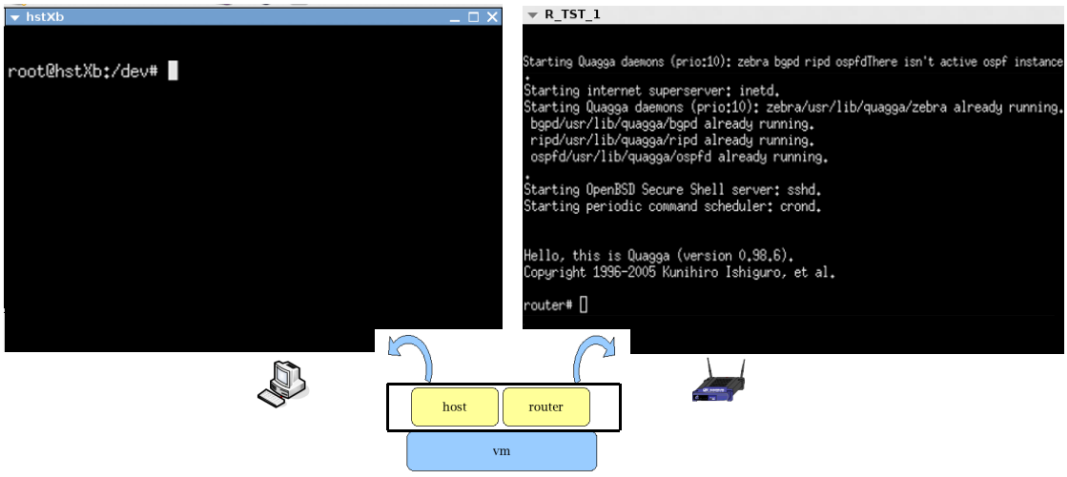


Figura 5.2: Capa intermedia que diferencia tipos de nodos en el escenario virtual

- **Host.** Son máquinas virtuales que representan los PC del Laboratorio. Para conseguirlo, hemos creado una imagen con una distribución muy básica de Linux Debian (que es también la distribución utilizada en los laboratorios docentes del Departamento de Ing. Telemática), para que la máquina virtual sea ligera, y la cual contiene todos los paquetes necesarios para realizar las prácticas de configuración de redes y para probar en el entorno virtual pequeños programas escritos en C y desarrollados en el entorno físico. Para poder utilizar en el entorno virtual programas desarrollados en el entorno físico, VNUML proporciona un espacio de intercambio entre el anfitrión y la máquina virtual. Este espacio de intercambio permite tener ficheros alojados en un determinado directorio y que éstos puedan ser accesibles desde la máquina virtual.

El interfaz de red de los "host" que van a configurar los alumnos será el mismo que el que se configura en la aulas, el "eth1". Además, para conseguir agilizar el acceso a cada máquina virtual, hemos eliminado la petición de login en cada nodo tipo host, de esta forma, cada vez que se arranca un entorno con N máquinas virtuales accedemos directamente a la máquina sin necesidad de hacer login.

- **Router.** Son máquinas virtuales que representan los routers Linksys que tenemos en el Laboratorio, y que tienen el mismo firmware instalado. Para conseguirlo, primero se creó una imagen con una distribución muy básica de Linux Debian, la misma que se utilizó para crear el software de los nodos host. Encima de este sistema operativo, se instalaron los demonios de enrutamiento Quagga que tienen el firmware de los routers físicos. Se configuró para que al arrancar el sistema operativo cargase el software del router. Estas máquinas virtuales tienen que tener 6 interfaces, cinco que representan las conexiones cableadas que se pueden configurar y uno que representa el interfaz wlan[4]. De esta forma, el alumno percibirá la misma interfaz que percibe en el entorno real.

El tamaño de las imágenes se ha reducido al máximo para que las máquinas virtuales ocupasen poco espacio y fueran muy ligeras, pero dejando cierto espacio libre para permitir que se puedan pasar ficheros a las máquinas virtuales si fuese necesario. En concreto, para los hosts, la imagen ocupa 300 MegaBytes (de los cuales, 50 MegaBytes están libres) y para los routers la imagen tiene un tamaño de 260 MegaBytes (con 30 Megabytes libres). Se puede consultar el proceso de creación de las imágenes, paso a paso, en el apéndice B.

Ahora que hemos conseguido diferenciar entre elementos de tipo host y elementos de tipo router, debemos saber como incluir esta diferencia en el XML para que se materialice esta capa de abstracción. Para hacer esto, basta con identificar en el tag "filesystem", hijo de "vm" el path completo del sistema que se quiera cargar para cada máquina virtual. Esto se puede ver claramente en el ejemplo de escenario simple que se incluye en el apéndice A.

Otro de nuestros objetivos era conseguir que, en un escenario, cada router pudiera arrancar con una configuración determinada y diferente de los demás. Para ello, dentro de la imagen del router se ha creado un script llamado "/etc/init.d/uc3m" que se ejecuta en el arranque de la máquina. Este script comprueba cuál es el nombre de la máquina virtual y según el resultado que obtenga, carga una configuración determinada o carga la configuración por defecto. Los distintos tipos de configuración disponibles se encuentran en el directorio "/etc/quagga/uc3m/". Si se desea añadir una configuración nueva o se desea eliminar una ya existente, simplemente habría que modificar el script de arranque (/etc/init.d/uc3m) y añadir o eliminar un fichero con la configuración en el directorio /etc/quagga/uc3m/. Obsérvese que este cambio implica una modificación sobre la imagen de los routers. En el apéndice B, sección B.1.4, se incluye una guía en la que se explica como se modifican las imágenes.

5.3.3. Modo de ejecución del entorno

En el capítulo 3 pudimos ver que hay tres posibles modos de ejecución de este entorno, cada uno con unas características especiales. Con el objetivo de utilizar una herramienta lo más homogénea posible, hemos decidido elegir la opción que mejor se ajusta a nuestras necesidades, descartando el

[4]De la interfaz wlan no se emulan los aspectos inalámbricos del interfaz, simplemente aparece como un interfaz IP más, el cuál tiene que se configurado con los comandos propios para la configuración de redes wlan.

resto.

El primer modo, sin privilegios de administrador sobre el entorno, no nos permite realizar configuraciones sobre las máquinas virtuales que componen nuestro escenario de red, por lo que fue descartado rápidamente.

El segundo modo, con algunos privilegios de administrador, nos permite acceder a las máquinas que forman parte del escenario y además incluyen al PC anfitrión como un nodo más del escenario, permitiendo incluso, que éste pueda encaminar paquetes y que el entorno tenga salida a Internet a través de este PC. Aunque esta opción es mucho mejor que la anterior, también fue descartada, ya que de este modo ofrecemos al alumno un entorno menos controlado. El PC anfitrión sería un ordenador del laboratorio y sobre este equipo no podemos aplicar las mismas restricciones que en el entorno virtual. Otro motivo por el cual este modo fue descartado es por que permite salida a Internet, lo cual no nos interesa porque nosotros queremos realizar configuraciones concretas que no tienen que interactuar con Internet. Por último, para acceder a cada máquina virtual tenemos que hacerlo a través de conexiones ssh y/o telnet, lo cual hace más tediosa la tarea de configuración de escenarios grandes.

El tercer modo, con privilegios de administrador para el entorno, nos permite acceder a cada máquina virtual de forma directa, ya que nos aparece un terminal de configuración por cada nodo que pertenece al escenario, como se muestra en la figura 5.3. Además conseguimos generar un entorno cerrado, en el cual el PC anfitrión no juega ningún papel dentro del escenario.

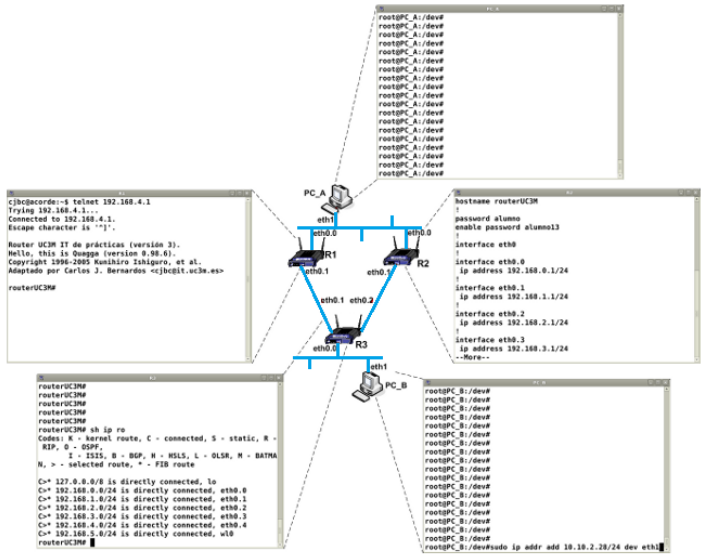


Figura 5.3: Ejemplo de un escenario de red. Cada terminal representa una máquina virtual y ofrece un acceso directo a cada máquina que compone el escenario.

Por otro lado, nos brinda la opción de poder tener una red privada entre nuestro PC anfitrión y el resto de máquinas virtuales. Esta red privada no interfiere en la configuración de los escenarios y es transparente para el alumno ya que es gestionada por el propio entorno virtual. La red de mantenimiento privada nos va a ser de gran utilidad para la autocorrección de escenarios que se explicará en

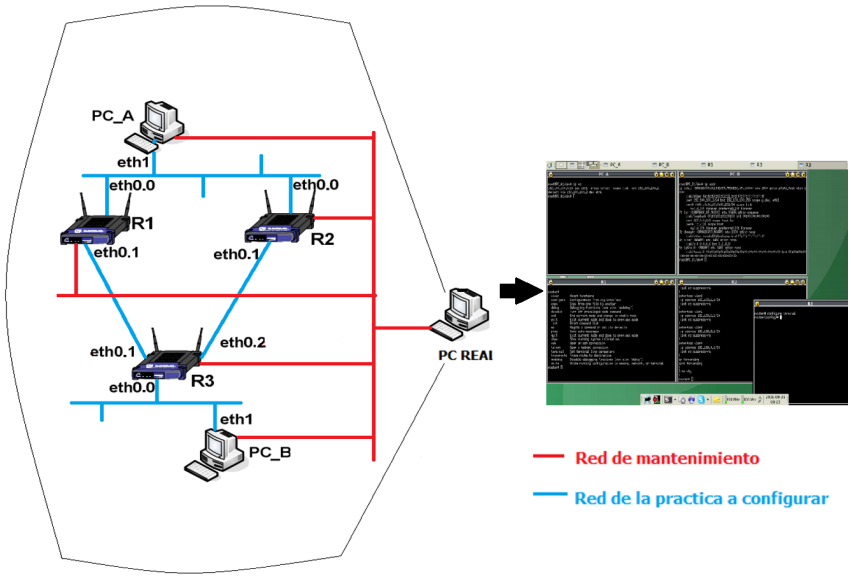


Figura 5.4: Visualización de la red de mantenimiento y la red del escenario.

los próximos capítulos, ya que permitirá obtener información de configuración de las máquinas virtuales mediante de scripts que se conectan a nodos del escenario. En la figura 5.4 se puede ver como conviven la red de mantenimiento y la red del escenario en el entorno.

Para configurar un escenario con todos los privilegios de administrador y poder generar la red de mantenimiento entre el PC anfitrión y el resto del escenario, es necesario configurar el tag "vm_mgmt", hijo de "global". Los atributos que describen este tag se pueden visualizar en la tabla 5.4, y el elemento hijo de este tag está definido en la tabla 5.5.

Atributos del tag vm_mgmt	
Atributo	**Descripción**
type	Define el tipo de red que vamos a crear, puede tomar dos valores: net y private. Con el tipo net se genera una red entre el PC y el entorno virtual (el anfitrión solo necesita un interfaz para conectarse a todos los nodos del entorno virtual), mientras que con el tipo private, se crean N redes privadas que conecta de forma "punto a punto" cada nodo del escenario con el PC (el anfitrión necesita un interfaz por cada nodo del escenario para conectarse a él). En nuestro caso, siempre utilizaremos el tipo net por ser más simple, en cuanto a gestión se refiere, si el escenario es muy grande.
network	Sirve para configurar que red vamos a elegir para que sea la red de mantenimiento.
mask	Definimos la máscara para la red de mantenimiento.

Tabla 5.4: Descripción de los atributos del nodo vm_mgmt

Elementos hijos del tag vm_mgmt	
Elemento	**Descripción**
mgmt_net	Define los atributos propios del interfaz de red que se genera en el PC anfitrión. En este tag se especifica el socket que se utilizará (atributo sock), la ip que tendrá este interfaz (atributo hostip)y el nombre con el cual podemos identificar al interfaz en el equipo (atributo autoconfigure).

Tabla 5.5: Descripción de los elementos hijos del nodo vm_mgmt

5.4. CONCLUSIONES

En este capítulo hemos explicado cual ha sido el proceso de adaptación del entorno real de prácticas del Laboratorio de Ingeniería Telemática al entorno virtual que se quería ofrecer a los alumnos. Se han estudiado las características propias de cada entorno y las posibilidades que ofrece la herramienta virtual para que los escenarios fuesen los más similares posible para que el alumno no note apenas diferencias entre un entorno y otro.

Hemos definido un conjunto de reglas sobre los ficheros de configuración de la herramienta virtual, los escenarios XML, para que el entorno virtual y entorno real sean muy parecidos.

Hemos creado una capa de abstracción intermedia que permite al alumno diferenciar rápidamente entre nodos virtuales que representan un host del laboratorio y nodos virtuales que representan un router. Esto lo hemos conseguido adaptando el software que tiene cada uno en el entorno real y creando las imágenes que cargarán las máquinas virtuales. Además estas imágenes tienen espacio libre para poder cargar en ellas ficheros externos si fuese necesario y/o permitirnos instalar software adicional, sin tener que crear una imagen nueva.

Hemos seleccionado el modo de ejecución que utilizaremos para realizar las prácticas: con todos los privilegios de administración. Gracias al modo de ejecución elegido podemos acceder a todos los nodos del escenario virtual de forma sencilla y rápida. Además, la red de mantenimiento que se crea con este modo nos permite sentar las bases de la autocorrección de escenarios, que se explican el los siguientes capítulos, permitiendo que el host anfitrión se conecte a cada máquina y obtenga la información de red que necesita para saber si la configuración de alumno es correcta o no.

Dado que la herramienta de la que partíamos como base se ejecuta sobre un entorno Linux, hemos podido adaptarla e instalarla en los Laboratorios y ofrecerla al alumnado. Además, al ser un entorno ligero, por estar basado en UML, es fácil exportar el entorno virtual y permitir utilizarlo remotamente, sin estar en los laboratorios del Departamento.

Gracias a todas las adaptaciones realizadas, hemos conseguido un entorno virtual para realizar las prácticas de configuración de redes y de desarrollo de protocolos de distintas asignaturas. El aprendizaje que requiere el uso del entorno virtual es mínimo, ya que se ha intentado adaptar al máximo el entorno virtual al real: las máquinas virtuales simulan a la perfección los PCs y routers del laboratorio y los comandos que se utilizan y pruebas que se aplican sobre el escenario son los mismo que si trabajásemos con los equipos físicos.

Hay que resaltar que aunque hemos intentado imitar al máximo el escenario real, hay ciertas cosas que no vamos a poder simular, como por ejemplo, ver que ocurre cuando en un escenario de red un cable se desconecta o ayudar a un alumno a practicar con el cableado físico. Estas limitaciones son conocidas, pero no son muy importantes, ya que en ningún caso queremos sustituir el entorno físico (donde estas limitaciones no existen) por el entorno virtual. Lo que buscabamos era poder ofrecer a los alumnos un entorno que les permitiera preparar las prácticas y ofrecer a los profesores una herramienta para proponer y preparar ejercicios complementarios.

Por último, hemos implementado un método para que las máquinas virtuales de tipo router carguen una configuración determinada en el inicio del escenario. Esto nos va a permitir utilizar un router determinado para hacer pruebas o para utilizarlo como router auxiliar.

Capítulo 6

DESARROLLO DEL ENTORNO AUTOCORRECCIÓN

6.1. INTRODUCCIÓN

Hasta ahora las prácticas de configuración de redes y routers se realizaban en el Laboratorio de Ingeniería Telemática con el material docente del que se disponía (PCs y routers Linksys), a una hora concreta, con un tiempo de ejecución limitado y bajo la supervisión de uno o varios profesores que resolvían a los alumnos las dudas que pudieran tener y corregían las configuraciones realizadas. Este método de trabajo es bueno, ya que el alumno tiene que enfrentarse a equipos reales para resolver los problemas que se le proponen, además tienen que hacerlo en un tiempo determinado lo que obliga al alumno a tener los conceptos sobre los que se trabaja claros y cierta agilidad en la ejecución de la configuración. Sin embargo, esta modelo de prácticas limita la complejidad de los ejercicios que se pueden proponer, ya que hay que estimarlos para que sea posible realizarlos en el tiempo establecido; impide al alumno poder practicar con los routers siempre que desee y adquirir la agilidad deseada y limita el tiempo de aprendizaje a las horas de sesiones de prácticas.

En la sección anterior hemos presentado un entorno de prácticas no presencial que permite a los alumnos realizar los ejercicios propuestos cuando quieran y las veces que quieran. Lo deseable sería tener una herramienta que nos permitiese corregir estos ejercicios sin necesidad de tener presente la figura del profesor. Gracias a esta herramienta de corrección ofreceríamos una doble funcionalidad: por un lado, los alumnos podrían corregir los ejercicios que realizasen y retroalimentarse de los errores que cometieran favoreciendo el auto aprendizaje, sin necesidad de que el profesor estuviera presente y por otro lado, se liberaría al profesor de una carga importante de trabajo, ofreciendo una herramienta que corrigiese los ejercicios que se proponen y realizando, sobre las configuraciones, las pruebas que el profesor considerara oportunas.

En esta sección vamos a presentar la herramienta de corrección que hemos diseñado e implementado para conseguir comprobar el correcto funcionamiento de las configuraciones de red realizadas con el entorno virtual. En primer lugar vamos a explicar detalladamente cuales son los objetivos que buscamos y los requisitos que nos hemos marcado inicialmente a la hora de diseñar la herramienta. Una vez claros los objetivos, haremos una introducción al entorno de corrección dando una visión global de esta herramienta para a continuación, explicar los bloques que la componen: el módulo de configuración y el módulo de corrección de los escenarios de red. Por último explicaremos cómo se

genera y se presenta la calificación obtenida en una práctica.

6.2. OBJETIVOS

Los objetivos o requisitos que nos hemos fijado a la hora de diseñar esta herramienta que corrige los escenarios de red configurados por los alumnos son los siguientes:

- Debe ser flexible para permitir la corrección de escenarios muy diferentes.
- Buscamos que la configuración de la corrección sea sencilla de hacer, es decir, debe ser simple para el profesor definir las pruebas a realizar, el peso a asignar a cada una de ellas, etc.
- Es importante que el uso de esta herramienta sea sencillo, ya que no queremos que el alumno necesite un proceso de aprendizaje previo para el uso de la herramienta.
- Desde el principio, uno de nuestros objetivos ha sido conseguir un entorno virtual que sea prácticamente igual que el entorno real. En la parte de corrección queremos seguir manteniendo esta filosofía, para ello queremos que las pruebas de corrección que se realicen en el entorno virtual sean las mismas que las que se realizan en el entorno real. Las pruebas que se quieren definir son las siguientes:
  - **Prueba de ping.** Lanzar una prueba de conectividad entre dos nodos y comprobar si obtenemos respuesta.
  - **Prueba de traceroute.** Comprobar que el camino seguido por un paquete para llegar de un nodo a otro es el esperado.
  - **Prueba de BGP**[1]. Comprobar la configuración de rutas realizadas con el protocolo BGP.
  - **Obtener datos propios de la configuración.** Incluimos en este grupo la tablas de rutas o las IPs configuradas en cada interfaz.
  - **Desconectar/conectar interfaces.** En el laboratorio es posible deshabilitar una interfaz y ver como se comporta el escenario, comprobando así la robustez de la configuración realizada. Esto es especialmente interesante cuando se han configurado protocolos de encaminamiento que son tolerantes a fallos y requieren un periodo de tiempo de convergencia (RIP[2], OSPF[3] y BGP) [4]. En el entorno virtual nos gustaría poder simular esta acción (deshabilitar un interfaz y especificar un tiempo de espera para que el algoritmo converja).
- Las pruebas de corrección deben poder ejecutarse tanto para IPv4 como para IPv6.

[1] BGP: Boder Gateway Protocol. Es un protocolo de red mediante el cual se intercambia información de encaminamiento entre sistemas autónomos, donde un sistema autónomo es un grupo de redes IP que poseen una política de rutas propia e independiente.

[2] RIP: Routing Information Protocol. Es un protocolo de enrutamiento que calcula la ruta más corta hacia una red de destino utilizando el algoritmo vector-distancia. La distancia o métrica, la determina el número de saltos que se producen, de router en router, hasta alcanzar la red de destino. Para obterner esta métrica, utiliza la información que le ofrece su nodo vecino más cercano.

[3] OSPF: Open Shortest Path First. Es un protocolo de enrutamiento que utiliza el algoritmo de Dijkstra (estado-enlace) para calcular la ruta más corta posible. Cada enlace tiene un coste determinado y la ruta más corta será aquella que tenga un coste menor.

[4] Cada uno de estos protocolos de encaminamiento implementa un algoritmo determinado, que ante fallos en la red es capaz de recalcular las tablas de rutas para evitar que algunos nodos queden sin conexión. El tiempo que tarda en calcular estas rutas depende del número de nodos de tipo router que existan en el escenario.

- La corrección de los escenarios de red puede ser utilizada con dos objetivos: calificar de forma oficial una práctica propuesta por el profesor o informar al alumno del resultado de su configuración (sin llegar a enviar al profesor ningún dato para calificar el ejercicio).
- Dado que queremos poder calificar a los alumnos mediante esta herramienta, debemos diseñar un módulo de seguridad que impida la copia o manipulación de la corrección haciendo este método de evaluación seguro y robusto.

6.3. ENTORNO DE AUTOCORRECCIÓN

El objetivo de está sección es dar una visión global de la herramienta de corrección, presentando los módulos que la componen y como interactúan entre ellos. En la figura 6.1 podemos ver como se estructura la herramienta de corrección, los distintos módulos que la componen y como interactúa con el entorno virtual.

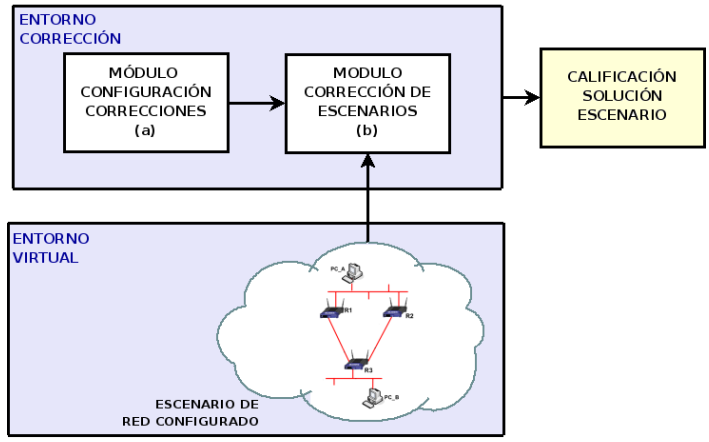


Figura 6.1: Esquema global del entorno de corrección.

Como podemos observar, la herramienta de corrección se compone de dos módulos importantes: el módulo de configuración (a), en el cual se define la corrección para cada escenario y el módulo de corrección (b), que es encargado de ejecutar las prueba definidas en el módulo de corrección e interpretar la salida que éstas nos proporcionen. Una vez realizadas las pruebas, la herramienta calificará el ejercicio siguiendo los criterios que el profesor haya fijado.

6.4. MÓDULO DE CONFIGURACIÓN: DISEÑO E IMPLEMENTACIÓN

Este es el primer módulo que compone el entorno de corrección. Uno de los objetivos que nos habíamos marcado era tener un entorno de corrección flexible, válido para cualquier tipo de escenario que se desee plantear en el entorno virtual. Por este motivo hemos decidido crear un módulo de configuración que permita elegir unos parámetros determinados para cada escenario y que permita

realizar un conjunto de pruebas diferentes y variables sobre la configuración realizada por el alumno.

Este módulo sirve para generar una corrección adaptada a cada ejercicio propuesto para realizar con el laboratorio virtual, el cual, como hemos visto también tiene un módulo de configuración: el fichero XML que define el escenario. Para seguir con la lógica del laboratorio virtual, hemos decidido que nuestro módulo de configuración se base también en un fichero XML que defina los parámetros de la corrección. Por lo tanto podemos decir que nuestro módulo de configuración está formado por dos elementos: el DTD que define las reglas y restricciones propias de los ficheros de corrección y el XML que siguiendo las reglas establecidas en el DTD define una corrección determinada para una configuración.

A continuación se va a explicar en detalle el formato del fichero XML de corrección de escenarios, el cual sigue el DTD definido por nosotros cuyo nombre es "autocorreccion.dtd" y que se puede consultar en el apéndice A. En la figura 6.2 se pueden ver las restricciones definidas sobre el XML y cómo se relacionan los nodos entre sí.

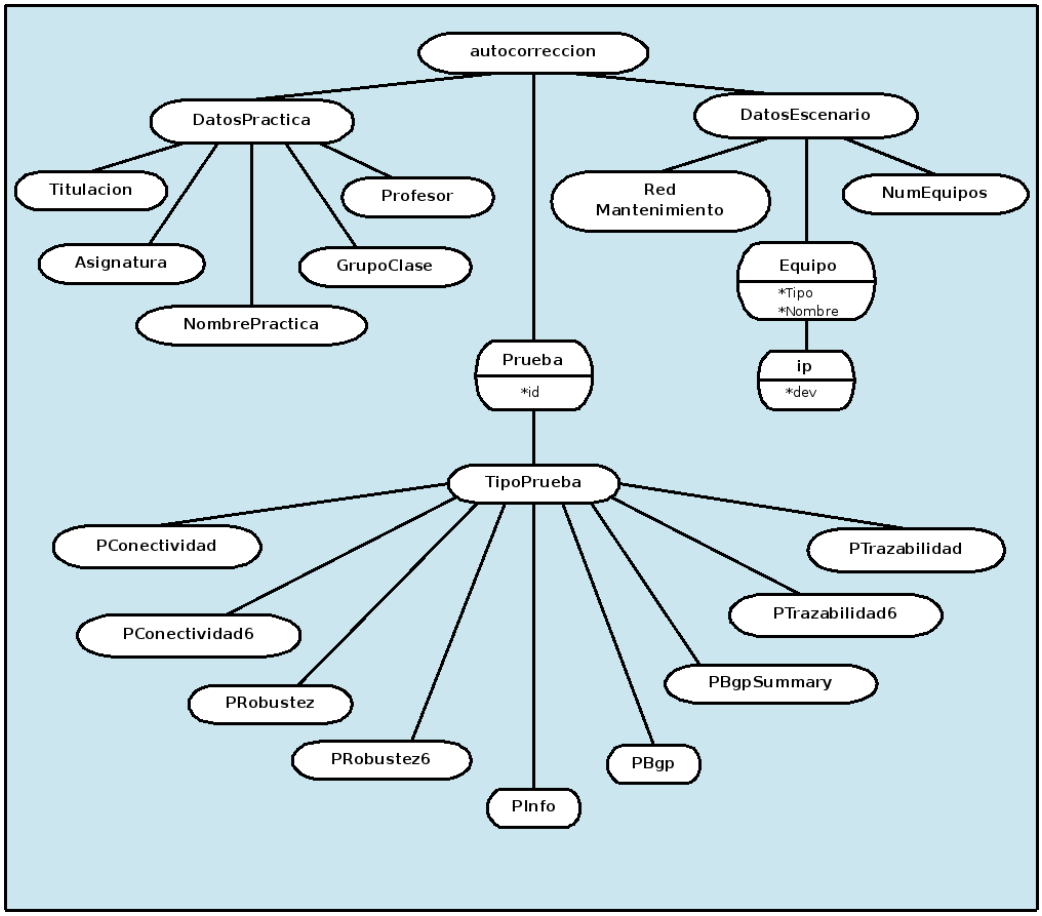


Figura 6.2: Elementos que componen el XML que describe la corrección de prácticas.

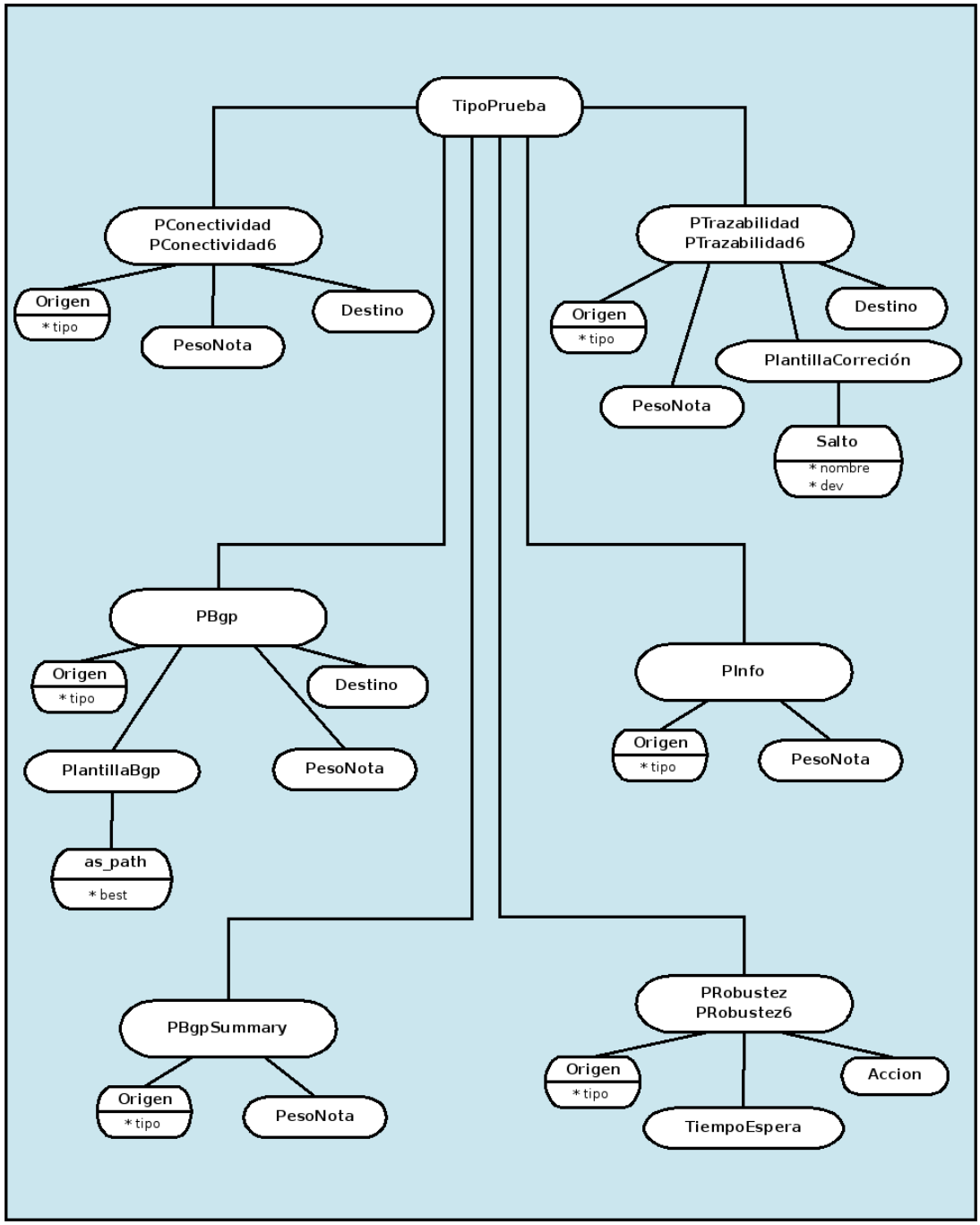


Figura 6.3: Elementos hijos del nodo “TipoPrueba” que definen las pruebas de corrección

- **autocorreccion**: es el nodo padre de todos. Los hijos de este nodo definen completamente la corrección de un escenario de red configurado con el entorno virtual.
- **DatosPractica**: este nodo recoge todos los datos generales de la práctica y permite que se pueda identificar fácilmente para que práctica fue creada una corrección determinada. Los hijos de este nodo se describen en la tabla 6.1.

Elementos hijos del tag DatosPractica	
Elemento	**Descripción**
Titulacion	Sirve para indicar a que titulación pertenece una corrección concreta.
Asignatura	Indica para que asignatura a sido creada la corrección.
NombrePractica	Especifica el nombre de la práctica que se va a corregir siguiendo las pautas que indica este fichero de corrección.
GrupoClase	Sirve para indicar el grupo de clase al cual se va a entregar este fichero de corrección.
Profesor	Especifica el login del profesor encargado de la asignatura indicada, que se imparte para una titulación concreta y cuyo número de grupo de clase ha sido especificado antes.

Tabla 6.1: Descripción de los elementos hijos del nodo DatosPractica

- **DatosEscenario**: en este nodo se definen datos generales del escenario que se va a corregir. Los hijos de este nodo se definen en la tabla 6.2.

Elementos hijos del tag DatosEscenario	
Elemento	**Descripción**
RedMantenimiento	Este parámetro está muy ligado al módulo del entorno virtual descrito en el capítulo anterior. Con este parámetro indicamos cual es la red definida en el entorno virtual que permite interconectar directamente nuestro equipo "anfitrión" con cada una de las máquina virtuales de la simulación.
NumEquipos	Indica el número de equipos que forman la simulación del escenario de red, sin tener en cuenta el "anfitrión".
Equipo	Define una máquina virtual de la simulación de red. En el XML habrá tantos nodo de este tipo como NumEquipos hayan sido especificados. Este nodo tiene dos atributos: tipo y nombre. El atributo "tipo" define si una máquina virtual es un nodo de tipo router o de tipo host. Los valores que puede tomar este atributo son: "H" para identificar a los host y "R" para identificar a los routers. El atributo "nombre" indica el hostname la máquina virtual.
ip	Identifica la ip de un equipo. Este nodo tiene un atributo: dev. La ip que vamos a definir en este nodo se configurará en el interfaz que el "dev" indique.

Tabla 6.2: Descripción de los elementos hijos del nodo DatosEscenario

- **Prueba:** este es el nodo más importante, gracias a él podemos definir las pruebas de corrección que se harán sobre el escenario. Debemos crear un elemento de este tipo por cada prueba de corrección que deseemos realizar sobre la configuración de red. Los hijos de este nodo, cuya estructura se pueden visualizar de forma gráfica en la imagen 6.3, definen las características propias de cada prueba y están descritos por los elementos que aparecen en las tablas 6.3.

Elementos hijos del nodo Prueba	
Elemento	**Descripción**
TipoPrueba	Indica el tipo de prueba que se va a aplicar en la simulación de red para corregir la configuración realizada por el alumno. Debe haber tantos nodos de este tipo como pruebas se deseen aplicar al escenario de red. Los hijos de este nodo son: PConectividad, PConectividad6, PTrazabilidad, PTrazabilidad6, PBgp, PBgpSummary, PInfo, PRobustez y PRobustez6. Estos nodos se definen en la tabla 6.4 y a su vez, los hijos de los tipos de pruebas están explicados en detalle en la tabla 6.5.

Tabla 6.3: Descripción del hijo del nodo Prueba

Elementos hijos del nodo TipoPrueba	
Elemento	**Descripción**
PConectividad / Pconectividad6	Define una prueba de ping[2] entre dos máquinas, tanto para el protocolo IPv4 como para IPv6.
PTrazabilidad / PTrazabilidad6	Define una prueba de traceroute[3] entre dos máquinas, tanto para el protocolo IPv4 como para IPv6.
PBgp	Define una prueba que comprueba la configuración de rutas realizadas con el protocolo BGP analizando los as_path[4].
PBgpSummary	Define una prueba que muestra las características generales de una tabla BGP configurada.
PInfo	Define una prueba que devuelve las características generales de configuración, tanto en nodos de tipo “host” como en nodos de tipo “router”, mostrando las IPs de cada interfaz y las tablas de rutas configuradas.
PRobustez / PRobustez6	Define poder realizar la acción de deshabilitar y habilitar un interfaz para ver como se comporta el escenario configurado.

Tabla 6.4: Descripción de los hijos del nodo TipoPrueba

[2]Ping: es el acrónimo de Packet INternet Groper. Es una utilidad de diagnóstico en redes de computadores que comprueba el estado de la conexión del host local con uno o varios equipos remotos.

[3]Traceroute: es una herramienta de diagnóstico que permite seguir la pista de los paquetes viendo los nodos por los que pasa hasta llegar al destino.

[4]as_path: es uno de los atributos definidos para el protocolo de red BGP. La red está dividida en AS (sistemas autónomos que son un conjunto de equipos que forman una red y siguen una política de rutas propia e independiente) y para llegar de un AS a otro suele haber más de una camino posible. Un as_path es la lista de AS que definen un camino posible, desde un nodo origen a un nodo destino.

Elementos hijos de los nodos que describen los diferentes tipos de pruebas	
Elemento	**Descripción**
Origen	Define el nodo origen desde el cual se va a lanzar una prueba corrección determinada. Tiene un atributo: tipo, el cual puede tomar los valores "H" o "R". Necesitamos estos datos para conectarnos a la máquina origen a través de la red de mantenimiento para lanzar desde ahí la prueba.
Destino	Define el nodo destino contra el que se lanza la prueba de corrección.
PlantillaCorreccion	Para un prueba de tipo traceroute, define cual es la salida correcta. Si el resultado de la prueba no coincide con el de esta plantilla se considera que el escenario no está bien configurado.
PlantillaBgp	Para una prueba de tipo BGP, define cuales son los as_path configurados y cual de ellos es el seleccionado como "best".
PesoNota	Sirve para especificar el valor que se le desea dar a una prueba de corrección, permitiendo así tener una corrección flexible capaz de valorar con mayor peso las pruebas cuya salida positiva impliquen mayor dificultad en la configuración.
Acción	Es un elemento definido sólo para las pruebas de robustez. En este nodo se especifica si la acción que se desea realizar es deshabilitar un interfaz de red: "Down" o si se desea habilitarlo:"Up".
TiempoEspera	Es un elemento definido sólo para las pruebas de robustez. Al deshabilitar un interfaz sobre un nodo, dependiendo de lo complejo que sea el escenario de red, éste necesita un tiempo mayor o menor para reconfigurar sus rutas en caso de que se hallan programado con tolerancia a fallos. En este nodo se define este tiempo de espera en milisegundos.

Tabla 6.5: Descripción de los nodos que describen los diferentes tipos de pruebas

6.5. MÓDULO DE AUTOCORRECCIÓN: DISEÑO E IMPLEMENTACIÓN

Este es el módulo más importante de este entorno ya que es el que realiza la corrección de las configuraciones que los alumnos han aplicado al escenario de red virtual. Es un módulo compuesto por varios submódulos que se relacionan entre sí para obtener la corrección a una práctica o ejercicio propuesto. En la figura 6.4 podemos observar cuales son estos submódulos y como se interrelacionan entre ellos.

El programa principal recibe el fichero de corrección como argumento (a), lo parsea[5] y obtiene los datos necesarios para generar las pruebas de corrección (b). Estas pruebas, son scripts que se lanzan sobre el entorno virtual ya configurado por el alumno (c) y devuelven una salida determinada (d). La salida obtenida tras la realización de las pruebas es analizada por el programa principal (e) que se encargará de generar un fichero en el que se incluye la nota global del ejercicio de configuración y la nota de cada prueba aplicada sobre el escenario (f). Este fichero es también de tipo XML para

[5]Parsear: se define así el proceso de analizar una secuencia de símbolos a fin de determinar su estructura gramatical respecto a una gramática formal dada. El parseo transforma una entrada de texto en una estructura de datos apropiada para ser procesada.

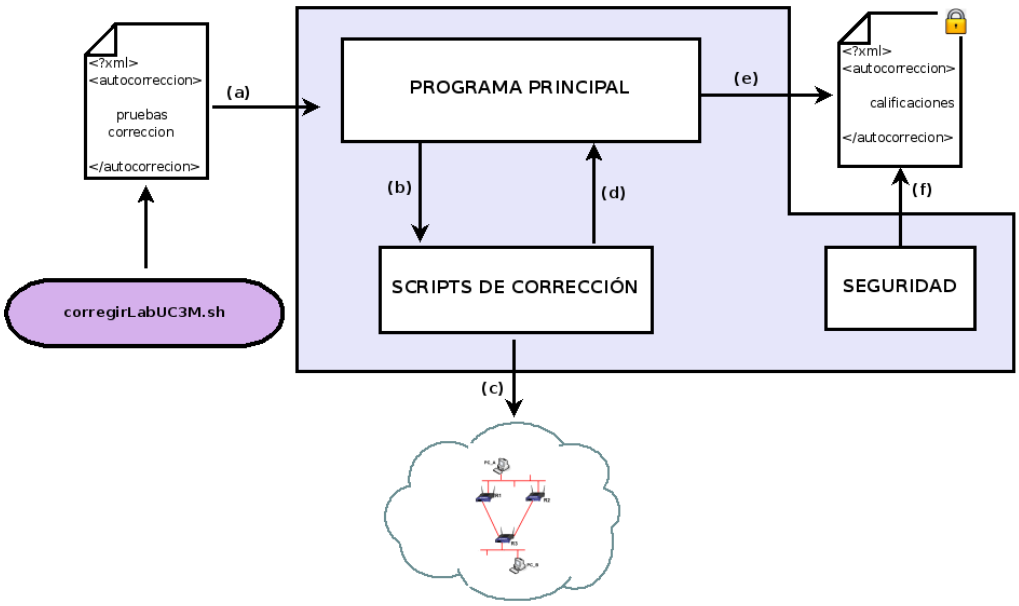


Figura 6.4: Esquema global del módulo de corrección.

seguir con la lógica de ficheros que hemos llevado durante todo el desarrollo del proyecto. Por último, a este fichero XML se le aplicarán ciertas medidas de seguridad para evitar la copia de prácticas y asegurarnos de que la práctica ha sido entregada por un usuario determinado. Todo el proceso de corrección lo iniciar el scripts "DFcorregirLabUC3M.sh".

A continuación se explicarán en detalle como hemos implementado cada uno de estos submódulos.

6.5.1. Programa principal: Autocorreccion.jar

Este programa es el corazón de la corrección de los escenarios, ya que centraliza todas las acciones necesarias para la corrección de ejercicios de configuración de red que se realizan con el entorno virtual. Ha sido desarrollado en Java, utilizando las librerías y paquetes que nos ofrece esta tecnología para tratar ficheros XML mediante JDOM[6]. A continuación, en el figura 6.5, vamos a presentar el diagrama de clases Java que forman la aplicación para entender como se relacionan entre ellas.

Dada la complejidad de esta parte del entorno de corrección y para que sea más fácil la compresión de la misma, vamos a dividir en bloques el programa principal, según la funcionalidad de las partes, el programa principal. De este modo hemos obtenido los siguientes bloques:

- Bloque parseador XML
- Bloque pruebas
- Bloque generador XML

[6]JDOM: es un API para leer, crear y manipular documentos XML de una manera sencilla. Java+XML=JDOM

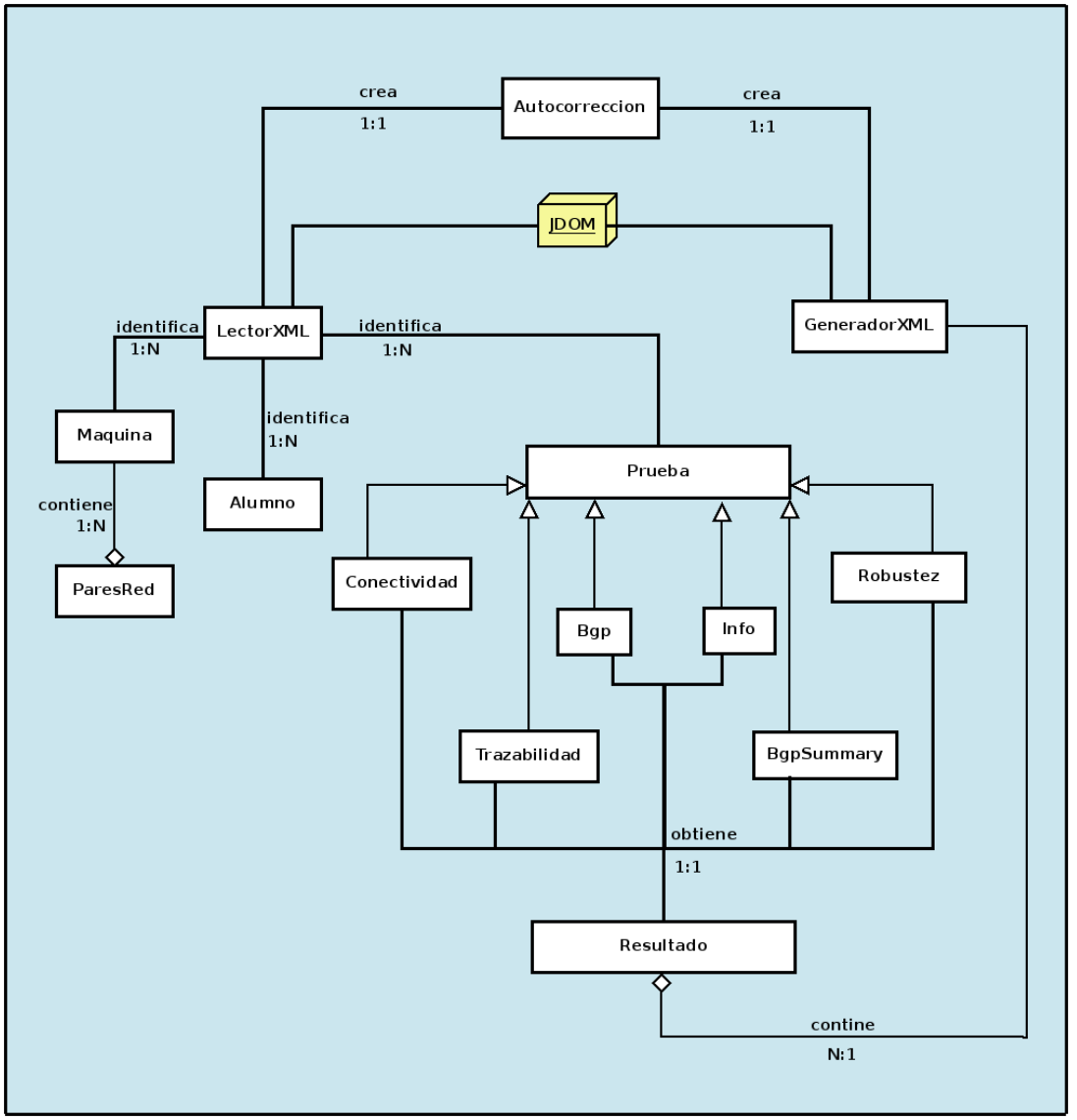


Figura 6.5: Diagrama de las clases Java que forman el programa principal y las relaciones existentes entre ellas.

6.5.1.1. Bloque parseador XML

Este bloque es que el encargado de recibir el fichero XML con los datos de la corrección y obtener de él los datos del escenario y los datos que se utilizarán para generar las pruebas de corrección especificadas. Las clases forman parte de este bloque son las que aparecen en la figura 6.6. Para simplificar, hemos decidido no mostrar en el diagrama los métodos get y set[7].

[7]Los métodos get y set de Java son funciones que permiten obtener y asignar datos a los atributos, ya que éstos son inaccesibles directamente para proporcionar el encapsulamiento de los atributos.

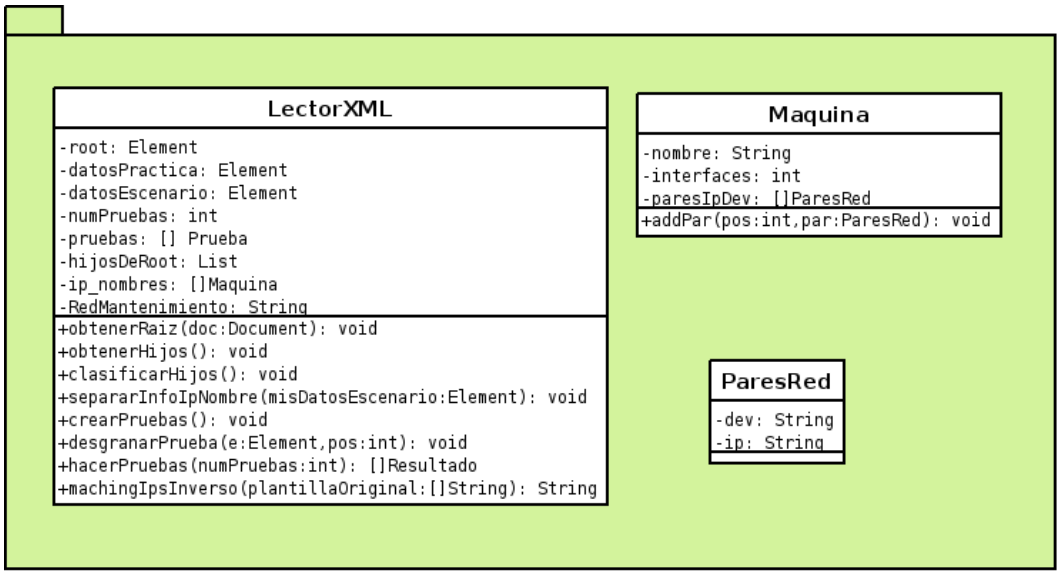


Figura 6.6: Clases que componen el bloque parseador XML.

- **LectorXML:** es la clase principal del bloque. Dado un fichero de entrada XML, se encarga de sacar toda la información contenida en él y clasificarla para posteriormente, poder utilizarla creando las pruebas indicadas en el XML con los datos que se especifiquen en el fichero.
- **Maquina:** es una clase que modela una máquina virtual, la cual puede representar un host o un router.
- **ParesRed:** es una clase que modela un objeto que representa un par ip-interfaz de red. Gracias a esta clase tendremos la relación entre el nombre de un interfaz de red y la ip que tiene dicho interfaz.

6.5.1.2. Bloque pruebas

Una vez extraída la información del fichero de corrección que indica las prueba a realizar y los datos necesarios para realizarlas es necesario construir dichas pruebas. Este bloque es el encargado de realizar esta tarea. En la figura 6.7 se pueden ver los atributos y métodos de las clases que componen este bloque. Para simplificar, hemos decidido no mostrar en el diagrama los métodos get y set.

- **Prueba:** es la clase padre que modela cualquier tipo de prueba que se puedan realizar sobre el escenario de red.
- **Conectividad:** esta clase modela una prueba de corrección de tipo conectividad. Esta clase construye la prueba de conectividad, envía la señal para que se ejecute el script correspondiente a este tipo de prueba de corrección y almacena el resultado obtenido tras la prueba.
- **Trazabilidad:** esta clase modela una prueba de corrección de tipo trazabilidad. Esta clase se encarga de construir la prueba de traceo, enviar la señal para que se ejecute el script correspondiente a este tipo de prueba de corrección y almacenar el resultado obtenido tras la prueba.

Figura 6.7: Clases que componen el bloque pruebas.

- **Bgp:** esta clase modela una prueba de corrección de tipo configuración de rutas para el protocolo BGP. La función de esta clase es construir la prueba de BGP, enviar la señal para que se ejecute el script correspondiente a este tipo de prueba de corrección y almacenar el resultado obtenido tras la prueba.

- **BgpSummary:** esta clase modela una prueba que se encarga de obtener información general de configuración propia del protocolo BGP. No es una prueba de corrección en sí, sino que sirve para obtener un resumen de la configuración realizada para este protocolo. Al igual que en las otras pruebas, esta clase se encarga de construir la prueba, enviar la señal para que se ejecute el script correspondiente y almacenar el resultado obtenido tras la prueba.

- **Informacion:** esta clase modela una prueba que se encarga de obtener información general de configuración dando a conocer las direcciones IPs configuradas en las máquinas y las tablas de rutas. Esta prueba, tampoco es una corrección en sí, sino que sirve para obtener un resumen de la configuración realizada. La función de esta clase es construir la prueba, enviar la señal para que se ejecute el script correspondiente y almacenar el resultado obtenido tras la prueba.

- **Robustez:** esta clase modela una prueba de tipo robustez. Esta prueba no se ejecuta en solitario, necesita apoyarse en alguna de las anteriores y tiene 3 fases: deshabilitar un interfaz, ejecutar una prueba de trazabilidad, conectividad y/o BGP y habilitar el interfaz de nuevo. La idea es comprobar si la configuración de red es tan buena como para ser capaz de recuperarse ante fallos de conexión. Esta clase construye la prueba en la que se especifica si vamos a tirar o levantar un interfaz y enviar la señal para que se ejecute el script encargado de realizar la acción.

6.5.1.3. Bloque generador XML

Este es el último bloque que compone el programa Autocorrecion.jar. En este bloque, recopilamos los datos de los alumnos que estén corrigiendo su ejercicio y toda la información obtenida tras realizar las pruebas para generar un fichero en formato XML que contiene toda la información relevante de la corrección del escenario. Las clases que componen este bloque, al igual que sus atributos y sus métodos se pueden ver en la figura 6.8. Para simplificar, hemos decidido no mostrar en el diagrama los métodos get y set.

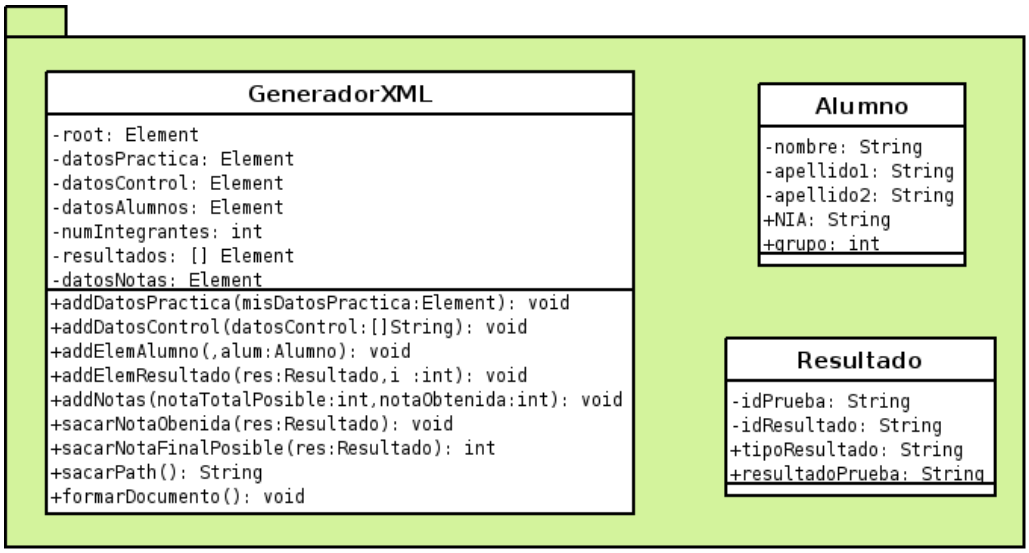


Figura 6.8: Clases que componen el bloque generador XML.

- **Resultado:** en esta clase se representa un resultado a una prueba de corrección.
- **Alumno:** esta clase modela a un alumno que está corrigiendo una práctica. En esta clase se registran datos importantes del alumno que permiten identificarlo inequívocamente para poder calificarlo. Las prácticas se pueden realizar en grupos por lo que una corrección de una simulación puede contener varios objetos de tipo Alumno.
- **GeneradorXML:** es la clase principal del bloque. Utiliza los datos almacenados en las otras dos clases del bloque para generar un fichero en formato XML que resume la corrección realizada, las notas obtenidas en cada prueba individual y la nota final del ejercicio. Este fichero

XML sigue la norma definida en el DTD “autocorrección.dtd” del cual ya hemos hablado y se puede visualizar en el apéndice A.

6.5.2. Scripts de corrección

Los scripts de corrección son un submódulo auxiliar del programa principal, Autocorreccion.jar, que interactúa directamente con él y con el escenario de red configurado. Son los encargados de realizar las correcciones y obtener los resultado directamente de escenario.

Para obtener y realizar las correcciones se decidió utilizar un submódulo externo al programa principal ya que el lenguaje en el que está desarrollado (Java) no tiene ninguna herramienta o librería que nos facilite esta tarea. Las máquinas virtuales que forman parte de la simulación son accesibles a través del equipo anfitrión y tanto las máquinas virtuales como el anfitrión tienen instalado un sistema Linux. Por este motivo se pensó que la mejor opción era tener un conjunto de programas simples, realizados en “Shell-Script”[8] , que se encargaran de realizar las pruebas de corrección y obtener los resultados de éstas. El programa principal Autocorreccion.jar invocará los scripts correspondientes a las pruebas que indique el fichero de corrección XML, los scripts realizarán la corrección y enviarán el resultado al programa principal para que éste los almacene.

En la figura 6.9, vemos cual es la correspondencia entre la prueba de corrección modelada en la clase Java y el script que realiza dicha prueba y, a continuación vamos a explicar en detalle cada uno de los scripts que forman parte de este submódulo, indicando los argumentos de entrada de cada uno, la lógica de funcionamiento y el resultado que devuelve cada uno.

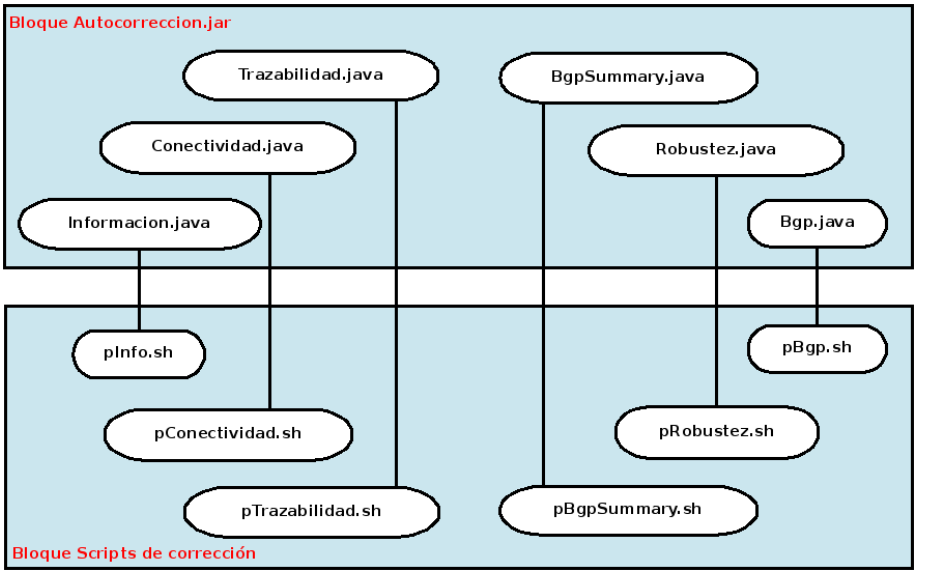


Figura 6.9: Correspondencia entre las clases Java del bloque del programa principal y del bloque de scripts de corrección.

[8] Shell-Script: un script es un archivo que incluye un conjunto de comandos. Un Shell-Script es un fichero donde hemos introducido cualquier comando que se puede ejecutar en la shell.

6.5.2.1. Script pConectividad.sh

En esta prueba se realiza un ping entre las dos máquinas virtuales que se hayan especificado en el XML. Devuelve "1" (si no hay conectividad) y "0" (si hay conectividad). El diagrama de funcionamiento se puede ver en la figura 6.10. Los argumentos de entrada que recibe este script son:

1. La IP de la red de mantenimiento de la máquina que inicia el ping. Este parámetro es necesario porque el anfitrión se conecta a la máquina origen para desde ahí lanzar la prueba, tal y como se haría la corrección en caso de hacerlo de forma manual.
2. El tipo de maquina virtual del nodo que lanza el ping: "H" si es un host y "R" si es router. Esto es porque dependiendo el tipo de nodo trataremos la salida de una forma u otra.
3. La IP destino, que es la máquina contra la que se lanza el ping.
4. El protocolo con el que se realiza la prueba: "4" para indicar que es IPv4 y "6" para indicar que quiere realizar una prueba de IPv6.

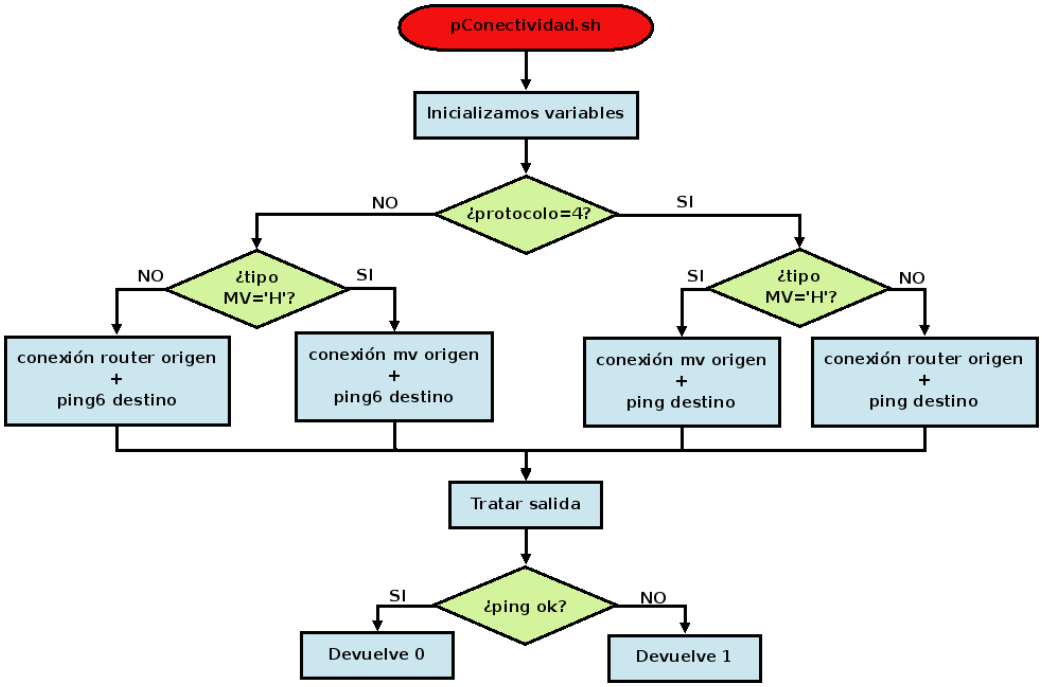


Figura 6.10: Diagrama de flujo del script pConectividad.sh

6.5.2.2. Script pTrazabilidad.sh

En esta prueba se realiza un traceroute entre las dos máquinas virtuales que se hayan especificado en el XML. Esta prueba devuelve la salida completa del traceroute En la imagen 6.10 podemos ver el diagrama de flujo de este script. Los argumentos de entrada que recibe este script son los mismos que para el script anterior:

1. La IP de la red de mantenimiento de la máquina que inicia el traceroute. Sirve para que el anfitrión se conecta a la máquina origen para desde ahí lanzar la prueba.

2. El tipo de maquina virtual que es quien lanza el traceroute: “H” si es un host y “R” si es router.

3. La IP destino, que es la máquina contra la que se lanza el traceroute.

4. El protocolo con el que se realiza la prueba: “4” para indicar que es IPv4 y “6” para indicar que quiere realizar una prueba de IPv6.

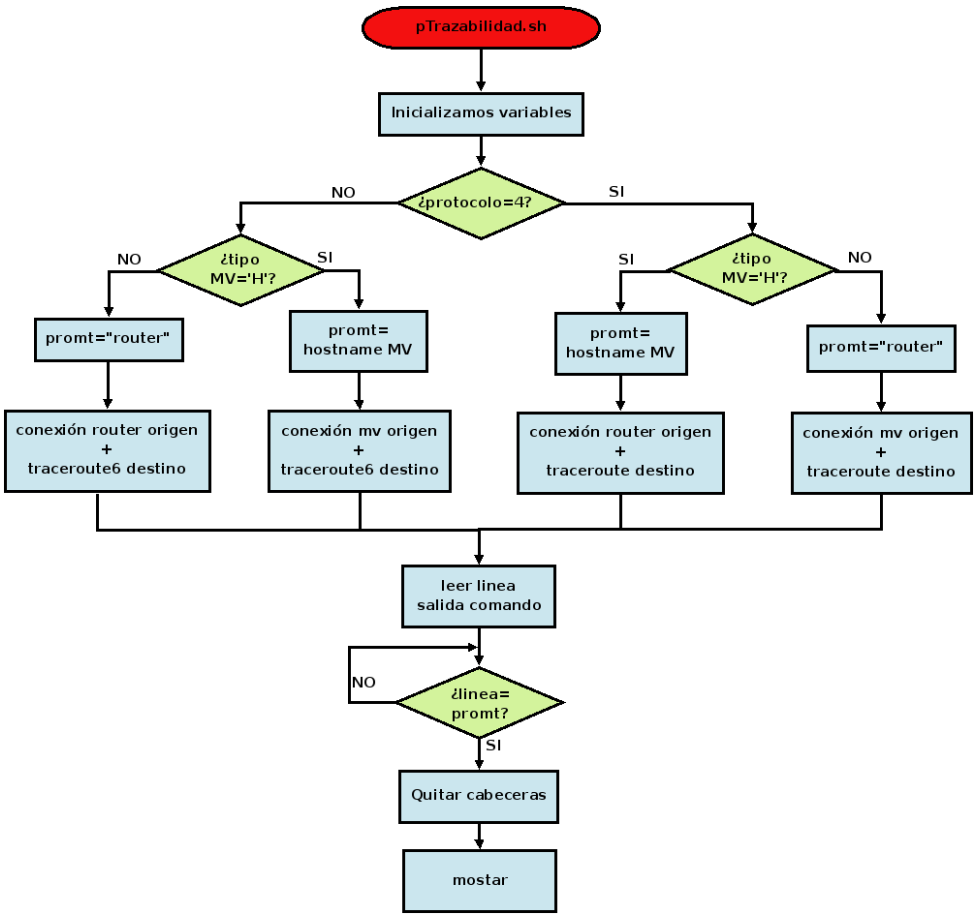


Figura 6.11: Diagrama de flujo del script pTrazabilidad.sh

6.5.2.3. Script pInfo.sh

Esta prueba obtiene las IPs configuradas en la máquina y su tablas de rutas, tanto en máquinas de tipo host como en máquinas de tipo router. El diagrama de flujo del script se puede ver en la figura 6.12. Los argumentos de entrada son los siguientes :

1. La IP de la red de mantenimiento de la máquina que se quiere obtener la información. Sirve para que el anfitrión se conecta a la máquina y ejecute ahí los comandos.

2. El tipo de maquina virtual de la que queremos obtener la información: “H” si es un host y “R” si es router. Distinguimos estos dos casos porque el comando que se lanza en cada tipo de máquina es diferente.

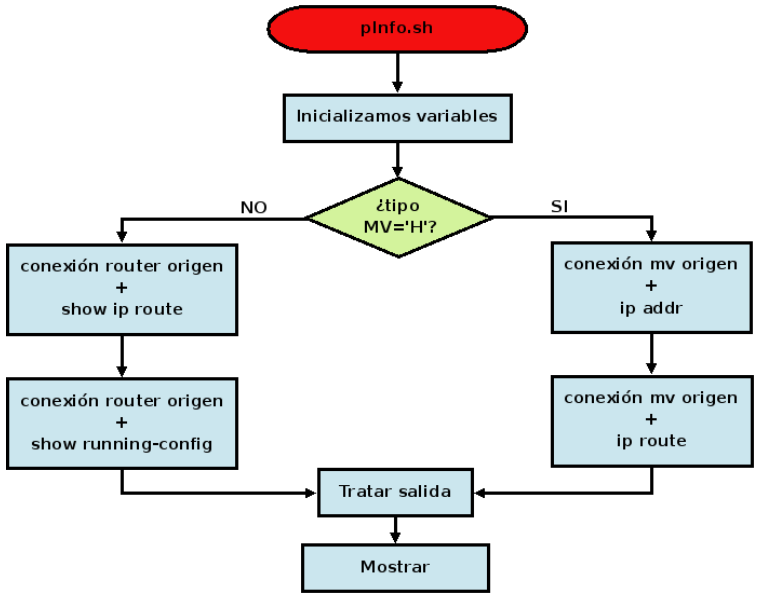


Figura 6.12: Diagrama de flujo del script pInfo.sh

6.5.2.4. Script pBgp.sh

Esta prueba se ejecutará sólo en máquinas virtuales de tipo router. Consiste en ejecutar un comando que devuelve información sobre los caminos BGP posibles para llegar de una máquina origen a una máquina destino. El comando que se ejecuta es “show ip bgp [destino]”. Esta prueba devuelve una linea de texto por cada as_path configurado en el router y marca el camino escogido como mejor con el texto “-best” tras el as_path correspondiente. En la imagen podemos ver un ejemplo de la salida original de este comando (a) y la salida del script PBgp.sh que ofreceremos al programa principal (b). Los argumentos de entrada que recibe este script son:

1. La IP de la red de mantenimiento de la máquina que lanza el comando. Sirve para que el anfitrión se conecta a la máquina origen para desde ahí lanzar la prueba.

2. La IP destino, que es la máquina contra la que se lanza el comando.

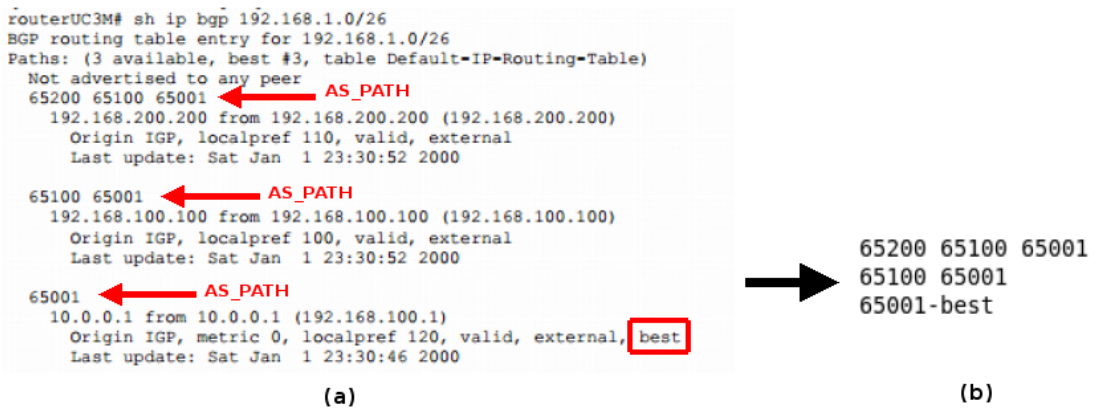


Figura 6.13: Ejemplo de la salida del comando ejecutado en el router y la salida de nuestro script.

El diagrama de flujo que describe el funcionamiento del script es el que se ve en la figura 6.14.

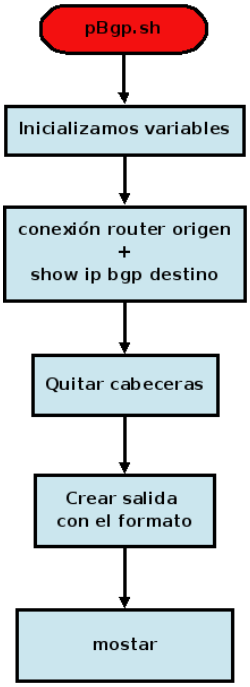


Figura 6.14: Diagrama de flujo del script pBgp.sh

6.5.2.5. Script pBgpSummary.sh

Esta prueba sólo puede ejecutarse en máquinas de tipo router. Obtiene los atributos generales de la tabla de BGP y el estado de los vecinos configurados ejecutando el comando “show ip bgp

summary". El diagrama de flujo del script aparece en la figura 6.15. Solo tiene un argumento de entrada:

1. La IP de la red de mantenimiento de la máquina que lanza el comando. Sirve para que el anfitrión se conecta a la máquina para desde ahí lanzar la prueba.

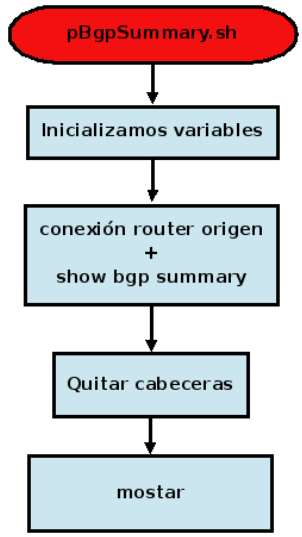


Figura 6.15: Diagrama de flujo del script pBgpSummary.sh

6.5.2.6. Script pRobustez.sh

Este script tiene dos funciones: deshabilitar un interfaz de red y habilitarlo dejando la misma configuración que tenía la máquina antes de deshabilitarlo. No devuelve ningún resultado al programa principal Autocorreccion.jar. Los argumentos de entrada de este script son los siguientes:

1. La IP de la red de mantenimiento de la máquina a la cual vamos a deshabilitar o habilitar un interfaz. Sirve para que el anfitrión se conecta a la máquina y ejecute ahí los comandos.

2. El nombre del interfaz que queremos deshabilitar.

3. El tipo de maquina virtual a la cual le vamos a deshabilitar o habilitar un interfaz: "H" si es un host y "R" si es router. Distinguimos estos dos casos porque el comando que se lanza en cada tipo de máquina es diferente.

4. El protocolo que se está utilizando: "4" para IPv4 y "6" para IPv6.

5. La acción que se desea realizar: "D" para deshabilitar el interfaz (down) y "U" para habilitarlo (up).

El diagrama de flujo de ese script se puede ver en la figura 6.16.

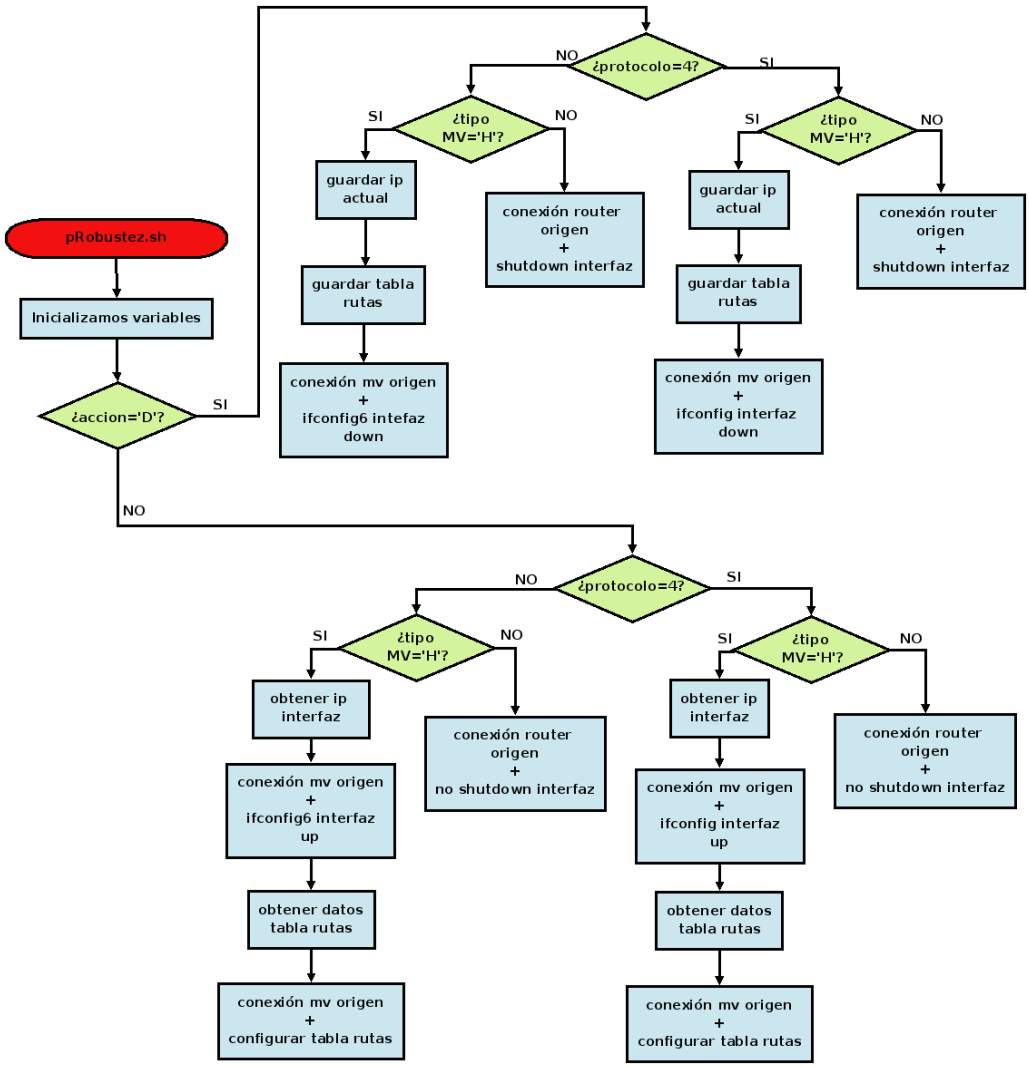


Figura 6.16: Diagrama de flujo del script pRobustez.sh

6.5.2.7. Script numIpbyDev.sh

Este script no tiene una clase que lo modele. Es un script auxiliar que utilizamos al comienzo de la corrección para comprobar que cada interfaz sólo tiene una IP configurada. En caso de tener más de una, la corrección terminaría y mostraría al usuario un mensaje de error indicándole que la configuración no es correcta porque en la simulación hay máquinas virtuales con más de una IP por interfaz. Este script devuelve "1" (si la configuración tiene más de una IP en algún interfaz) y "0" (si sólo hay, como máximo, una IP por interfaz. Los argumentos de entrada que recibe el script son los siguientes:

1. La IP de la red de mantenimiento de la máquina que queremos conocer el número de IPs por

interfaz.

2. El tipo de maquina virtual de la cual queremos obtener información: “H” si es un host y “R” si es router.

A continuación, en la figura 6.17, podemos observar el diagrama de flujo del funcionamiento del script.

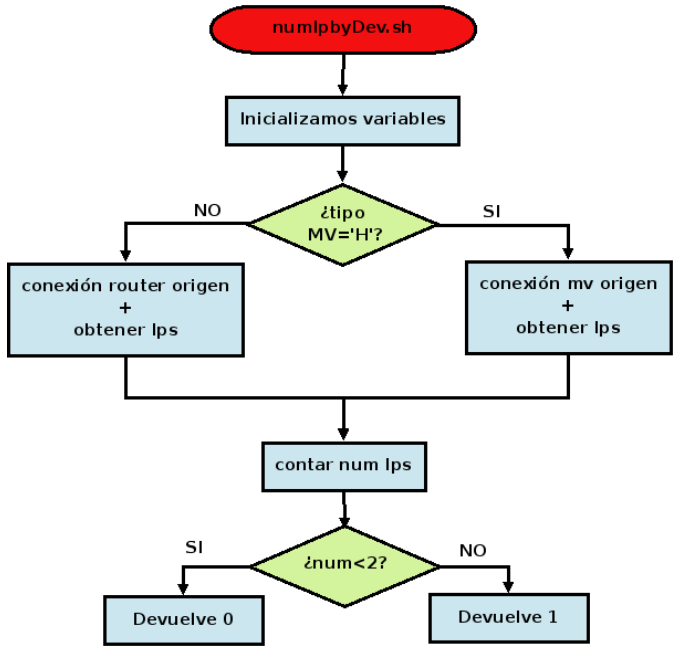


Figura 6.17: Diagrama de flujo del script numIpDev.sh

6.5.3. Seguridad

Se ha diseñado e implementado una herramienta de corrección de practicas, lo que implica que sea necesario tener un software de anticopia para asegurar, tanto al profesor como al alumno, que este método de corrección es fiable, seguro y robusto.

Como hemos explicado en las secciones anteriores, el entorno de corrección genera un fichero en el cual aparece la nota de cada prueba. Este fichero se envía al profesor vía correo electrónico. Es relativamente fácil editar este fichero, cambiar la nota obtenida tras la corrección y enviarlo al profesor, por este motivo, hemos decido aplicar un módulo de seguridad que impida que esto se pueda hacer. Este módulo de seguridad tiene que cumplir los siguientes objetivos:

1. No queremos que el alumno puede editar el fichero, modificarlo a su gusto y enviarlo.
2. Tampoco queremos que sobre un mismo escenario ya configurado se puedan ejecutar n correcciones, haciendo creer al profesor que cada alumno ha configurado su propio escenario y lo ha corregido.
3. Estamos enviando un fichero con datos confidenciales (nombres, NIAs, calificación) a través de la red. Esto puede provocar que los datos sean leídos, violando la intimidad de los alumnos, y/o los datos sean modificados mal intencionadamente. Por estos motivos es necesario que el envío de estos datos esté cifrado de alguna forma.

Para cumplir con cada uno de nuestros objetivos hemos aplicado distintos métodos de seguridad nos proporcionan la tranquilidad necesaria para utilizar este entorno de forma fiable y segura.

El primero de los objetivos fijados lo hemos conseguido aplicando una firma digital al documento que se envía. Las firmas digitales son un método de verificación de la autenticidad del mensaje enviado que dependen tanto del contenido del mensaje como del remitente del mismo. El proceso de firma digital de documentos tiene dos partes: la de creación de firma y la de verificación de la misma en el destino.

Para crear la firma digital de un documento se aplica una función hash sobre este, que transforma el mensaje en un resumen de tamaño fijo independientemente del tamaño del fichero. Una vez obtenido el resumen, se aplica el algoritmo de firma digital utilizando la clave privada del remitente. Lo que se envía es el mensaje original y el resumen firmado.

Para verificar la firma, se aplica al resumen la clave pública correspondiente, posteriormente se realiza un resumen hash del mensaje original y por último se comparan los dos resúmenes. Si el texto ha sido modificado durante el envío, los resúmenes no coincidirán, por lo que podremos asegurar que el texto recibido no es igual que el original.

Es importante destacar que una posible manipulación, posterior a la firma del mensaje, es detectada en el destino al aplicar la verificación.

Conociendo en que consiste la firma digital, vamos a explicar como hemos decido firmar nuestros documentos para conseguir el primer objetivo. Hemos decidido que el firmante de las correcciones que se envían va a ser un mismo usuario común para todos, que vamos a identificar como el usuario de la aplicación: uvil. De esta forma, con un solo par de claves se pueden comprobar las correcciones, ya que si cada alumno tuviera que tener su propio par de claves sería mucho más difícil de gestionar por el gran número de alumnos que realizan estas prácticas. Obsérvese que el usuario uvil solo va a utilizarse para firmar las correcciones. En el proceso de entrega de la práctica, la aplicación se encargará de firmar el documento de forma completamente transparente para el alumno, impidiendo que

este pueda manipularlo. Esto nos permite detectar que quien ha corregido la práctica y ha enviado el correo electrónico al profesor ha sido la aplicación y no el alumno. Como se ha explicado más arriba, al enviar un documento firmado tienen que enviarse la firma (resumen al cual se le ha aplicado la clave privada del remitente) y el documento. Para evitar que los datos confidenciales de los alumnos viajen en claro por la red, vamos a cifrar el fichero XML con la corrección mediante cifrado asimétrico[9] , utilizando el par de claves del profesor destino. En la imagen 6.18 se muestra el proceso de firmado y verificación que hemos explicado.

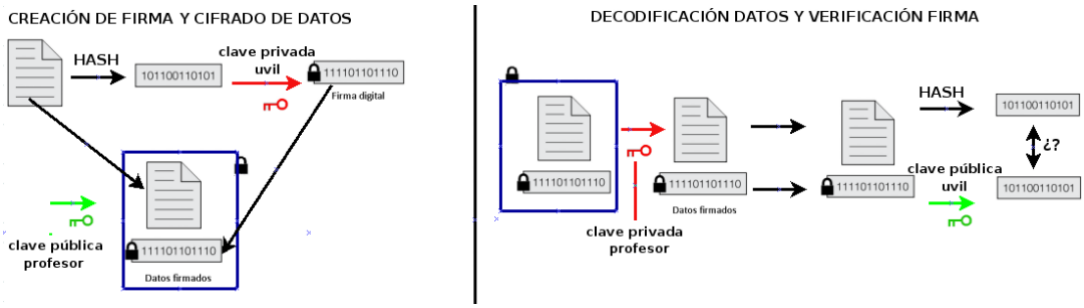


Figura 6.18: Esquema de cómo se genera la firma y el cifrado de los datos y cómo en el destino se descifran los datos y se verifica la firma.

Para conseguir el segundo de nuestros objetivos, hemos introducido unos datos de control que le permiten al profesor detectar que cada corrección se ha aplicado sobre un escenario diferente. Estos datos de control son:

- **Fecha:** se refiere a la fecha en la que se lanzó el escenario de red que se tenía que configurar. El formato de la fecha es ”DiaSemana_Mes_DiaMes_Año__Hora:Minuto:Segundo”
- **Dirección IP:** es la Ip de la máquina en la cual se ha lanzado el escenario para proceder a su configuración.
- **PID:** es el identificador de proceso del escerario de red. En Linux/Unix, cuando se lanza un proceso éste queda registrado en el sistema con un identificador único para esa máquina. Cuando lanzamos nuestro escenario de red para ser configurado, se registra cual es el PID correspondiente a ese escenario.
- **Hostname:** es el nombre del equipo en el cual se lanzó el escenario de red.

Es imposible que se entreguen dos prácticas en las que coincidan estos datos de control si éstas se han realizado de forma “legal”.

[9]Cifrado asimétrico: método criptográfico que usa una clave pública y una privada para el envío y recepción de documentos. Este par de claves son complementarias entre sí, además cada clave pública se corresponde sólo con una clave privada. La clave pública es conocida por todos y la clave privada es sólo conocida por el remitente. Cuando se envía un documento con este método criptográfico, se cifra el mensaje con la clave privada y se envía por la red. Cuando llega a su destinatario, si éste tiene la clave pública correspondiente podrá desencriptar el mensaje y leerlo. De esta manera se evita que los datos viajen en claro por la red y se limita la posibilidad de desencriptado del fichero sólo a aquellas personas en quien confía el remitente y a las cuales se les ha facilitado previamente la clave pública.

El último de nuestros objetivos era conseguir que el alumno no pudiese modificar el fichero XML de la calificación manipulando los resultados de las pruebas de corrección y/o los datos de control. Para conseguirlo, simplemente hemos forzado a que el propietario del fichero sea el usuario uvil, y los permisos sobre este fichero son sólo de lectura. De esta forma el fichero se podrá leer o consultar pero nunca se podrá escribir.

6.5.4. Script corregirLabUC3M.sh

Hasta ahora hemos explicado cómo funciona todo el proceso de corrección, pero no hemos explicado cómo lanza el alumno este proceso y que tiene que tener en cuenta a la hora de autocorregir un escenario.

Uno de nuestros objetivos era conseguir un entorno que fuese sencillo de utilizar por los alumnos. Para conseguir esto, hemos creado un script llamado "corregirLabUC3M.sh" que interactúa con el alumno solicitándole cierta información necesaria para realizar la corrección.

Este script permite ejecutar dos modos de autocorrección:

- **Modo entrega de práctica.** Si se elige este modo de ejecución, se realizará la corrección del escenario, se generará la salida con la nota y se enviará este fichero al profesor de prácticas. Todo se realiza en un mismo paso, el alumno sólo tiene que ejecutar el script rellenando los campos solicitados y este se encargará de toda la ejecución de la corrección.
- **Modo consulta estado práctica.** Si se ejecuta el script en este modo, se realizará una corrección del escenario, pero no se enviará el fichero con la calificación al profesor. Este modo de ejecución es útil para el alumno ya que permite conocer si la configuración realizada es correcta o no, pudiendo perfeccionarla antes de entregarla. De este modo se fomenta el auto-aprendizaje del alumno.

Los datos que se le piden introducir al alumno son los siguientes:

- El modo de ejecución: Permite dos posibles valores numéricos. El alumno introducirá "0" si se desea sólo consultar la corrección de la práctica y "1" si se desea corregir y entregar la práctica.
- Número de integrantes del grupo de prácticas.
- Nombres de cada integrante del grupo de prácticas.
- Primer apellido de cada integrante del grupo de prácticas.
- Segundo apellido de cada integrante del grupo de prácticas.
- NIAs de cada uno de los integrantes del grupo de prácticas.
- Nombre del fichero de corrección. Este fichero es el que incluye la configuración de la corrección con las pruebas que se van a realizar sobre el escenario de red y el valor que se le asigna a cada una de las pruebas.
- Login del profesor. Este datos sólo se pedirá en caso de que se ejecute el script en modo entrega de práctica. El login del profesor se utiliza para el envío del mail.

En la figura 6.19 podemos observar el diagrama de flujo de este script. Como se puede ver, es él quien se encarga de realizar toda la corrección de escenario ocultándole a alumno la complejidad del entorno.

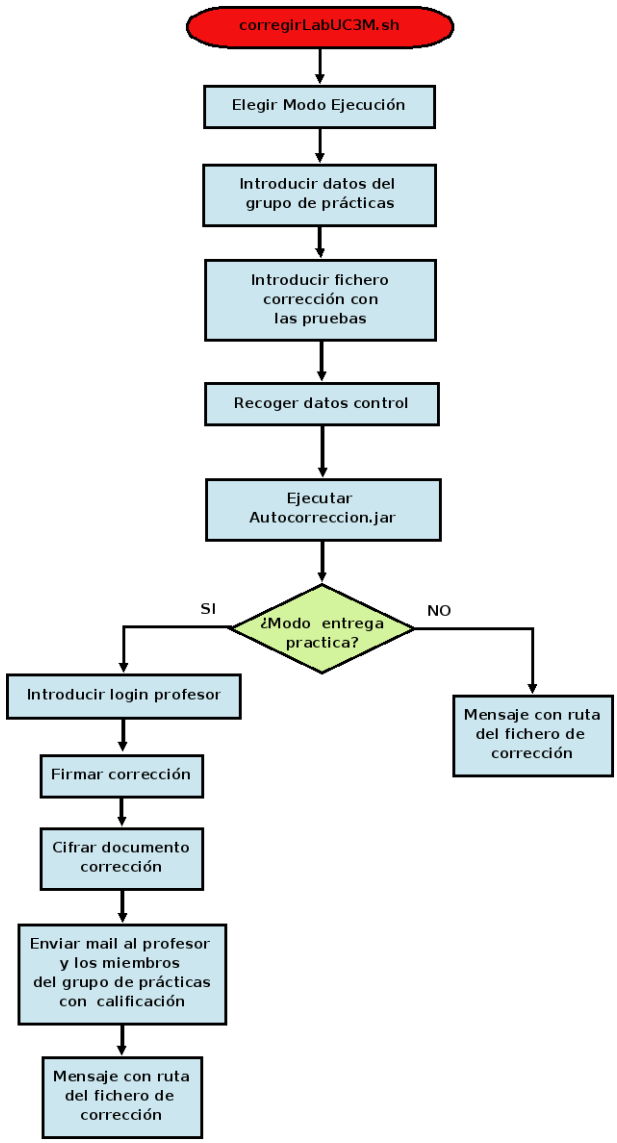


Figura 6.19: Diagrama de flujo del script corregirLabUC3M.sh.

6.6. GENERACIÓN Y PRESENTACIÓN DE LA CALIFICACIÓN

A lo largo de todo el capítulo hemos hablado del fichero XML que se genera tras la corrección de un escenario de red y que contiene el resultado de las pruebas realizadas y la calificación de la práctica. En este capítulo vamos a explicar en detalle cómo es este fichero y cuáles son los tags que lo componen. Además explicaremos como se presentan los resultados tanto al profesor como al alumno.

Este XML también sigue las reglas definidas en el DTD autocorrección.dtd que hemos nombrado varias veces. Las restricciónes definidas sobre el XML de resultados y las relaciones entre los nodos que lo componen se pueden ver en las figuras 6.20 y 6.21.

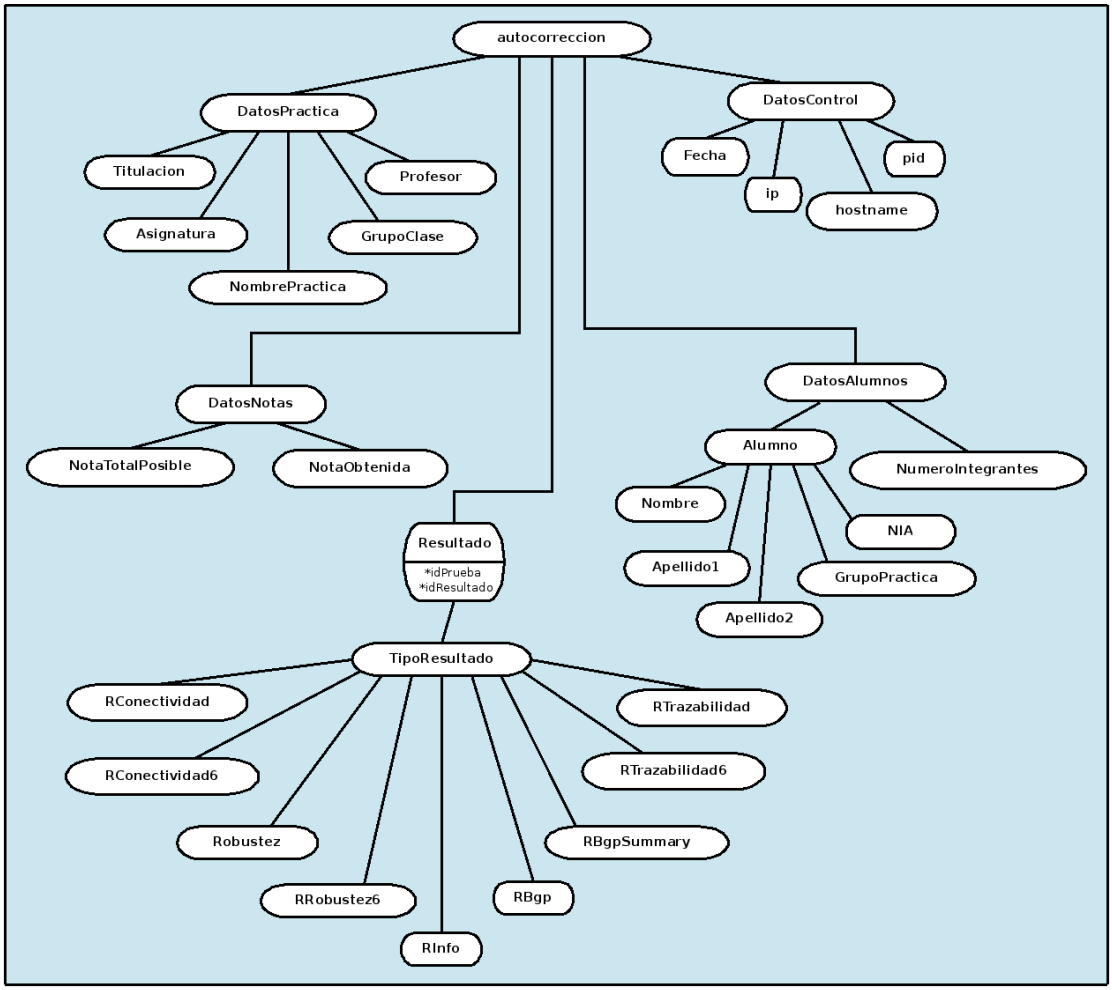


Figura 6.20: Elementos que componen el XML que describe los resultados de la corrección.

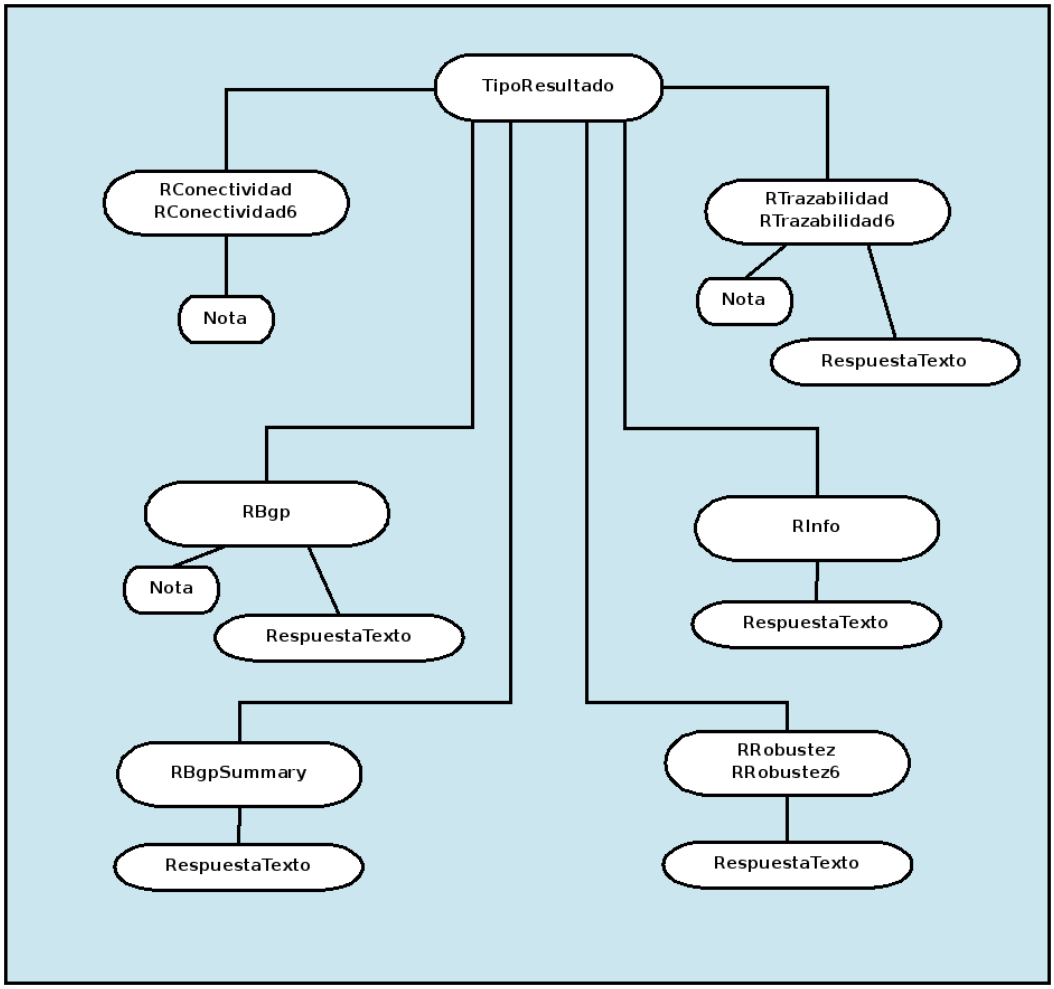


Figura 6.21: Elementos hijos del nodo TipoResultado que definen la corrección de cada prueba.

A continuación vamos a explicar con más detalle los nodos del XML que no han sido definidos hasta ahora.

- **autocorreccion:** es el nodo padre de todos los demás.
- **DatosPractica:** este nodo recoge los datos generales de la práctica y permite que se pueda identificar fácilmente a qué practica corresponde la corrección.
- **DatosControl:** este nodo representa los datos de control que se han explicado en la sección de Seguridad. Contiene la información recogida durante la corrección del ejercicio para evitar la copia de prácticas.
- **DatosAlumnos:** este nodo contiene la información de los alumnos que forman un grupo de prácticas. Los hijos de este nodo se describen en la tabla 6.6.

Elementos hijos del nodo DatosAlumnos	
Elemento	**Descripción**
NumeroIntegrantes	Indica el número de alumnos que forman el grupo de prácticas.
Alumno	Define un integrante del grupo de prácticas. Debe haber tantos nodos de este tipo como indique el nodo "NumeroIntegrantes". Los hijos de este nodo son: Nombre, Apellido1, Apellido2, NIA, GrupoPractica.
Nombre	Indica el nombre de un alumno.
Apellido1	Indica el primer apellido de un alumno.
Apellido2	Indica el segundo apellido de un alumno.
NIA	Es el número de identificación de alumnos. Con este número se identifica inequívocamente a un alumno.
GrupoPractica	Indica el grupo de prácticas al que pertenece el alumno.

Tabla 6.6: Descripción del hijo del nodo DatosAlumnos

- **DatosNotas:** en este nodo alberga la información relativa a la calificación total del ejercicio. Los hijos de este nodo se describen en la tabla 6.7..

Elementos hijos del nodo DatosNotas	
Elemento	**Descripción**
NotaTotalPosible	Es la nota máxima que se puede alcanzar al realizar la práctica si todas las pruebas devuelven un resultado correcto.
NotaObtenida	Es la nota que el alumno ha conseguido realizando la práctica. Se obtiene de sumar la nota obtenida en cada prueba de corrección.

Tabla 6.7: Descripción del hijo del nodo DatosNotas

- **Resultado:** este nodo representa el resultado de una de las prueba aplicadas en el escenario de red configurado con el entorno virtual. Debemos crear un elemento de este tipo por cada prueba que se haya ejecutado sobre el entorno. Los hijos de este nodo, cuya estructura se pueden visualizar de forma gráfica en la imagen 6.21, definen las características propias de cada resultado y están descritos por los elementos que aparecen en la tabla 6.8.

Elementos hijos del nodo Resultado	
Elemento	**Descripción**
TipoResultado	Indica el tipo de resultado que está asociado a un tipo de prueba concreto. Debe haber tantos nodos de este tipo como pruebas se hayan ejecutado. Los hijos de este nodo son: RConectividad, RConectividad6, RTrazabilidad, Rrazabilidad6, RBgp, RBgpSummary, RInfo, RRobustez y RRobustez6. Estos nodos se definen en la tabla 6.9.

Tabla 6.8: Descripción del hijo del nodo Resultado

Elementos hijos del nodo TipoResultado	
Elemento	**Descripción**
RConectividad / Rconectividad6	Define un resultado a una prueba de conectividad, tanto para el protocolo IPv4 como para IPv6.
RTrazabilidad / RTrazabilidad6	Define un resultado a una prueba de trazabilidad, tanto para el protocolo IPv4 como para IPv6.
RBgp	Define un resultado a una prueba de configuración de rutas realizadas con el protocolo BGP.
RBgpSummary	Define un resultado a una prueba que muestra las características generales de una tabla BGP configurada.
RInfo	Define un resultado a una prueba que devuelve las caracteriscas generales de configuración, tanto en nodos de tipo "host" como en nodos de tipo "router", mostrando las IPs de cada interfaz y las tablas de rutas configuradas.
RRobustez / RRobustez6	Define un resultado a una prueba en la que se deshabilitan y habilitan interfaces de red.

Tabla 6.9: Descripción de los hijos del nodo TipoResultado

Leyendo este documento XML se puede obtener toda la información relacionada con una corrección determinada. Los XML permiten estructurar la información de forma ordenada y tener cierta semántica en los datos que almacenan, sin embargo, leer estos documentos se puede convertir en una tarea tediosa. Nosotros hemos querido que la visualización del resultado de la nota obtenida tenga un formato más atractivo y fácil de leer, para ello, hemos implementado una plantilla XSLT que transforma el XML en una pagina HTML, lo que permite consultar los datos a través de cualquier navegador.

XSLT es un lenguaje de programación que se utiliza para convertir documentos XML en otros documentos XML (que pueden incluso obedecer a DTD diferentes) o para convertir XML en otros documentos HTML (los cuales, por no ser HTMLs puros se denominan XHTML).

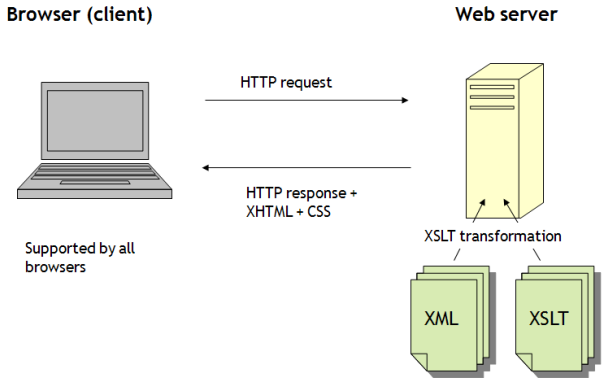


Figura 6.22: Esquema de funcionamiento de XSLT

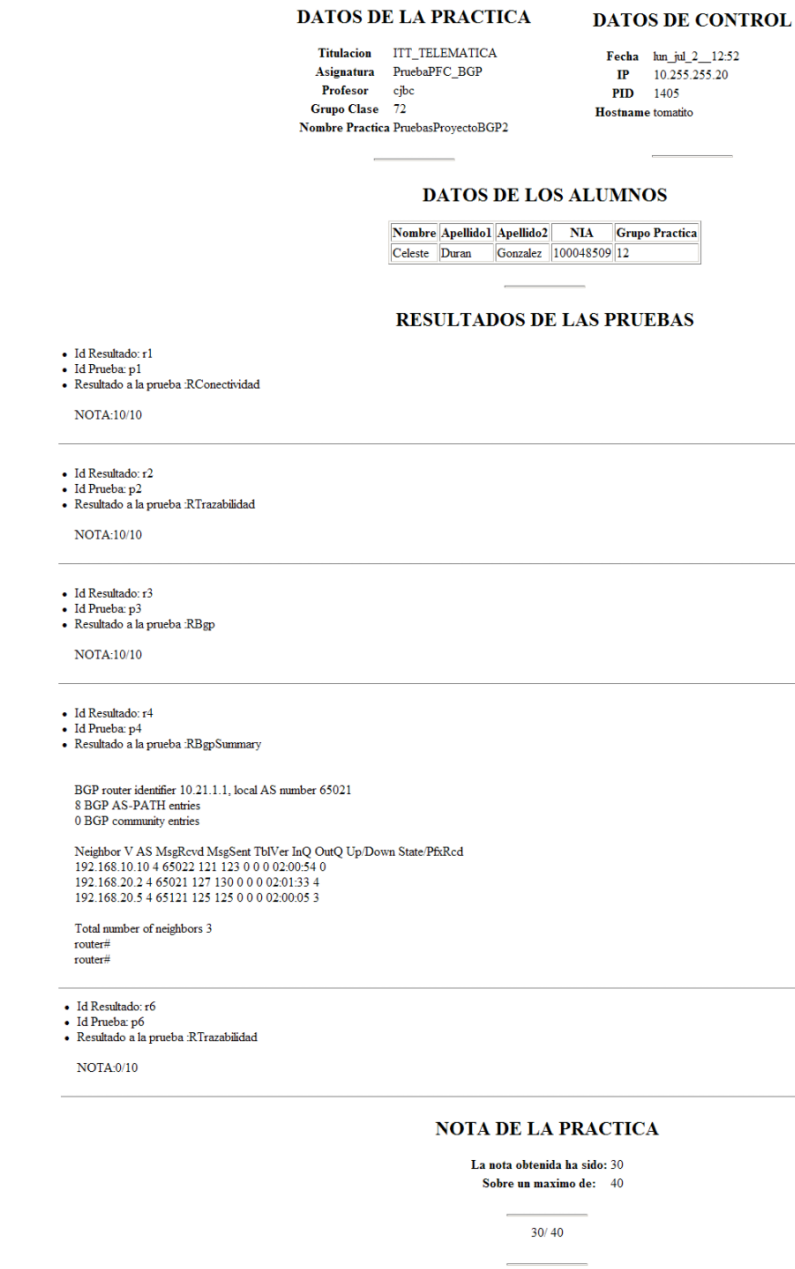
En la figura 6.22 se puede ver cómo funciona XSLT y se visualiza el resultado final. Nótese que en nuestro caso, el servidor web es el propio PC del alumno donde se ejecuta la corrección.

En las figura 6.23 podemos ver un ejemplo de fichero de resultados de prácticas en XML. Este tipo de ficheros se puede ver en texto plano o en un navegador. La imagen que estamos mostrando es del fichero visualizado a través de un navegador, ya que en texto plano es mucho más complejo de ver y entender la estructura que tiene el XML.

Figura 6.23: Ejemplo de XML de corrección sin formato.

En la figura 6.24 vemos como queda el texto tras haberle aplicado la plantilla XSLT. El formato en XHTML es más cómodo de leer, de entender y más atractivo visualmente que el código en XML puro.

La plantilla XSLT se puede consultar íntegramente en el Apéndice A.



DATOS DE LA PRACTICA

Titulacion ITT_TELEMATICA
Asignatura PruebaPFC_BGP
Profesor cjbc
Grupo Clase 72
Nombre Practica PruebasProyectoBGP2

DATOS DE CONTROL

Fecha lun_jul_2__12:52
IP 10.255.255.20
PID 1405
Hostname tomatito

DATOS DE LOS ALUMNOS

Nombre	Apellido1	Apellido2	NIA	Grupo Practica
Celeste	Duran	Gonzalez	100048509	12

RESULTADOS DE LAS PRUEBAS

- Id Resultado: r1
- Id Prueba: p1
- Resultado a la prueba :RConectividad

NOTA:10/10

- Id Resultado: r2
- Id Prueba: p2
- Resultado a la prueba :RTrazabilidad

NOTA:10/10

- Id Resultado: r3
- Id Prueba: p3
- Resultado a la prueba :RBgp

NOTA:10/10

- Id Resultado: r4
- Id Prueba: p4
- Resultado a la prueba :RBgpSummary

BGP router identifier 10.21.1.1, local AS number 65021
8 BGP AS-PATH entries
0 BGP community entries

Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/PfxRcd
192.168.10.10 4 65022 121 123 0 0 0 02:00:54 0
192.168.20.2 4 65021 127 130 0 0 0 02:01:33 4
192.168.20.5 4 65121 125 125 0 0 0 02:00:05 3

Total number of neighbors 3
router#
router#

- Id Resultado: r6
- Id Prueba: p6
- Resultado a la prueba :RTrazabilidad

NOTA:0/10

NOTA DE LA PRACTICA

La nota obtenida ha sido: 30
Sobre un maximo de: 40

30/ 40

Figura 6.24: Ejemplo de XML de corrección con formato una vez se ha aplicado la plantilla XSLT.

6.7. CONCLUSIONES

En este capítulo hemos explicado como se ha diseñado e implementado el módulo de autocorrección de las prácticas de configuración de escenarios de red. No es un entorno sencillo, ya que debido a la funcionalidad y versatilidad que hemos querido ofrecer, es un entorno compuesto por varios módulos que se relacionan entre sí para conseguir el objetivo principal: que las prácticas realizadas en el entorno virtual pueden ser corregidas de forma automática.

Hemos definido un conjunto de pruebas para realizar las autocorrecciones de los escenarios que son prácticamente las mismas pruebas que se realizan en el entorno real, teniendo en cuenta todos los protocolos con los que se realizan prácticas en el Laboratorio.

Gracias al módulo de configuración que hemos diseñado hemos dotado de gran flexibilidad al entorno ya que podemos programar correcciones para multitud de escenarios, dando plena libertad al profesor sobre la topología, el número de máquinas virtuales implicadas en una configuración, los protocolos que se desean pedir que se configuren, etc. Además la forma de configurar estas pruebas es relativamente sencilla, ya que simplemente hay que conocer la estructura del XML que define la corrección y definir en él las pruebas. No requiere un método de programación a bajo nivel.

Aunque es un entorno complejo, el alumno no lo percibirá ya que él simplemente ejecutará el script de autocorrección "corregirLabUC3M.sh" que lanzará todos los procesos que se encargan de realizar las pruebas y obtener una nota. En los Laboratorios del Departamento de Ingeniería Telemática estarán instalados todos los ficheros necesarios para el proceso de autocorrección. Además este script tiene dos modos de ejecución: modo entrega de práctica, en el cual se le enviará la nota al profesor y, modo consulta del estado de la configuración, en el cuál el alumno puede corregir el escenario y saber cual sería su nota, sin mandársela al profesor, con el fin de poder modificar la configuración en caso de que ésta no estuviese correcta.

Se ha diseñado un módulo de seguridad con el cual se pretende evitar, en la medida de lo posible, la copia de prácticas.

Por último, hemos implementado un método gracias al cual es cómoda la visualización de la corrección del alumno.

Capítulo 7

DISEÑO Y DESARROLLO DE LA HERRAMIENTA DE GENERACIÓN DE PLANTILLAS

7.1. INTRODUCCIÓN

En los capítulos anteriores hemos hablado de los ficheros XML que utilizamos para crear los escenarios y las correcciones. Son la base de los dos entornos, y por eso es muy importante que estos ficheros estén bien formados, sean correctos y coherentes.

El formato XML proporciona cierta semántica a los datos que almacena y permite que estos estén estructurados haciendo sencilla su manipulación a través de un procesado jerárquico. Sin embargo, la creación de este tipo de ficheros puede ser tediosa, y dependiendo del tipo de XML que se quiera escribir (según la cantidad de atributos y opciones de los nodos) puede ser una tarea más o menos complicada.

En nuestro caso, queremos facilitar la tarea de generación de XML, ya que el objetivo es que el profesor se pueda centrar en la creación de prácticas y en el tipo de pruebas que se van a aplicar para corregirlas. No queremos que la tarea de crear los ficheros base para nuestro entorno suponga un esfuerzo extra para el profesor, por este motivo, hemos desarrollado una herramienta que permite crear los XML (tanto de los escenarios como de las correcciones) de forma sencilla, sin necesidad de tener que tener conocimientos sobre XML o sobre los DTDs concretos que definen nuestros ficheros.

En este capítulo vamos a explicar cómo se ha diseñado esta herramienta, qué tipo de tecnología se ha empleado para implementarla y qué decisiones hemos tenido que tomar para cumplir nuestros objetivos. Explicaremos en detalle cuales son los pasos que hay que seguir para crear cada tipo de plantilla necesaria para nuestros entornos: XMLs para la generación de escenarios de red y XMLs para generación las pruebas de corrección.

7.2. OBJETIVOS

Los objetivos o requisitos que nos hemos fijado a la hora de diseñar esta herramienta que permite generar los documentos XML para los dos entornos (virtual y de autocorrección) son los siguientes:

- Se ha buscado desarrollar una herramienta que facilitase la tarea de creación de ficheros XML para nuestro entorno virtual y nuestro entorno de autocorrección.
- De fácil acceso. Dado que este entorno está pensado para ser utilizado en diferentes prácticas y para diferentes profesores, sería deseable que la herramienta de generación de XMLs fuese accesible por cualquier profesor, sin necesidad de tener que instalar nada en los PCs de estos.
- Debe ser intuitiva y fácil de utilizar, indicando que dato introducir en cada momento.

7.3. HERRAMIENTA DE GENERACIÓN DE PLANTILLAS

Con el fin de conseguir estos requisitos se ha decido que esta herramienta sea un portal web de creación de ficheros de configuración. Esta herramienta ha sido desarrollada en HTML (para crear la interfaz) y PHP (para procesar los datos). Además, el diseño del portal web incluye el uso de una hoja de estilos CSS[1], dotándolo de una imagen homogénea.

Nuestra herramienta web está compuesta, como se ve en la figura 7.1, por cuatro opciones, cada una de ellas con una función determinada:

Figura 7.1: Pestañas del portal web diseñado para la generación de plantillas XML

[1]CSS: Cascading Style Sheets. Lenguaje que describe la presentación de los documentos estructurados, es decir, describe como se va a mostrar un documento en pantalla.

1. Inicio: esta parte de la web está destinada a dar la bienvenida al sistema, además explica en que consiste este proyecto y que motivaciones han llevado a su desarrollo.
2. Generador Escenarios: en esta pestaña de la Web, es donde se puede crear el fichero XML de configuración para un escenario de red determinado.
3. Plantillas Corrección: en esta zona de la web, es donde se puede crear el fichero XML donde se definen las pruebas de corrección.
4. Contacto: en esta zona, aparecen los datos de contacto del webmaster[2].

En las siguientes secciones vamos a explicar con más detalle en que consisten y como se utilizan las pestañas de "Generación Escenarios" (2) y "Plantillas Corrección" (3) de la web.

7.3.1. GENERACIÓN DE ESCENARIOS DE RED

En esta parte del portal web se procederá a rellenar un formulario para posteriormente generar el XML de configuración que describe un escenario de red, necesario para utilizar el entorno virtual. El funcionamiento de esta parte del portal se puede ver en el diagrama7.2.

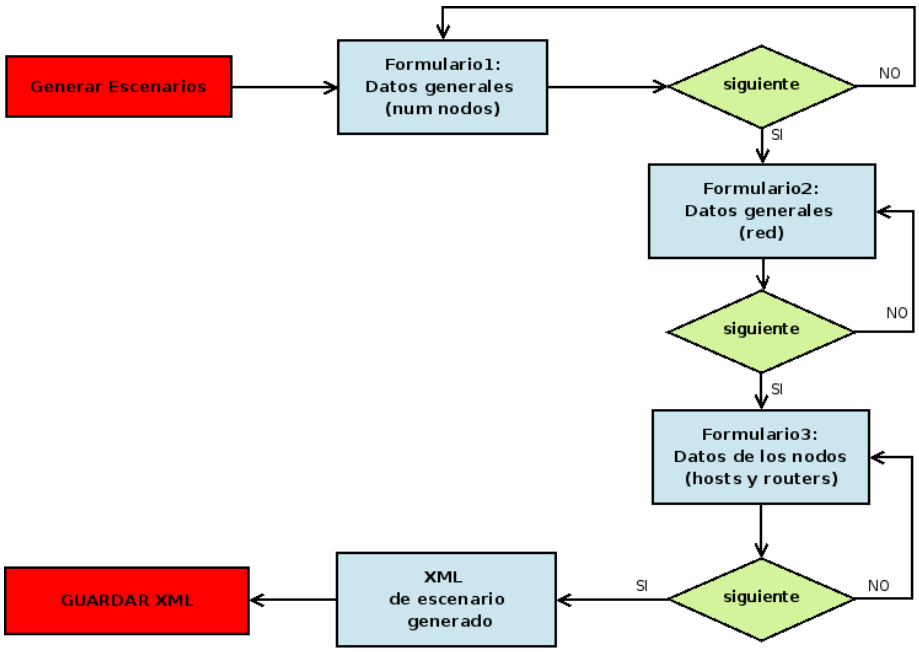


Figura 7.2: Diagrama del funcionamiento de la parte de generación de escenarios de red en el portal

Este formulario de creación de XML consta de 3 partes. Para entender mejor como se rellena, vamos a mostrar cada parte del formulario siguiendo un ejemplo concreto. Imaginemos que queremos crear un escenario como el que se ve en la figura 7.3. Se trata de un escenario muy simple en el que tenemos dos PCs conectados a través de un router. La red de color rojo representa la red de mantenimiento, mientras la red de color azul representa la red propia del escenario.

[2]Webmaster: creador y/o administrador de la página web.

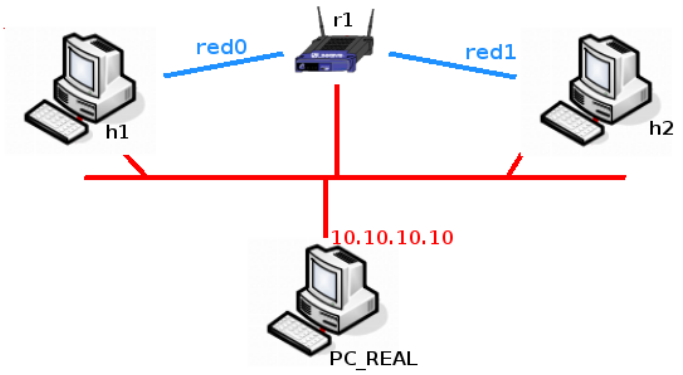


Figura 7.3: Ejemplo de un escenario de red simple.

A continuación vemos que rellenaríamos en cada parte del formulario para generar este escenario:

1. **Parte 1**. Número de nodos que intervienen en la simulación y número de redes que hay que definir. La pantalla que el usuario ve es la que aparece en la figura 7.4.

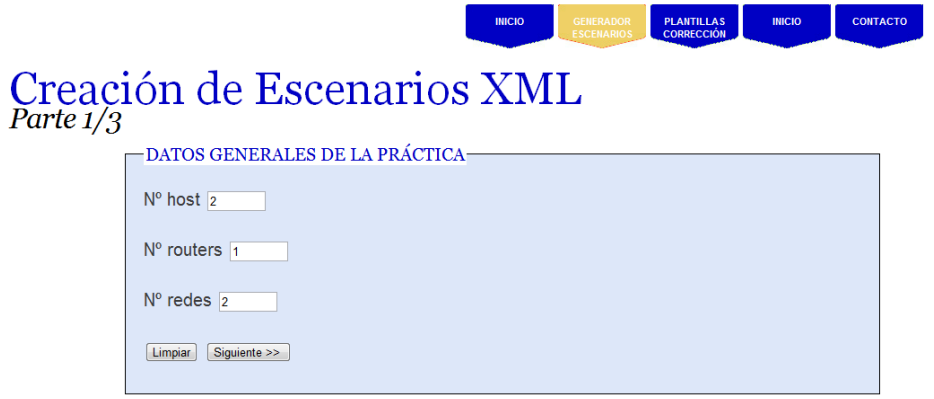


Figura 7.4: Parte 1 del formulario. Rellenamos datos referentes al número de nodos que intervienen en la simulación.

2. **Parte 2**. Definición de datos generales de la simulación y datos comunes para todos los nodos. En esta pantalla, la mayor parte de los datos aparecerán con valores por defecto, que pueden ser modificados si es necesario. Los únicos datos que hay que rellenar porque no tienen valor por defecto son: el nombre de la simulación y los nombres que queremos dar a las redes que servirán para hacer las conexiones entre las máquinas virtuales (en el dibujo 7.3 son las líneas de color azul). Para la red de mantenimiento (líneas de color rojo de la imagen 7.3) se propone un valor por defecto, que puede modificarse en caso de que así se desee. En la figura 7.5 vemos como se rellenaría esta parte del formulario para nuestro ejemplo.

Figura 7.5: Parte 2 del formulario. Rellenamos datos comunes para la simulación.

3. **Parte 3**. Características propias de cada nodo. Es la parte más importante del formulario, ya que aquí se configuran los nombres de los nodos, la red a la que se conecta cada interfaz, el protocolo a utilizar (IPv4 o IPv6) y el gateway (estas dos últimas sólo para el caso de máquinas tipo host). Una vez hemos terminado de rellenar esta parte del formulario, podemos pinchar en el botón "Finalizar" para ver nuestro XML. En la imagen 7.6 vemos cómo habría que rellenar el formulario para nuestro ejemplo. Nótese que el router dispone de 5 interfaces ethernet y en este caso, sólo vamos a utilizar 2. Los otros interfaces pueden conectarse a cualquier red. Una vez generado el XML podemos guardarlo pinchando en la opción guardar del navegador que estemos utilizando.

Figura 7.6: Parte 3 del formulario. Rellenamos los datos específicos de cada máquina virtual.

7.3.2. GENERACIÓN DE PLANTILLAS DE CORRECCIÓN

En esta otra parte del portal web se procederá a rellenar un formulario para posteriormente generar el XML necesario para el entorno de autocorrección. En este fichero de configuración se describen las pruebas de corrección que se desean realizar sobre un escenario de red configurado utilizando el entorno virtual. El funcionamiento de esta parte del portal se puede ver en el diagrama7.7.

Este formulario de creación de XML también consta de 3 partes. Siguiendo la explicación del caso de creación de escenarios, vamos a explicar como se rellenaría siguiendo un ejemplo concreto. En este caso vamos a suponer que queremos hacer una corrección para el escenario anterior 7.3, y sobre la cual vamos a aplicar dos pruebas: una de conectividad entre los host h1 y h2 y otra de trazabilidad entre los mismos host.

Vamos explicar que rellenaríamos en cada parte del formulario para generar esta corrección:

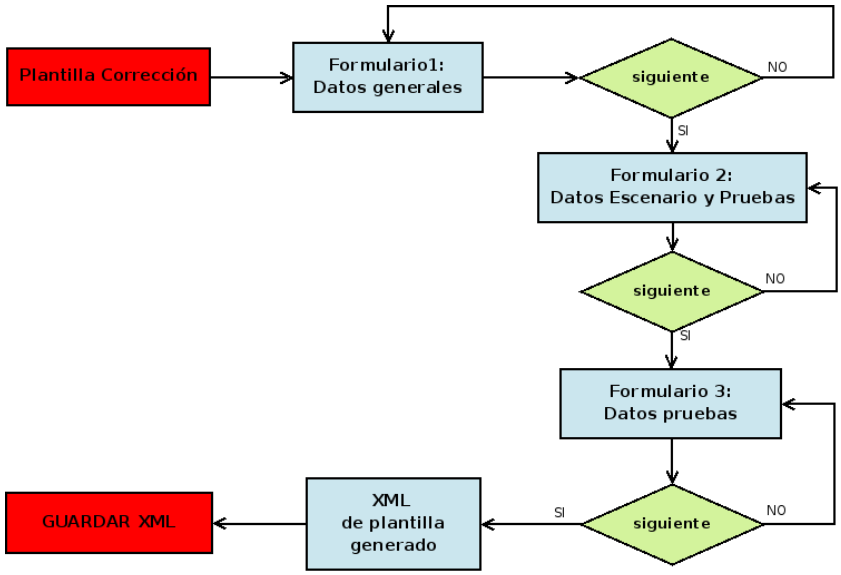


Figura 7.7: Diagrama del funcionamiento de la parte de generación de correcciones a través del portal.

1. **Parte 1**. Datos generales de la práctica y del escenario a corregir 7.8.

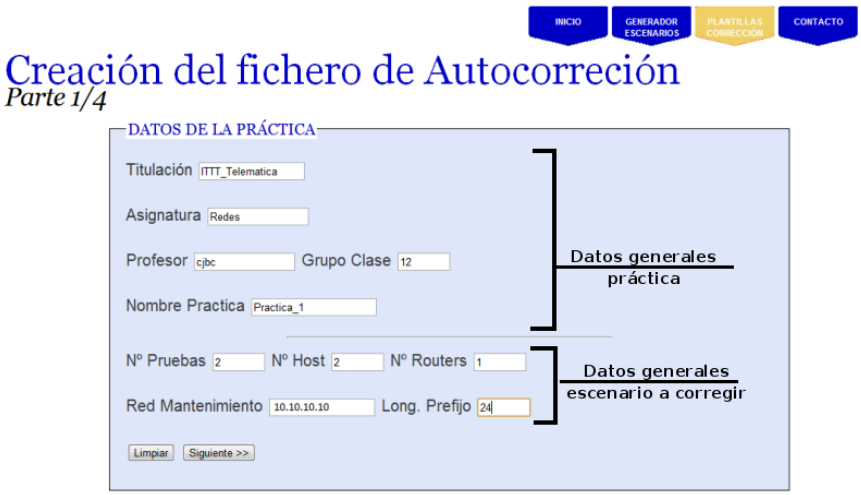


Figura 7.8: Parte 1 del formulario. Rellenamos datos referentes al número de nodos que intervienen en la simulación.

2. **Parte 2**. Datos del escenario y tipos de pruebas a aplicar sobre él. En los datos del escenario se deben rellenar las IPs que debe tener cada interfaz de cada nodo del escenario según el enunciado de la práctica propuesta. En los interfaces donde no haya que configurar nada, podemos

dejar el campo en blanco. En la figura 7.9 podemos ver los datos que se han rellenado para nuestro ejemplo.

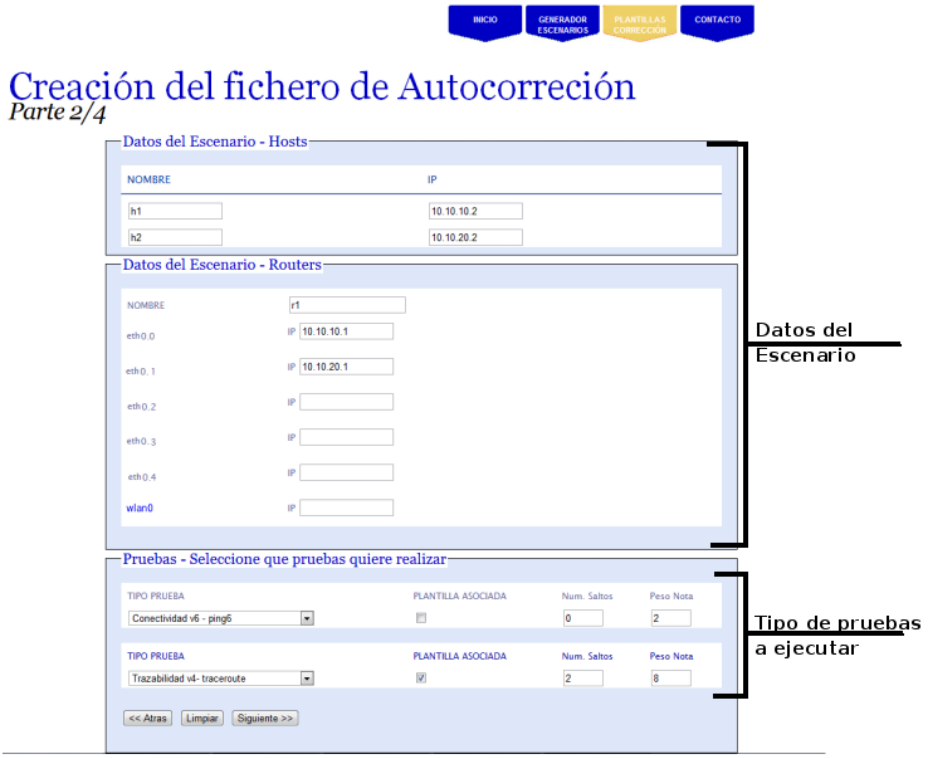


Figura 7.9: Parte 2 del formulario. Rellenamos datos referentes a la configuración de IPs del escenario y los tipos de pruebas a ejecutar.

3. **Parte 3**. Datos de las pruebas. En esta parte, se configuran las pruebas que se han definido en el paso anterior. En la figura 7.10 podemos ver los datos que se han rellenado para nuestro ejemplo. Obsérvese que para el caso de la prueba de traceroute, hay que rellenar los datos de la plantilla asociada (que serán los nodos por los que pasen los paquetes desde un origen a un destino determinado). Una vez generado el XML podemos guardarlo en la ruta que deseemos pinchando en la opción guardar del navegador que estemos utilizando.

Figura 7.10: Parte 3 del formulario. Rellenamos datos referentes a la configuración de cada prueba.

7.4. CONCLUSIONES

En este capítulo hemos visto cuales han sido las motivaciones que nos han llevado desarrollar esta herramienta auxiliar y como se utiliza.

Hemos visto como la herramienta desarrollada permite crear los ficheros XML para nuestros entornos de una forma sencilla, rápida e intuitiva. Gracias a este portal web desarrollado, nos aseguramos de que los ficheros XML que se crean no tienen ningún fallo de formato, ni de inconsistencia respecto a su DTD.

Gracias a este entorno, la persona que desee crear un escenario y/o una corrección, no necesitará tener conocimientos acerca de los XML en general ni de como tienen que definirse estos en concreto. El portal web crea una capa intermedia permitiendo al usuario no tener que llegar a tan bajo nivel para realizar los ficheros de configuración.

Capítulo 8

VALIDACIÓN Y PRUEBAS REALIZADAS

8.1. INTRODUCCIÓN

En este capítulo se expondrán los resultado obtenidos en la pruebas realizadas y los escenarios en los que han sido probados los entornos. Se realizará una comparativa entre, el sistema anterior (el entorno real) y el sistema desarrollado (entorno virtual con autocorrección). Esta comparativa se centra en el coste económico de los entornos y en la diferencia de tiempo que se emplea cuando se utiliza uno u otro.

8.2. VALIDACIÓN DE LOS ENTORNOS

Dado que el sistema desarrollado está dividido en tres partes bien diferenciadas, vamos a exponer los resultados obtenidos para cada una de ellas, ya que cada una de las partes ha sido probada y validada de forma diferente.

El entorno virtual fue el primero que se desarrolló y ha sido el más probado y validado con usuarios reales. Este entorno virtual ha sido instalado en los Laboratorios del Departamento de Ingeniería Telemática y ha sido utilizado por los alumnos de la Universidad para la realización de ejercicios. En un primer momento, se brindó la posibilidad de utilizarlo de forma opcional y una vez probado por los alumnos, se comenzó a implantar poco a poco en las distintas asignaturas, evaluando la satisfacción de los alumnos y profesores a través de los cuestionarios y de las encuestas de evaluación docente que se realizan tras cursar todas las asignaturas de la Universidad Carlos III de Madrid.

En la actualidad, incluso se está utilizando el entorno virtual para la realización de exámenes prácticos, en los cuales, o bien se les pide un pequeño desarrollo y se les ofrece la posibilidad de probarlo en el entorno virtual, para que el alumno pueda saber si el desarrollo realizado es bueno o tiene que ser corregido antes de entregarlo, o bien, se les pide hacer una configuración de red en un escenario dado.

El uso del entorno virtual permite a los alumnos practicar con los nodos simulados (routers y PCs del Laboratorio) y adquirir cierta soltura en el manejo de éstos. Esto les permite mejorar la ejecución de sus prácticas con el entorno físico y mejorar sus calificaciones. En concreto, se hizo un estudio en

el curso 2007/2008 en el cual se propuso un ejercicio opcional que realizó el 46.5 %. Los alumnos que realizaron el ejercicio mejoraron su nota media en un 8.5 %. Este estudio y sus resultados están recogidos en un articulo publicado en el 2009 con el título "Computer networking teaching experiences using COTS routers and virtual environments: the UC3M Laboratory" [13], que se presentó en el ICERI2009 (Conferencia Internacional de Educación, Investigación e Innovación) como póster.

Durante este curso académico, 2011/2012, esta herramienta ha sido utilizada por 325 alumnos. El número de alumnos que utilizan la herramienta se distribuye entre las distintas titulaciones de telecomunicaciones como se puede ver en la gráfica 8.1.

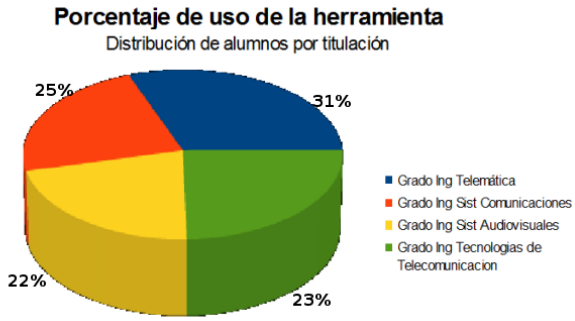


Figura 8.1: Distribución de alumnos que han utilizado el entorno virtual en el curso 2011/2012

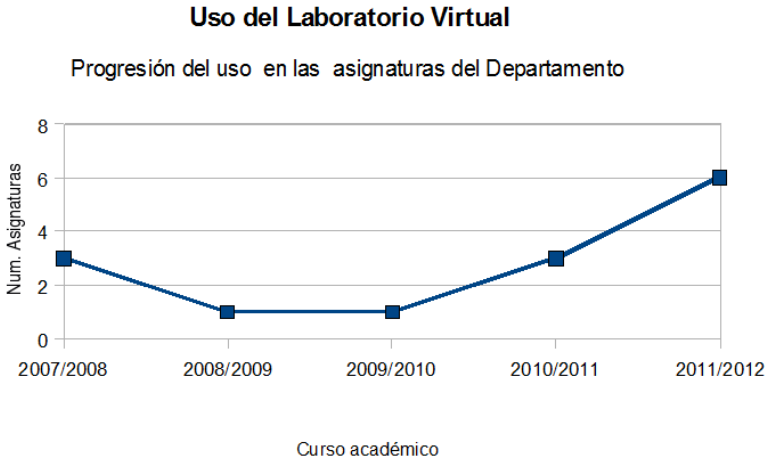


Figura 8.2: Progresión del número de asignaturas que incluyen el uso del Laboratorio Virtual como herramienta para sus prácticas

En la figura 8.2 vemos cual ha sido la progresión de uso de esta herramienta en las distintas asignaturas cuyas prácticas se imparten en los Laboratorios del Departamento de Ingeniería Telemática y

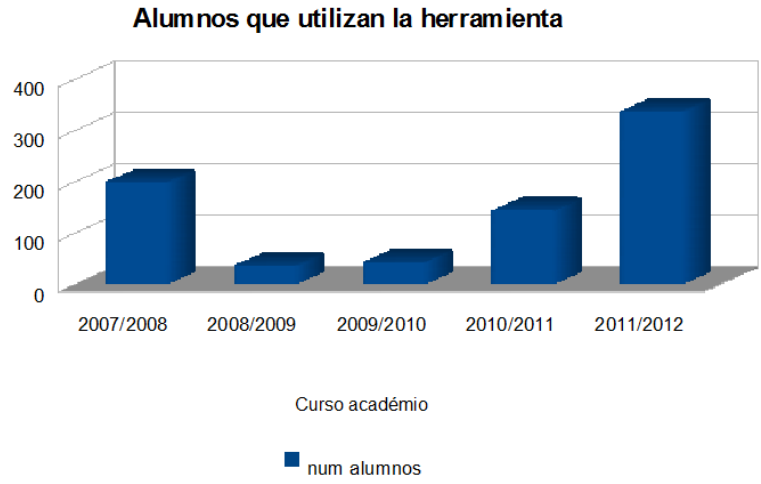


Figura 8.3: Gráfica del volumen de alumnos que utilizan el entorno virtual

en el gráfico 8.3 podemos ver el volumen de alumnos que han utilizado este entorno.

Como podemos ver, durante el primer año se fomento mucho el uso de este entorno, esto es debido a que durante ese año, participamos en un Proyecto de Innovación Docente titulado “Laboratorio virtual: construyendo redes de comunicaciones en un PC”, el cual estaba subvencionado con 1000 euros para material informático y un becario.

Para el entorno de autocorrección hemos realizado exhaustivas pruebas, comprobando su correcto funcionamiento. Se han realizado pruebas en diferentes escenarios, de diferente complejidad en los cuales se han aplicado cada una de las pruebas de corrección que hemos definido. Los escenarios que se han utilizado para aplicar estas pruebas son tanto escenarios creado por nosotros mismos, como escenarios que se han propuesto en las prácticas de configuración de redes a los alumnos en diferentes asignaturas. En la tabla 8.1 se observan los escenarios utilizados, la complejidad de cada uno de ellos (número de nodos que componen la topología) y los tipos de pruebas de corrección aplicadas sobre el escenario.

Hay que resaltar que aún no hemos desplegado el entorno de autocorrección en los Laboratorios del Departamento, por lo que aún no tenemos datos de uso real de esta aplicación. Se pretende implantarla poco a poco en las prácticas de redes, primero de forma optativa y para posteriormente empezar a utilizarlo de forma habitual, tal y como se hizo con el entorno virtual.

Escenarios de validación				
Núm hosts	**Núm routers**	**Pruebas aplicadas**	**Utilizado en prácticas**	**Comentarios**
2	1	Conectividad Trazabilidad Robustez	No	Se comprueban las pruebas básicas
2	1	Conectividad IPv6 Trazabilidad IPv6 Robustez IPv6	No	Se comprueban las pruebas básicas para IPv6
2	2	Conectividad Trazabilidad Robustez Información	"Interconexión de equipos" (curso 2007/2008)	Se comprueban las pruebas básicas en un escenario de prácticas
2	6	Conectividad Trazabilidad BGP BGP Summary Información	"Tendencias en Sistemas Telemáticos" (curso 2007/ 2008)	Se comprueban las pruebas de BGP en un escenario de prácticas
5	9	Conectividad Trazabilidad BGP BGP Summary Información Robustez	"Tendencias en Sistemas Telemáticos" (curso 2010/ 2011)	Se comprueban las pruebas de BGP y la convergencia de rutas en un escenario de prácticas
2	4	Conectividad Trazabilidad Robustez Información	No	Se comprueban el funcionamiento de la prueba de robustez [1]
2	3	Conectividad Trazabilidad Robustez Información	No	Se comprueban el funcionamiento de la prueba de robustez [1]

Tabla 8.1: Escenarios en los que hemos comprobado que las pruebas de corrección funcionan correctamente.

Se han realizado pruebas sobre el script "corregirLabUC3M.sh", para comprobar su funcionamiento en caso de que el alumno que corrija los ejercicios introduzca mal los datos solicitados.

El carácter novedoso del entorno de autocorrección nos ha permitido enmarcarlo dentro de otro Proyecto de Innovación Docente titulado "Corrección automática de ejercicios prácticos de redes de comunicaciones", que tuvo lugar durante el curso 2009-2010.

Por último, para validar la herramienta web que genera los XML, hemos utilizado el validador paginas HTML de la World Wide Web Consorcium (W3C) para asegurarnos de forma objetiva

[1] En estos escenarios se configuraron los protocolos de RIP y OSPF para permitir que el escenario convergiera en caso de que se produjese algún problema en alguno de los enlaces.

que nuestra herramienta sigue los estándares de definidos por este organismo. Tras evaluar nuestra herramienta web hemos obtenido los siguientes resultados:

- Test CSS de nivel 3 pasado correctamente 8.4. Este test certifica que la hoja de estilo hemos utilizado sigue las restricciones de estilo de nivel 3 (nivel más exigente).

Figura 8.4: Resultado de la validación del CSS

- Test XHTML 1.0 Strict pasado correctamente 8.5. Este test certifica que nuestra página web sigue las normas del estándar XHTML 1.0 Strict.

Figura 8.5: Resultado de la validación de la herramienta web.

Hemos comprobado que el uso de la web permite generar XMLs perfectamente bien formados en el 100 % de los casos, ya que la estructura del fichero es generada por la herramienta y el usuario simplemente elige los datos concretos de la simulación y/o la corrección.

8.3. COMPARATIVA

Las prácticas realizadas en el Laboratorio, con el entorno físico se suelen realizar en grupos de 2 o 3 personas. Cada grupo tiene que configurar un escenario que puede tener, como máximo, al rededor de 8 nodos físicos (entre PCs y routers involucrados en la topología). En el entorno virtual, cada nodo es una máquina virtual, por lo que el coste de uno y otro entorno son muy diferentes. En las tablas 8.2 y 8.3 aparece una estimación de la diferencia de coste entre los dos entornos.

Precio de nodos en el entorno real			
Nodo	**Núm nodos**	**Precio unidad (euros)**	**Precio total (euros)**
Router Linksys	5	47,90	240
PCs laboratorio	3	600	1800
TOTAL			2040

Tabla 8.2: Coste de los nodos físicos requeridos para realizar una práctica en el Laboratorio con el entorno real.

Precio de nodos en el entorno virtual			
Nodo	**Núm nodos**	**Precio unidad (euros)**	**Precio total (euros)**
PCs laboratorio	1	600	600
TOTAL			600

Tabla 8.3: Coste de los nodos físicos necesarios para realizar una práctica utilizando el entorno virtual.

El tiempo que los alumnos tardan en configurar el entorno real es superior al tiempo que se tarda en configurar el entorno virtual. Esto es porque en el entorno físico real, hay que realizar las conexiones físicas entre los equipos que forman nuestro escenario y hay que realizar configuraciones de red que nos permitan acceder a cada nodo para proceder a realizar el ejercicio solicitado. Sin embargo, en el entorno virtual, al lanzar la simulación los alumnos pueden acceder directamente a cada nodo.

El uso del entorno de autocorrección permite que se puedan corregir N prácticas a la vez, de forma concurrente, sin embargo, con el entorno real, el profesor de prácticas sólo puede corregir una práctica cada vez, haciendo que la corrección de todos los grupos sea más lenta y generándose colas de espera entre los grupos de prácticas. Además, cuando el profesor realiza la corrección en el entorno físico, si el resultado de ésta no es correcto, se suele perder más tiempo, ya que además de realizar la corrección, también revisa la configuración para detectar dónde puede estar el fallo. Esto hace que el tiempo de corrección con un grupo se pueda alargar más de la cuenta, haciendo que la espera entre distintos grupos se mayor. Con el entorno de autocorrección el tiempo que se emplea en corregir una práctica no se alarga, ya que se realizan las pruebas y se obtienen los resultados. Si las pruebas no son satisfactorias, es tarea del alumno comprobar que parte de la configuración hay que corregir.

Hemos comparado el tiempo que se tarda en generar un fichero XML para cada entorno de forma manual y el tiempo que tarda si utilizamos la herramienta web que hemos creado. Según el número de nodos que formen el escenario de red que queremos generar y según el número de pruebas que queramos definir para la corrección de un escenario, emplearemos más o menos tiempo ya que la complejidad del fichero es directamente proporcional al número de nodos y número de pruebas a definir.

Comparativa para ficheros del entorno virtual	
Fichero generado de forma manual	**Fichero generado con la herramienta web**
39’30”	15’

Tabla 8.4: Comparativa del tiempo empleado en generar un fichero XML de configuración para el entorno virtual.

En la tabla 8.4, vemos una comparativa del tiempo que se emplea para generar un fichero XML de configuración del entorno virtual en el que se definen 10 nodos para una topología.

En la tabla 8.5 vemos la comparativa que hemos hecho del tiempo que se emplea para generar el fichero XML de configuración para el entorno de autocorrección, en el cual se han definido 8 pruebas para un escenario compuesto por 6 nodos, es decir, un escenario de complejidad media-alta.

Comparativa para ficheros del entorno autocorrección	
Fichero generado de forma manual	**Fichero generado con la herramienta web**
53'15"	15'20"

Tabla 8.5: Comparativa del tiempo empleado en generar un fichero XML de configuración para el entorno autocorrección.

Capítulo 9

CONCLUSIONES Y TRABAJOS FUTUROS

9.1. INTRODUCCIÓN

En este capítulo, se presentan unas conclusiones finales basadas en el desarrollo global del trabajo y cuáles pueden ser los trabajos futuros para mejorar todo lo desarrollado y proporcionar mayor funcionalidad al sistema.

9.2. CONCLUSIONES

Se ha proporcionado una solución tecnológica que engloba todo lo necesario para realizar prácticas y/o ejercicios de configuración de redes sin la necesidad de utilizar nodos físicos que además, es capaz de auto-corregirse basándose en las plantillas de corrección definidas previamente por el profesor.

La solución tecnológica implementada consta de tres partes: un entorno virtual que simula los escenarios de red, un entorno de autocorrección que evalúa las soluciones planteadas por los alumnos y una herramienta web que genera de forma sencilla los ficheros de configuración para los dos entornos anteriores.

Se ha conseguido tener un entorno virtual que imita a la perfección los nodos físicos (routers Linksys y PCs) del Laboratorio del Departamento de Ingeniería Telemática.

Nuestro entorno virtual permite la realización de las prácticas de configuración de redes y desarrollo de protocolos que se realizan en el Departamento. Los alumnos que lo utilicen pueden aplicar en el entorno virtual (comandos, scripts, etc.) todo lo que se aplica en el entorno real. Hay que destacar que aunque este entorno virtual funciona imitando a la perfección los nodos físicos, y permitiría un ahorro considerable de material docente, no se desea en ningún momento sustituir las prácticas que se realizan con el entorno real por nuestro entorno virtual, pero sí queremos que sea una herramienta complementaria para el alumno.

La experiencia de la implantación de este entorno virtual en las asignaturas del Departamento ha sido positiva, tanto para los alumnos como para los profesores, una prueba de esto, es que el

número de asignaturas que se han decidido a usarlo ha aumentado desde que se comenzó a implantar en el segundo cuatrimestre del año 2007, al igual que el número de alumnos que lo utilizan. Este entorno virtual permite que el aprendizaje de los alumnos no se limite al tiempo de las sesiones en los Laboratorios, y que puedan seguir practicando con los nodos que simulan los routers en cualquier momento. Gracias a esto, los alumnos cogen más soltura en el manejo de los equipos para realizar sus prácticas y esto repercute directamente en su aprendizaje (lo cual se ve reflejado en unas mejores calificaciones).

El entorno de autocorrección que se ha implementado permite corregir de forma automática las configuraciones realizadas por los alumnos en el entorno virtual, liberando al profesorado de cierta carga de trabajo. Con el método de evaluación tradicional, el profesor tiene que corregir uno a uno cada escenario configurado "in-situ", es decir, durante el tiempo que dura la práctica. Esto provoca cierto estrés, tanto al profesor como a los alumnos. Hay que tener en cuenta que durante el tiempo que el profesor está atendiendo a un grupo de prácticas no puede atender a otro, y por lo tanto se generan colas de espera entre los grupos de prácticas. El entorno de autocorrección permite que el trabajo de corregir ejercicios sea más fluido, ya que es ejecutado por los propios alumnos, enviándoles a estos un feed-back que les permite corregir sus propias configuraciones, en caso de que éstas no sean correctas, antes de entregar las prácticas. Por este motivo, podemos concluir que el uso del entorno de autocorrección fomenta el auto-aprendizaje del alumno.

Dado que nuestro entorno de autocorrección está pensada para el uso docente, hemos generado un pequeño módulo de seguridad que nos permite detectar copias y manipulación de prácticas. De esta forma el profesorado puede "confiar" en la herramienta y permitir que se corrijan los ejercicios a través de este nuevo método.

Hemos generado una herramienta web, de uso interno, sólo para profesores, que permite generar los escenarios de red y las plantillas de corrección de forma sencilla, rápida y robusta.

Dado el carácter de este trabajo, hemos podido enmarcarlo dentro en dos proyectos de innovación docente reconocidos por la Universidad Carlos III de Madrid: 'Laboratorio virtual: construyendo redes de comunicaciones en un PC" (2007-2008), el cual estaba subvencionado, y 'Corrección automática de ejercicios prácticos de redes de comunicaciones" (2009-2010). Además, ha sido la base de una publicación docente: 'Computer networking teaching experiences using COTS routers and virtual environments: the UC3M Laboratory" que se presentó en el ICERI2009 (Conferencia Internacional de Educación, Investigación e Innovación). La herramienta web que hemos diseñado y desarrollado también ha sido la base de un artículo que ha sido enviado recientemente a un congreso.

9.3. TRABAJOS FUTUROS

Una vez desarrollado el sistema completo, hemos detectado algunos puntos de mejora que pueden impulsar la creación de nuevos proyectos fin de carrera/ trabajos fin de grado. A continuación se enumerarán las posibles líneas futuras, que permitirán mejorar el servicio que se ofrece con este sistema.

- Un primer trabajo futuro inmediato (que no sería para un proyecto fin de carrera o trabajo fin de grado) es poner en funcionamiento el entorno de autocorrección con usuarios reales.
- Gracias al entorno virtual podemos practicar las configuraciones y comandos de los nodos físicos, sin embargo, no podemos practicar la interconexión de equipos. Podría implementarse

una herramienta gráfica que complementara al entorno virtual y que simulara los interfaces de cada nodo y permitiera elegir a alumno en cual hacer las interconexiones.

- Otra posible mejora que se puede aplicar al entorno desarrollado es adaptarlo para su uso fuera de los Laboratorios del Departamento. Para ello se podría incluir el software en una distribución tipo CD Live o generar un paquete para su instalación en deversas distribuciones Linux. De esta forma, el alumno podría utilizar el entorno virtual no sólo en los Laboratorios del Departamento, sino en cualquier parte.
- El sistema de autocorrección que hemos creado permite corregir ejercicios realizados con el entorno virtual. Sería interesante que el entorno de autocorrección permitiese corregir ejercicios y/o prácticas realizadas en el entorno real con los nodos físicos.
- En la actualidad, el entorno de autocorrección permite corregir ejercicios en los cuales se fuercen unas IPs determinadas en cada nodo. Sería útil que la autocorrección permitiese configurar en cada nodo una IP cualquiera de un rango determinado. De esta forma los alumnos tendrían mayor libertad a la hora de hacer sus prácticas.
- Dado que el número de asignaturas que utilizan el entorno ha aumentado y hay un mayor número de prácticas que se realizan con este entorno, sería interesante implementar un gestor de prácticas y correcciones, que permitiese almacenar, buscar o modificar prácticas anteriores.

Bibliografía

[1] «Apache2 web oficial». Última consulta: Junio de 2012.
http://httpd.apache.org

[2] «JAVA web oficial». Última consulta: Julio de 2012.
http://www.oracle.com/technetwork/java/index.html

[3] «JDOM web oficial». Última consulta: Julio de 2012.
http://www.jdom.org

[4] «PHP web oficial». Última consulta: Julio de 2012.
http://php.net/index.php

[5] «Quagga web oficial». Última consulta: Junio de 2012.
http://www.quagga.net

[6] «QUEMU web oficial». Última consulta: Mayo de 2012.
http://www.quemu.org

[7] «UML web oficial». Última consulta: Abril de 2012.
http://www.uml.org

[8] «VirtualBox web oficial». Última consulta: Abril de 2012.
https://www.virtualbox.org

[9] «VMWare web oficial». Última consulta: Abril de 2012.
http://www.vmware.com

[10] «VNUML web oficial». Última consulta: Mayo de 2012.
http://www.dit.upm.es/vnml

[11] «Manual Linksys de configuración del módelo del router Linksys WRT54GS/GL», 2008. Última consulta: Julio de 2012.
http://www.it.uc3m.es/linksys/files/manual_linksys_uc3m.pdf

[12] «Xen web oficial», 2012.
http://www.xen.org

[13] BERNARDOS, CARLOS JESÚS; MARTÍNEZ, ALBERTO GARCÍA y DURÁN, CELESTE: «Computer networking teaching experiences using cots routers and virtual enviroments: the UC3M laboratory». ICERI2009, 2009.

[14] BURKE, ERIC M.: Java and XSLT. O'Reilly, 2001.

[15] GALÁN MÁRQUEZ, FERMÍN: «Tecnología XML en la herramienta de simulación de redes vnuml», 2004.

[16] LOPERA MORLAS, FRANCISCO: «PGP y GnuPGP Teoría y práctica», 2001.

[17] TANENBAUM, ANDREW S.: Redes de Computadoras. Prentice-Hall, 1998.

Apéndice A

DTDs y XMLs

A.1. DTD VNUML

```
    <!– VNUML DTD version 1.8 –>
<!ELEMENT vnuml (global,net*,vm*,host?)>
<!ELEMENT global (version,simulation_name,ssh_version?,ssh_key*,automac?,netconfig?,vm_mgmt?,
tun_device?,vm_defaults?)>
<!ELEMENT vm_defaults (filesystem?,mem?,kernel?,shell?,basedir?,
mng_if?,console*,xterm?,route*,forwarding?,user*,filetree*)>
<!ATTLIST vm_defaults exec_mode (net|mconsole|pst) ”net”>
<!ELEMENT net (bw?)>
<!ATTLIST net name CDATA #REQUIRED
mode (virtual_bridge|uml_switch) #REQUIRED
sock CDATA #IMPLIED
type (lan|ppp) ”lan”
external CDATA #IMPLIED
vlan CDATA #IMPLIED
uml_switch_binary CDATA #IMPLIED
hub (yes|no) ”no”
scope (shared|no-shared) ”no-shared”
capture_file CDATA #IMPLIED
capture_expression CDATA #IMPLIED
capture_dev CDATA #IMPLIED>
<!ELEMENT vm (filesystem?,mem?,kernel?,shell?,basedir?,mng_if?,console*,xterm?,if*,
route*,forwarding?,user*,filetree*,exec*)>
<!ATTLIST vm name CDATA #REQUIRED
order CDATA #IMPLIED>
<!ELEMENT host (hostif*,physicalif*,route*,forwarding?,exec*)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT simulation_name (#PCDATA)>
<!ELEMENT ssh_version (#PCDATA)>
<!ELEMENT ssh_key (#PCDATA)>
<!ELEMENT automac EMPTY>
<!ATTLIST automac offset CDATA ”0”>
<!ELEMENT netconfig EMPTY>
```

<!ATTLIST netconfig stp (on|off) ”off”
promisc (on|off) ”on”>
<!ELEMENT vm_mgmt (mgmt_net?,host_mapping?)>
<!ATTLIST vm_mgmt type (private|net|none) #REQUIRED
network CDATA ”192.168.0.0”
mask CDATA ”24”
offset CDATA ”0”>
<!ELEMENT mgmt_net EMPTY>
<!ATTLIST mgmt_net sock CDATA #REQUIRED
hostip CDATA #REQUIRED
autoconfigure CDATA #IMPLIED>
<!ELEMENT host_mapping EMPTY>
<!ELEMENT shell (#PCDATA)>
<!ELEMENT tun_device (#PCDATA)>
<!ELEMENT basedir (#PCDATA)>
<!ELEMENT bw (#PCDATA)>
<!ELEMENT filesystem (#PCDATA)>
<!ATTLIST filesystem type (direct|cow|hostfs) #REQUIRED>
<!ELEMENT mem (#PCDATA)>
<!ELEMENT kernel (#PCDATA)>
<!ATTLIST kernel initrd CDATA #IMPLIED
devfs (mount|nomount) #IMPLIED
root CDATA #IMPLIED
modules CDATA #IMPLIED
trace (on|off) ”off”>
<!ELEMENT mng_if (#PCDATA)>
<!ELEMENT console (#PCDATA)>
<!ATTLIST console id CDATA #REQUIRED>
<!ELEMENT if (mac?,ipv4*,ipv6*)>
<!ATTLIST if id CDATA #REQUIRED
net CDATA #REQUIRED>
<!ELEMENT route (#PCDATA)>
<!ATTLIST route type (ipv4|ipv6) #REQUIRED
gw CDATA #REQUIRED>
<!ELEMENT forwarding EMPTY>
<!ATTLIST forwarding type (ip|ipv4|ipv6) ”ip”>
<!ELEMENT user (group*,ssh_key*)>
<!ATTLIST user username CDATA #REQUIRED
group CDATA #IMPLIED>
<!ELEMENT group (#PCDATA)>
<!ELEMENT filetree (#PCDATA)>
<!ATTLIST filetree root CDATA #REQUIRED
seq CDATA #REQUIRED
user CDATA #IMPLIED
mode (net|mconsole|pst) ”net”>
<!ELEMENT exec (#PCDATA)>
<!ATTLIST exec type (verbatim|file) #REQUIRED

seq CDATA #REQUIRED
user CDATA #IMPLIED
mode (net|mconsole|pst) ”net”>
<!ELEMENT hostif (ipv4*,ipv6*)>
<!ATTLIST hostif net CDATA #REQUIRED>
<!ELEMENT mac (#PCDATA)>
<!ELEMENT ipv4 (#PCDATA)>
<!ATTLIST ipv4 mask CDATA ”255.255.255.0”>
<!ELEMENT ipv6 (#PCDATA)>
<!ATTLIST ipv6 mask CDATA #IMPLIED>
<!ELEMENT physicalif EMPTY>
<!ATTLIST physicalif name CDATA #REQUIRED
type (ipv4|ipv6) ”ipv4”
ip CDATA #REQUIRED
mask CDATA #IMPLIED
gw CDATA #IMPLIED>
<!ELEMENT xterm (#PCDATA)>

A.2. Ejemplo XML de un escenario

<?xml version=”1.0” encoding=”UTF-8” ?>
<!DOCTYPE vnuml SYSTEM /usr/share/xml/vnuml/vnuml.dtd”>

<vnuml>
<global>
<version>1.8</version>
<simulation_name>TST-10</simulation_name>
<automac/>
<vm_mgmt type=”net”network=”10.255.255.0”mask=”24”>
<mgmt_net sock=/var/run/vnuml/RedAux.ctl”hostip=”10.255.255.20” autoconfigure=”tap0”/>
</vm_mgmt>
<vm_defaults exec_mode=”net”>
<kernel>/usr/share/vnuml/kernels/linux-2.6.18.1-bb2-xt-4m</kernel>
<console id=”0”>xterm</console>
</vm_defaults>
</global>
<!– Redes del AS Grupo_X y Grupo_Y–>
<net name=”RedX_a”mode=”uml_switch” />
<net name=”RedX_b”mode=”uml_switch” />
<net name=”RedY_a”mode=”uml_switch” />
<net name=”RedY_b”mode=”uml_switch” />
<net name=”RdX_RdY”mode=”uml_switch” />
<net name=”RXb_RXa”mode=”uml_switch” />
<net name=”RYb_RYa”mode=”uml_switch” />
<net name=”RdXRdY2”mode=”uml_switch” />

```
    <!– Redes de los AS Grupo_X/Grupo_Y con los Proveedores A y B–>
<net name=”RXbRPXb”mode=”uml_switch” />
<net name=”RXaRPXa”mode=”uml_switch” />
<net name=”RYbRPYb”mode=”uml_switch” />
<net name=”RYaRPYa”mode=”uml_switch” />

    <!– Redes entre los Proveedores A y B de X y los Proveedores A y B de Y–>
<net name=”RPb_XY”mode=”uml_switch” />
<net name=”RPa_XY”mode=”uml_switch” />

    <!– Redes entre los Proveedores y R_TST–>
<net name=”R_Pract”mode=”uml_switch” />

    <!– Redes del R_TST–>
<net name=”TST”mode=”uml_switch” />

    <!– Interfaces sin conexion–>
<net name=”ninguna”mode=”uml_switch” />

    <!–Los host implicados en la topología son los siguiente –>
<vm name=”hstXa”>
<filesystem type=”cow”>/usr/share/vnuml/filesystems/nuevoHost.ext3</filesystem>
<if id=”1”net=”RedX_a”>
<ipv4>192.100.100.1</ipv4>
</if>
<route type=”ipv4”gw=”192.100.100.2”>default</route>
</vm>
<vm name=”hstXb”>
<filesystem type=”cow”>/usr/share/vnuml/filesystems/nuevoHost.ext3</filesystem>
<if id=”1”net=”RedX_b”>
<ipv4>192.100.100.1</ipv4>
</if>
<route type=”ipv4”gw=”192.100.100.2”>default</route>
</vm>

    <vm name=”hstYa”>
<filesystem type=”cow”>
/usr/share/vnuml/filesystems/nuevoHost.ext3</filesystem>
<if id=”1”net=”RedY_a”>
<ipv4>192.100.100.1</ipv4>
</if>
<route type=”ipv4”gw=”192.100.100.2”>default</route>
</vm>
<vm name=”hstYb”>
<filesystem type=”cow”>/usr/share/vnuml/filesystems/nuevoHost.ext3</filesystem>
<if id=”1”net=”RedY_b”>
<ipv4>192.100.100.1</ipv4>
```

```
</if>
<route type="ipv4"gw="192.100.100.2">default</route>
</vm>
<vm name="hstTST">
<filesystem type="cow">/usr/share/vnuml/filesystems/nuevoHost.ext3</filesystem>
<if id="1"net="TST">
<ipv4>10.0.200.1</ipv4>
</if>
<route type="ipv4"gw="10.0.200.2">default</route>
</vm>

    <!–A continuación se representan los routers que forman la topologia –>
<vm name="R_X_a">
<filesystem type="cow">/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id="1"net="RedX_a">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="2"net="RXaRPXa">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="3"net="RdX_RdY">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="4"net="RXb_RXa">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="5"net="ninguna">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="6"net="ninguna">
<ipv4>192.168.5.1</ipv4>
</if>
<forwarding type="ip" />
</vm>

    <vm name="R_X_b">
<filesystem type="cow">/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id="1"net="RedX_b">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="2"net="RXbRPXb">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="3"net="RXb_RXa">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="4"net="ninguna">
```

<ipv4>
192.168.5.1</ipv4>
</if>
<if id=”5”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”6”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<forwarding type=”ip” />
</vm>

<vm name=”R_Y_a”>
<filesystem type=”cow”>/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id=”1”net=”RedY_a”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”2”net=”RYaRPYa”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”3”net=”RdX_RdY”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”4”net=”RYb_RYa”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”5”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”6”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<forwarding type=”ip” />
</vm>

<vm name=”R_Y_b”>
<filesystem type=”cow”>/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id=”1”net=”RedY_b”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”2”net=”RYbRPYb”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”3”net=”RYb_RYa”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”4”net=”ninguna”>

<ipv4>
192.168.5.1</ipv4>
</if>
<if id=”5”net=”ninguna”>
<ipv4>
192.168.5.1</ipv4>
</if>
<if id=”6”net=”ninguna”>
<ipv4>
192.168.5.1</ipv4>
</if>
<forwarding type=”ip” />
</vm>

<vm name=”RPr_B_X”>
<filesystem type=”cow”>/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id=”1”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”2”net=”R_Pract”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”3”net=”RXbRPXb”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”4”net=”RPb_XY”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”5”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”6”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<forwarding type=”ip” />
</vm>

<vm name=”RPr_A_X”>
<filesystem type=”cow”>/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id=”1”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”2”net=”R_Pract”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”3”net=”RXaRPXa”>
<ipv4>192.168.5.1</ipv4>

```
</if>
<if id="4"net="RPa_XY">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="5"net="ninguna">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="6"net="ninguna">
<ipv4>192.168.5.1</ipv4>
</if>
<forwarding type="ip" />
</vm>

    <vm name="RPr_B_Y">
<filesystem type="cow">/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id="1"net="ninguna">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="2"net="R_Pract">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="3"net="RYbRPYb">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="4"net="RPb_XY">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="5"net="ninguna">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="6"net="ninguna">
<ipv4>192.168.5.1</ipv4>
</if>
<forwarding type="ip" />
</vm>

    <vm name="RPr_A_Y">
<filesystem type="cow">/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id="1"net="ninguna">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="2"net="R_Pract">
<ipv4>192.168.5.1</ipv4>
</if>
<if id="3"net="RYaRPYa">
<ipv4>192.168.5.1</ipv4>
</if>
```

<if id=”4”net=”RPa_XY”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”5”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”6”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<forwarding type=”ip” />
</vm>

<vm name=”R_TST”>
<filesystem type=”cow”>/usr/share/vnuml/filesystems/imagenRouter.ext3</filesystem>
<if id=”1”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”2”net=”R_Pract”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”3”net=”TST”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”4”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”5”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<if id=”6”net=”ninguna”>
<ipv4>192.168.5.1</ipv4>
</if>
<forwarding type=”ip” />
</vm>

</vnuml>

A.3. DTD AUTOCORRECCIÓN

<!ELEMENT autocorreccion (DatosPractica, DatosControl*, DatosEscenario*, DatosAlumnos*, Prueba*, Resultado*, DatosNotas*) >

<!ELEMENT DatosPractica (Titulacion, Asignatura, Profesor, GrupoClase, NombrePractica)>
<!ELEMENT Titulacion (#PCDATA)>
<!ELEMENT Asignatura (#PCDATA)>
<!ELEMENT Profesor (#PCDATA)>

```
<!ELEMENT GrupoClase (#PCDATA) >
<!ELEMENT NombrePractica (#PCDATA)>

<!ELEMENT DatosControl (Fecha, ip, pid, Hostname)>
<!ELEMENT Fecha (#PCDATA)>
<!ELEMENT ip (#PCDATA)>
<!ELEMENT pid (#PCDATA)>
<!ELEMENT Hostname (#PCDATA)>

<!ELEMENT DatosEscenario (RedMantenimiento,PlantillaAsociada,NumEquipos,Equipo+)>
<!ELEMENT RedMantenimiento (#PCDATA)>
<!ELEMENT PlantillaAsociada (#PCDATA)>
<!ELEMENT NumEquipos (#PCDATA)>
<!ELEMENT Equipo (ip)>
<!ATTLIST Equipo
tipo ID #REQUIRED
nombre ID #REQUIRED>
<!ELEMENT ip (#PCDATA)>
<!ATTLIST ip
dev ID #REQUIRED>

<!ELEMENT DatosAlumnos (NumeroIntegrantes, (Alumno)+)>
<!ELEMENT NumeroIntegrantes (#PCDATA)>
<!ELEMENT Alumno (Nombre,Apellido1,Apellido2,NIA, GrupoPractica)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Apellido1 (#PCDATA)>
<!ELEMENT Apellido2 (#PCDATA)>
<!ELEMENT NIA (#PCDATA)>
<!ELEMENT GrupoPractica (#PCDATA)>

<!ELEMENT Prueba (TipoPrueba)>
<!ATTLIST Prueba
id ID #REQUIRED>
<!ELEMENT TipoPrueba
(PConectividad|PConectividad6 |PTrazabilidad|PTrazabilidad6|
PRobustez|PRobustez6|PInfo|PBgp|PBgpSummary)>
<!ELEMENT PConectividad (Origen, Destino, PesoNota)>
<!ELEMENT PConectividad6 (Origen, Destino, PesoNota)>
<!ELEMENT PTrazabilidad (Origen, Destino,PlantillaCorreccion,PesoNota)>
<!ELEMENT PTrazabilidad6 (Origen, Destino,PlantillaCorreccion,PesoNota)>
<!ELEMENT PRobustez (Origen, Accion, TiempoEspera)>
<!ELEMENT PRobustez6 (Origen, Accion, TiempoEspera)>
<!ELEMENT TiempoEspera (#PCDATA)>
<!ELEMENT PInfo (Origen)>
<!ELEMENT PBgp (Origen,Destino,PlantillaBgp,PesoNota)>
<!ELEMENT PBgpSummary (Origen)>
<!ELEMENT PesoNota (#PCDATA)>
```

<!ELEMENT Origen (#PCDATA)>
<!ATTLIST Origen
tipo ID #REQUIRED
dev ID #IMPLIED>
<!ELEMENT Destino (#PCDATA)>
<!ATTLIST Destino
tipo ID #IMPLIED
dev ID #IMPLIED
real ID #IMPLIED>
<!ELEMENT PlantillaCorreccion (Salto)>
<!ATTLIST Salto
nombre ID #REQUIRED
dev ID #REQUIRED>
<!ELEMENT PlantillaBgp (AS_PATH)>
<!ELEMENT as_path (#PCDATA)>
<!ATTLIST as_path
best ID #REQUIRED>
<!ELEMENT Resultado (TipoResultado)>
<!ATTLIST Resultado
idResultado ID #REQUIRED
idPrueba ID #REQUIRED>
<!ELEMENT TipoResultado
(RConectividad|RConectividad6|RTrazabilidad|RTrazabilidad6|
RRobustez|RRobustez6|RInfo|RBgp|RBgpSummary)>
<!ELEMENT RConectividad (Nota)>
<!ELEMENT RConectividad6 (Nota)>
<!ELEMENT RTrazabilidad (RespuestaTexto,Nota)>
<!ELEMENT RTrazabilidad6 (RespuestaTexto, Nota)>
<!ELEMENT RRobustez (RespuestaTexto)>
<!ELEMENT RRobustez6 (RespuestaTexto)>
<!ELEMENT RInfo (RespuestaTexto)>
<!ELEMENT RBgp (RespuestaTexto, Nota)>
<!ELEMENT RBgpSummary (RespuestaTexto)>
<!ELEMENT RespuestaBinaria (#PCDATA)>
<!ELEMENT RespuestaTexto (#PCDATA)>
<!ELEMENT Nota (#PCDATA)>
<!ELEMENT DatosNostas (NotaTotalPosible*,NotaTotalObtenida*)>
<!ELEMENT NotaTotalPosible (#PCDATA) >
<!ELEMENT NotaTotalObtenida (#PCDATA) >

A.4. Ejemplo XML de un fichero de auto-corrección

<?xml version=”1.0” encoding=”iso-8859-1”?>
<!DOCTYPE autocorreccion SYSTEM /home/mceleste/Proyecto/dtds/autocorreccion.dtd”>
<autocorreccion>

<DatosPractica>
<Titulacion> ITT_TELEMATICA </Titulacion>
<Asignatura> Tendencias_Sistemas_Telematicos </Asignatura>
<Profesor> cjbc </Profesor>
<GrupoClase> 72 </GrupoClase>
<NombrePractica> PruebasProyectoBGP </NombrePractica>
</DatosPractica>
<DatosEscenario>
<NumEquipos>8</NumEquipos>
<Equipo tipo="H"nombre="hstXa">
<ip dev="eth1">10.0.13.2</ip>
</Equipo>
<Equipo tipo="H"nombre="hstXb">
<ip dev="eth1">10.0.13.66</ip>
</Equipo>
<Equipo tipo="H"nombre="hstYa">
<ip dev="eth1">10.0.14.2</ip>
</Equipo>
<Equipo tipo="H"nombre="hstYb">
<ip dev="eth1">10.0.14.66</ip>
</Equipo>

<Equipo tipo="R"nombre="R_X_a">
<ip dev="eth0.0">10.0.13.1</ip>
<ip dev="eth0.1">10.0.100.13</ip>
<ip dev="eth0.2">10.0.200.13</ip>
<ip dev="eth0.3">192.168.10.1</ip>
<ip dev="eth0.4">10.0.13.65</ip>
<ip dev="wlan0">192.168.5.1</ip>
</Equipo>
<Equipo tipo="R"nombre="R_Y_a">
<ip dev="eth0.0">10.0.14.1</ip>
<ip dev="eth0.1">10.0.100.14</ip>
<ip dev="eth0.2">10.0.200.14</ip>
<ip dev="eth0.3">192.168.10.2</ip>
<ip dev="eth0.4">10.0.14.65</ip>
<ip dev="wlan0">192.168.5.1</ip>
</Equipo>
<Equipo tipo="R"nombre="R_TST_1">
<ip dev="eth0.0">192.168.0.1</ip>
<ip dev="eth0.1">10.0.100.101</ip>
<ip dev="eth0.2">192.168.150.1</ip>
<ip dev="eth0.3">192.168.3.1</ip>
<ip dev="eth0.4">192.168.4.1</ip>
<ip dev="wlan0">192.168.5.1</ip>
</Equipo>
<Equipo tipo="R"nombre="R_TST_2">

<ip dev=”eth0.0”>192.168.0.1</ip>
<ip dev=”eth0.1”>10.0.200.102</ip>
<ip dev=”eth0.2”>192.168.150.2</ip>
<ip dev=”eth0.3”>192.168.3.1</ip>
<ip dev=”eth0.4”>192.168.4.1</ip>
<ip dev=”wlan0”>192.168.5.1</ip>
</Equipo>
</DatosEscenario>

<Prueba id=”p1”>
<TipoPrueba>
<PConectividad>
<Origen tipo=”H”>10.10.10.2</Origen>
<Destino>10.10.10.4</Destino>
</PConectividad>
</TipoPrueba>
</Prueba>
<Prueba id=”p2”>
<TipoPrueba>
<PTrazabilidad>
<Origen tipo=”H”>10.10.10.2</Origen>
<Destino>10.10.10.4</Destino>
<PlantillaCorreccion>
<Salto nombre=”R_X_a”dev=.^{et}h0.4”/>
<Salto nombre=”R_TST_2”dev=.^{et}h0.1”/>
<Salto nombre=”R_Y_a”dev=.^{et}h0.2”/>
<Salto nombre=”hstYb”dev=”eth1”/>
</PlantillaCorreccion>
</PTrazabilidad>
</TipoPrueba>
</Prueba>

<Prueba id=”p3”>
<TipoPrueba>
<PBgp>
<Origen tipo=”R”>10.10.10.5</Origen>
<Destino>10.10.10.4</Destino>
<PlantillaBgp>
<as_path best=”true”>65102 65014</as_path>
<as_path best=”false”>65101 65014</as_path>
</PlantillaBgp>
</PBgp>
</TipoPrueba>
</Prueba>
<Prueba id=”p4”>
<TipoPrueba>
<PBgpSummary>

<Origen tipo=”R”>10.10.10.5</Origen>
</PBgpSummary>
</TipoPrueba>
</Prueba>
<Prueba id=”p5”>
<TipoPrueba>
<PInfo>
<Origen tipo=”R”>10.10.10.5</Origen>
</PInfo>
</TipoPrueba>
</Prueba>
</autocorreccion>

Apéndice B

INSTALACIÓN SOFTWARE

B.1. ENTORNO VIRTUAL

B.1.1. INSTALACIÓN VNUML

Siendo súper-usuario (root), se deben seguir los siguientes pasos:

1. Agregamos los repositorios para poder instalar VNUML con apt-get.
   > echo ”deb http://jungla.dit.upm.es/ vnuml/debian binary/”>> /etc/apt/sources.list

2. Descargamos e instalamos
   > apt-get install vnuml

3. Instalamos el sistema de archivos que utiliza el VNUML. En nuestro caso, utilizamos la versión 0.5.0.
   > wget http://ufpr.dl.sourceforge.net/sourceforge/vnuml/root_fs_tutorial-0.5.0.bz2

4. Movemos el archivo descargado donde lo vamos a descomprimir
   > mv root_fs_tutorial-0.5.0.bz2 /usr/share/vnuml/filesystems/
   > cd /usr/share/vnuml/filesystems/
   > bunzip2 root_fs_tutorial-0.5.0.bz2
   > ln -s root_fs_tutorial-0.50 root_fs_tutorial

5. Instalamos el kernel de las máquinas virtuales
   > wget http://ufpr.dl.sourceforge.net/sourceforge/vnuml/linux-um_2.6.18.1-bb2-xt-4m.orig.tar.gz
   > mv linux-um_2.6.18.1-bb2-xt-4m.orig.tar.gz /usr/share/vnuml/kernels/
   > cd /usr/share/vnuml/kernels/
   > tar -xzf linux-um_2.6.18.1-bb2-xt-4m.orig.tar.gz
   > rm linux-um_2.6.18.1-bb2-xt-4m.orig.tar.gz

6. Instalamos el paquete screen que permite gestionar ventanas.
   > apt-get install screen

B.1.2. CREACIÓN DE IMÁGENES PARA LAS MÁQUINAS VIRTUALES: ROUTERS

Siendo súper-usuario (root), se deben seguir los siguientes pasos:

1. Creamos el filesystem donde se va a almacenar la imagen y lo montamos
   > dd if=/dev/zero of=/usr/share/vnuml/filesystems/debian.ext3 bs=1024k seek=600 count=0
   > mkfs.ext3 /usr/share/vnuml/filesystems/debian.ext3
   > tune2fs -c 0 -i 0 /usr/share/vnuml/filesystems/debian.ext3
   > mkdir /mnt/debian
   > mount -o loop /usr/share/vnuml/filesystems/debian.ext3 /mnt/debian

2. Obtenemos una imagen simplificada de debian.
   > debootstrap etch /mnt/debian/ ftp://ftp.debian.org/debian/
   > mount -t proc none /mnt/debian/proc
   > chroot /mnt/debian
   > env -update

3. Editamos el fichero de fuentes para instalar los paquetes auxiliares
   > vi /etc/apt/sources.list

4. Añadimos al final la siguiente línea
   > deb http://ftp.debian.org/debian etch main contrib non-free

5. Actualizamos e instalamos los paquetes auxiliares.
   > apt-get update
   > apt-get install bzip2 dnsutils fdutils file fileutils iproute iputils-tracepath ipv6calc less modutils ncurses-term openssh-client openssh-server openssl perl perl-modules pidentd sharutils shellutils ssh ssl-cert syslinux tcsh telnet textutils time ucf vlan sudo vim

6. Instalamos el software de quagga, que nos proporciona los protocolos de routing.
   > cd /tmp/ > wget http://www.it.uc3m.es/cjbc/quagga_0.99.5-5etch3_i386.deb
   > dpkg -i quagga_0.99.5-5etch3_i386.deb
   > wget http://www.it.uc3m.es/cjbc/quagga.tar.gz
   > tar xzvf quagga.tar.gz
   > cp -p quagga/* /etc/quagga/
   > (edit /etc/quagga/daemons)
   > sudo /etc/init.d/quagga start

7. Desinstalamos los paquetes que no se van a necesitar > apt-get clean

8. Modificamos fecha local.
   > ln -sf /usr/share/zoneinfo/Europe/Madrid /etc/localtime

9. Modificamos el /etc/hosts, debe quedar del siguiente modo
   127.0.0.1 localhost
   ::1 localhost

   # The following lines are desirable for IPv6 capable hosts
   ::1 ip6-localhost ip6-loopback
   fe00::0 ip6-localnet
   ff00::0 ip6-mcastprefix
   ff02::1 ip6-allnodes
   ff02::2 ip6-allrouters
   ff02::3 ip6-allhosts

10. Editamos el fichero */etc/inittab*. Debe tener la siguiente línea.
    0:2345:respawn:/sbin/getty 38400 tty0

11. Editamos el fichero */etc/network/interfaces*. El fichero debe ser así:
    # Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
    # /usr/share/doc/ifupdown/examples for more information.
    iface lo inet loopback
    auto lo

    iface eth0.0 inet static
    pre-up ifrename -i eth1 -n eth0.0 address 192.168.0.1
    netmask 255.255.255.0
    auto eth0.0

    iface eth0.1 inet static
    pre-up ifrename -i eth2 -n eth0.1
    address 192.168.1.1
    netmask 255.255.255.0
    auto eth0.1

    iface eth0.2 inet static
    pre-up ifrename -i eth3 -n eth0.2
    address 192.168.2.1
    netmask 255.255.255.0
    auto eth0.2

    iface eth0.3 inet static
    pre-up ifrename -i eth4 -n eth0.3
    address 192.168.3.1
    netmask 255.255.255.0
    auto eth0.3

    iface eth0.4 inet static
    pre-up ifrename -i eth5 -n eth0.4
    address 192.168.4.1

netmask 255.255.255.0
auto eth0.4

iface eth1 inet static
pre-up ifrename -i eth6 -n eth1
address 192.168.5.1
netmask 255.255.255.0
auto eth1

12. Para quitar el login al arrancar la máquina virtual instalamos el siguiente paquete:
    > apt-get install console-tools

13. Vemos la ocupación real de la imagen y reducimos el tamaño del filesystem para que ocupe el mínimo espacio posible (Imagen para nuestros routers —-> 260M ocupados 211M).
    > dd if=/dev/zero of=/usr/share/vnuml/filesystems/imagenRouter.ext3 bs=1024k seek=260 count=0
    > mkfs.ext3 /usr/share/vnuml/filesystems/imagenRouter.ext3
    > tune2fs -c 0 -i 0 /usr/share/vnuml/filesystems/imagenRouter.ext3
    > mount -o loop /usr/share/vnuml/filesystems/imagenRouter.ext3 /mnt/debian
    > mkdir /mnt/otrofs
    > mount -o loop /usr/share/vnuml/filesystems/debian.ext3 /mnt/otrofs/
    > cp -a /mnt/otrofs/* /mnt/debian/
    > umount /mnt/otrofs
    > umount /mnt/debian

B.1.3. CREACIÓN DE IMÁGENES PARA LAS MÁQUINAS VIRTUALES: HOST

Para crear la imagen de los host vamos a partir de la imagen que tenemos ya creada. Para ello realizamos los siguientes pasos con el usuario root:

1. Creamos un filesystem para la nueva máquina virtual.
   > dd if=/dev/zero of=/usr/share/vnuml/filesystems/imagenHost.ext3 bs=1024k seek=300 count=0
   > mkfs.ext3 /usr/share/vnuml/filesystems/imagenHost.ext3
   > tune2fs -c 0 -i 0 /usr/share/vnuml/filesystems/imagenHost.ext3

2. Hacemos una copia de la imagen de los routers en el filesystem para los host
   > mkdir /mnt/debian
   > mount -o loop /usr/share/vnuml/filesystems/imagenHost.ext3 /mnt/debian
   > mount -o loop /usr/share/vnuml/filesystems/debian.ext3 /mnt/otrofs/
   > cp -a /mnt/otrofs/* /mnt/debian/
   > umount /mnt/otrofs
   > umount /mnt/debian
   > mount -t proc none /mnt/debian/proc
   > chroot /mnt/debian

3. Desinstalamos los paquetes que no necesitamos.
   > apt-get remove quagga, ifrename

4. Instalamos paquetes propios de los host.
   > apt-get install bzip2 dnsutils fdutils file fileutils iperf iproute iputils-tracepath ipv6calc less modutils ncurses-term openssh-client openssh-server openssl perl perl-modules pidentd sharutils shellutils ssh ssl-cert syslinux tcpdump tcsh telnet textutils time tshark ucf vlan sudo vim wireless-tools madwifi-tools

B.1.4. CÓMO MODIFICAR UNA IMAGEN EXISTENTE

Una vez creada la imagen para la máquina virtual, podemos modificarla montándola en un filesystem del sistema, haciendo los cambios que necesitemos y desmontándola después. El proceso sería del siguiente modo:

1. Montamos la imagen en un filesystem.
   > mount -o loop /usr/share/vnuml/filesystems/[nombreImagen].ext3 /mnt/debian
   > mount -t proc none /mnt/debian/proc
   > chroot /mnt/debian

2. Realizamos los cambios oportunos (instalación de software nuevo, modificación de ficheros, etc...)

3. Desmontamos la imagen
   > exit
   > umount /mnt/debian/proc/
   > fuser -k /mnt/debian
   > umount /mnt/debian

B.2. ENTORNO AUTOCORRECCIÓN

B.2.1. INSTALACIÓN JAVA

Siendo súper-usuario (root), se deben seguir los siguientes pasos:

1. Descargar el binario de la siguiente dirección:
   https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jdk-6u11-oth-JPR@CDS-CDS_Developer

2. Abrir una consola o terminal y crear el directorio /usr/java:
   > mkdir /usr/java

3. Asignar permisos de forma que lo puedan ejecutar varios usuarios, y no solo el superusuario del sistema:
   > chown -R usuario /usr/java

4. Copiar el binario en el directorio creado y asignar permisos de ejecución:
   > cp /home/usuario/jdk-6u11-linux-i586.bin /usr/java
   > chmod a+x /usr/java/jdk-6u11-linux-i586.bin

5. Ejecutar el instalador:
   > ./jdk-6u11-linux-i586.bin

6. Aceptar la licencia.

7. Asignar los mismos permisos que en el punto 3 para la carpeta creada durante la instalación:
   >chown -R usuario /usr/java/jdk1.6.0_11

8. Configuramos las variables de entorno. Para ello, editamos el fichero */root/.bashrc* y el */home/usuario/.bashrc*, y añadimos las siguientes líneas:
   export JAVA_HOME=/usr/java/jdk1.6.0_05
   export PATH=$JAVA\_HOME/bin$:PATH

9. Cerrar la consola y abrir una nueva.

B.2.2. INSTALACIÓN MODULO DE SEGURIDAD

Como súper usuario (root), hemos seguido los siguiente pasos:

1. Instalamos el paquete GnuPG, con el que vamos a cifrar y firmar las prácticas:
   > apt-get install gnupg

2. Creamos el usuario de la aplicación con el cuál vamos a firmar las prácticas. El nombre del usuario es uvil.

3. Generamos el par de claves (pública y privada) que se utilizarán para firmar y cifrar los documentos.
   > su - uvil
   > gpg –gen-key

B.3. INSTALACIÓN DE PARA LA HERRAMIENTA WEB

Para instalar apache2 y php5 se deben seguir los siguientes pasos:

> apt-get install apache2.2-common apache2
> apt-cache search php5 libapache2-mod-php5
> apt-get install php5-gd
> apt-get install phpmyadmin

Apéndice C

MANUAL DE USUARIO

C.1. ENTORNO VIRTUAL: LANZAR UN ESCENARIO DE RED

Para lanzar el entorno virtual hay que seguir los siguientes pasos:

1. Configuramos y habilitamos el interfaz del PC para crear la red de mantenimiento.
   > sudo sh /usr/dist/bin/labVirtualUC3M/configUp.sh [ipRedMantenimiento] [máscaraRed]

   - **[ipRedMantenimiento]**: es la IP que se asigna al PC anfitrión para comunicarse con los nodos virtuales. Este dato se proporciona en el enunciado del ejercicio.

   - **[máscaraRed]**: Debe ir en formato XXX.XXX.XXX.XXX

2. Ejecutamos el script que arranca la simulación.
   > sh /usr/dist/bin/uc3m_lab_virtual arrancar [escenario]

   - **arrancar**: La opción arrancar levanta el escenario que se le indique.

   - **[escenario]**: Debe indicarse sobre qué escenario se desea realizar la acción de arrancar. Hay que indicar el path completo donde esté alojado el escenario.

C.2. ENTORNO VIRTUAL: PARAR UN ESCENARIO DE RED

Para parar el entorno virtual hay que seguir los siguientes pasos:

1. Ejecutamos el script que para la simulación.
   > sh /usr/dist/bin/uc3m_lab_virtual parar|purgar [escenario]

   - **parar**|**purgar**: son las posibles acciones que se pueden realizar sobre un escenario para pararlo. La opción parar destruye la simulación que se indique (previamente ha de estar ejecutándose). La opción purgar destruye la simulación indicada y elimina los ficheros temporales que queden en el equipo referentes a dicha simulación.

- **[escenario]**: Debe indicarse sobre qué escenario se desea realizar la acción de parada. Hay que indicar el path completo donde esté alojado el escenario.

2. Deshabilitamos el interfaz del PC para crear la red de mantenimiento.
   > sudo sh /usr/dist/bin/labVirtualUC3M/configDown.sh

C.3. ENTORNO VIRTUAL: LANZAR AUTOCORRECCIÓN

Para lanzar el entorno de autocorrección simplemente hay que ejecutar el siguiente comando:

> sh /usr/dist/bin/labVirtualUC3M/corregirLabUC3M.sh

El programa irá pidiendo datos al alumno para que éste los rellene y se proceda a generar el fichero de corrección

Apéndice D

PRESUPUESTO

PRESUPUESTO DE PROYECTO

Laboratorio virtual UC3M: Desarrollo de un entorno no presencial para la realización de prácticas de configuración de red y su autocorrección

1.- Autor:

María Celeste Durán González

2.- Departamento:

Ingeniería Telemática

3.- Descripción del Proyecto:

- Titulo: Laboratorio virtual UC3M: Desarrollo de un entorno no presencial para la realización de prácticas de configuración de red y su autocorrección
- Duración (meses): 18
- Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):

0 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)
Duran Gonzalez, Mª Celeste		Ingeniero Técnico	12	2.694,39	32.332,68
		Hombres mes 12		**Total**	32.332,68

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Equipo desarrollo	1.000,00	100	12	60	200,00
				Total	200,00

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} \text{ x } C \text{ x } D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	32.333
Amortización	200
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	6.507
Total	39.039