



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

Simulación de redes con GNS3
Documentación Técnica



Presentado por Javier García González
en Universidad de Burgos -- 21 de enero de 2019
Tutores: Alejandro Merino Gómez y Daniel Sarabia Ortiz

Índice general

Apéndice A	6
A.1. Introducción	6
A.2. Planificación temporal	6
Sprint 0 (28/09/2018 – 11/10/2018)	8
Sprint 1 (11/10/2018 – 26/10/2018)	9
Sprint 2 (26/10/2018 – 03/12/2018)	10
Sprint 3 (03/12/2018 – 18/12/2018)	11
Sprint 4 (18/12/2018 – 08/01/2019)	11
Sprint 5 (08/01/2019 – 14/02/2019)	12
Resumen final de planificación	13
A.3. Estudio de viabilidad	14
Viabilidad Económica	14
Viabilidad Legal	15
Apéndice B	17
B.1. Introducción	17
B.2. Diseño de la red	17
B.3. Catálogo de requisitos del proyecto	18
RF-1 Importar Dockers a GNS3	18
RF-2 Conectar GNS3 a Internet	18
RF-3 Utilizar GNS3 como Cliente-Servidor	19
RF-4 Implementar Servidor Web	20
RF-5 Implementar Servidor DHCP:	20
RF-6 Implementar Red VLAN:	21
RF-7 Configurar enrutamiento de la red:	21
RF-8 Programa de configuración remota:	22
C.1. Introducción	24
C.2. Requisitos de usuarios	24
C.3. Instalación	24

VirtualBox (versión instalada 5.2.18)	25
VMWare Workstation 12 player	25
GNS3 2.1.0	26
C.4. Manual del usuario	28
Conectar GNS3 a redes externas (Internet)	28
Conectar GNS3 contra un servidor remoto	30
Configuración de redes virtuales VLAN.	32
Servidor Web con apache sobre Ubuntu.	34
Servidor DHCP.	37
Servidor DHCP. Configurar retransmisor	41
Enrutamiento dinámico	42
Configuración remota	43

Índice de figuras

Ilustración 1: Proyecto en GitHub	7
Ilustración 2: Tiempo empleado en el sprint 0.	9
Ilustración 3: Tiempo empleado en el sprint 1.	9
Ilustración 4: Tiempo empleado en el sprint 2.	10
Ilustración 5: Tiempo empleado en el sprint 3.	11
Ilustración 6: Tiempo empleado en el sprint 4.	12
Ilustración 7: Tiempo empleado en el sprint 5.	13
Ilustración 8: Comparación final de tiempos.	13
Ilustración 9: Licencia GNS3	16
Ilustración 10: Topología de la red final.	17
Ilustración 11: Oracle VM VirtualBox	25
Ilustración 12: VMWare	26
Ilustración 13: GNS3 VM	27
Ilustración 14: Entorno de GNS3	27
Ilustración 15: Compartir Internet en adaptador de red.	28
Ilustración 16: Conexión a Internet	29
Ilustración 17: IP equipo Mesa	30
Ilustración 18: IP equipo Portátil	30
Ilustración 19: GNS3 configuración servidor remoto	31
Ilustración 20: Opciones ejecución GNS3	31
Ilustración 21: Redes VLAN	32
Ilustración 22: VLAN	33
Ilustración 23: Prueba VLAN	33
Ilustración 24: Redes VLAN con administración	34
Ilustración 25: Apache configuración	35
Ilustración 26: Apache configuración 2	35
Ilustración 27: Apache configuración 3	36
Ilustración 28: Prueba Apache	37
Ilustración 29: Fichero configuración DHCP	38
Ilustración 30: Fichero configuración DHCP 2	38
Ilustración 31: Fichero configuración DHCP3	39
Ilustración 32: Topología DHCP	39
Ilustración 33: Fichero configuración DHCP 4	39
Ilustración 34: Retransmisor DHCP	41
Ilustración 35: Leer fichero desde Python	43
Ilustración 36: Función telnet.	44
Ilustración 37: Función menú.	45

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se desglosa el estudio previo al proyecto, en el cuál se calculan los tiempos que pueden llevarnos cada uno de los objetivos. Gracias a este estudio se van a conocer los recursos necesarios para poder llevar a cabo el proyecto.

Se elabora una planificación temporal en la cual se marca cada parte del proyecto con un tiempo estimado de 1-2 semanas.

Se realiza a su vez un estudio de viabilidad del proyecto, dividido en viabilidad económica y viabilidad legal:

- Económica: Se van a calcular los costes del proyecto, así como los beneficios posibles que nos pueda dar el proyecto una vez concluido.
- Legal: Para este proyecto lo más importante es la licencia de cada uno de los elementos empleados en el proyecto.

A.2. Planificación temporal

Para el desarrollo de todas las partes del proyecto se ha utilizado la herramienta GitHub, la cual nos ha permitido crear un proyecto y ver en cada momento que iteraciones tenemos pendientes, en proceso o ya realizadas, de esta forma se ha podido gestionar mediante una metodología ágil de tipo SCRUM.

- En un principio el tiempo que se ha estimado para el proyecto ha sido de aproximadamente dos horas diarias debido a que se ha tenido que compaginar con el trabajo a jornada partida, dedicando tiempo a partir de las 19:00h cada día. De esta forma se ha destinado al proyecto 2 horas diarias los días laborables y

para los fines de semana, dependiendo de las fechas, se ha destinado al proyecto un tiempo de 4-5 horas diarias.

- Se han generado iteraciones para cada periodo de tiempo, tras lo cuales se han establecido reuniones de control.
- El tiempo de cada iteración se ha establecido entre una y dos semanas, en función de la complejidad de la iteración.
- Las tareas que realizar han sido controladas mediante la aplicación GitHub en su modulo de proyectos, donde se ha ido guardando cada tarea en su correspondiente campo:

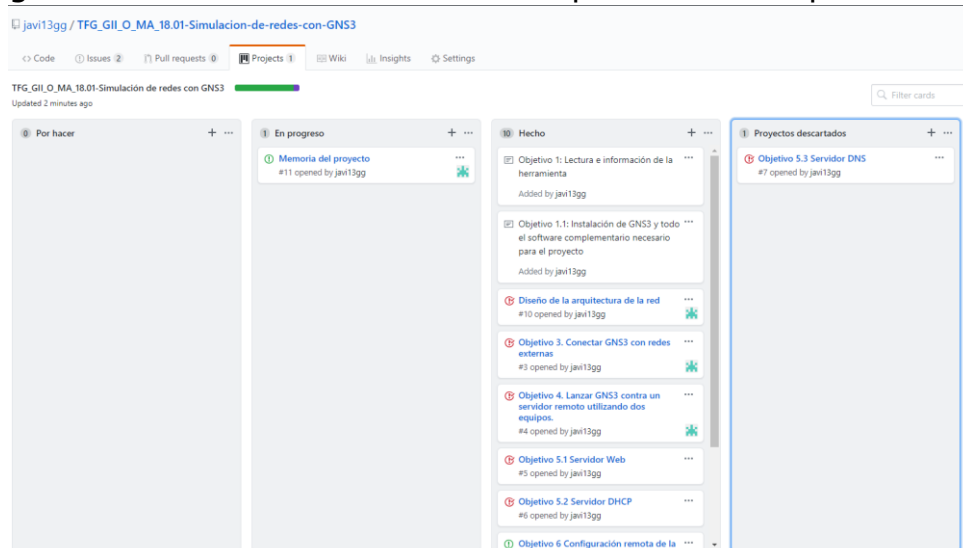


Ilustración 1: Proyecto en GitHub

Teniendo en cuenta el tiempo disponible se ha adaptado el proyecto modificando sobre la marcha alguno de los objetivos, centrando más el desarrollo en algunos puntos considerados más interesantes y dejando de lado aquellos que bien por no tener tiempo suficiente para realizarlos íntegramente o haber sido ya estudiados en las asignaturas cursadas a lo largo del grado.

La estimación de tiempo inicial se basó en los Issues generados en GitHub, y se calculó el proceso completo:

Issues	Estimación Temporal
Issue 1: Información	40 horas
Issue 2: Diseño red y practica	30 horas
Issue 3: Dockers	16 horas
Issue 4: Conexión Internet	8 horas
Issue 5: GNS3 cliente - servidor	16 horas
Issue 6: Servidor Web	10 horas
Issue 7: Servidor DHCP	10 horas
Issue 8: Servidor DNS	10 horas
Issue 9: Red VLAN	10 horas
Issue 10: Configuración remota	50 horas
Issue 11: Memoria	60 horas
Issue 12: Presentación	40 horas
Tiempo total	300 horas

Tabla 1: Tabla de tiempos

Se desglosa el trabajo realizado en cada sprint donde vemos los tiempos reales empleados para cada iteración:

Sprint 0 (28/09/2018 – 11/10/2018)

En este primer sprint, en el cual se dio comienzo al proyecto, se estableció la primera reunión con los tutores, donde se establecieron los puntos a estudiar, se explico el proyecto, y se tuvieron en cuenta los tiempos disponibles para planificar cada paso. Debido al tiempo disponible se decidió realizar la planificación para presentar el trabajo en segunda convocatoria, el día 14 de febrero.

Tras este primer paso, se establecieron los primeros objetivos, que fueron familiarizarse con la aplicación, buscar información y empezar a instalar todos los componentes. Para este primer paso se había pensado en unas 40 horas. Pero finalmente fueron más, debido a problemas con las aplicaciones y a varias reinstalaciones hasta dejar todo operativo. Se invirtieron finalmente 45 horas entre las instalaciones y la búsqueda de información.

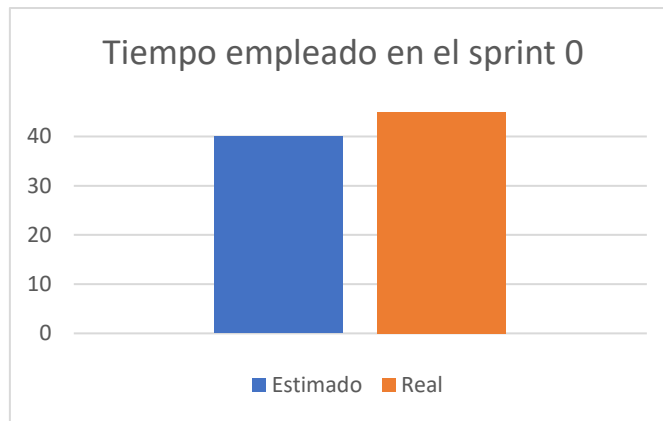


Ilustración 2: Tiempo empleado en el sprint 0.

Sprint 1 (11/10/2018 – 26/10/2018)

En este sprint se finalizó la documentación y estudio sobre el programa y se comenzó con el siguiente paso, que consistió en crear la topología de la red y realizar pruebas, como fueron las prácticas de clase para poder aprender el manejo y hacer las primeras pruebas. Se realizó la segunda reunión donde se establecieron los nuevos objetivos, aparte de finalizar los de las semanas anteriores.

El tiempo que se estimó en un principio fue de unas 30 horas, finalmente se realizó en un periodo de 35 aproximadamente, sin contar el tiempo empleado en este sprint para poder finalizar el anterior, que esta incluido en el sprint 0. De ahí la duración de más de esos 15 días.

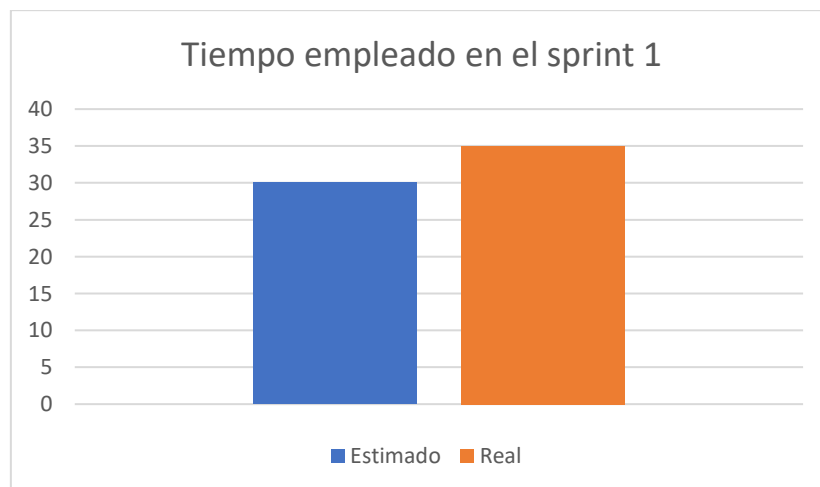


Ilustración 3: Tiempo empleado en el sprint 1.

Sprint 2 (26/10/2018 – 03/12/2018)

Para este periodo de tiempo se establecieron varios objetivos, concretamente los Issues del 3 al 5.

Para cada uno de ellos se calculó un tiempo, para el primero de ellos, el uso de Dockers, fueron 16 horas, para la conexión a Internet de GNS3 fueron 8 horas y para la conexión de GNS3 cliente – servidor se estableció un tiempo de 16 horas.

El Issue 3 llevó algo más de tiempo ya que nos obligó a modificar los objetivos del proyecto. Nos encontramos aquí con uno de los problemas, los dockers no podían ser exportados de forma que el proyecto no podía ser ejecutado en otros equipos. Esto nos hizo descartar dicha idea, aun que si que nos permitió hacer un estudio comparativo del uso de dockers frente a máquinas virtuales y VPCS.

El Issue 4 fue de los más sencillos y se pudo realizar en un tiempo menor al establecido en un principio, bajando de las 8 horas estimadas a 6. Teniendo en cuenta que lo más complicado fue obtener la información necesaria.

En cuanto al Issue 5, se realizó un estudio comparativo donde se comprobó que si utilizamos un equipo servidor que realice la carga de recursos, y utilizamos un equipo cliente donde lanzar la aplicación la carga es menor y podemos emplear menos recursos para la simulación.

Teniendo en cuenta el computo de tiempo general, en este sprint se cumplió con lo calculado inicialmente.

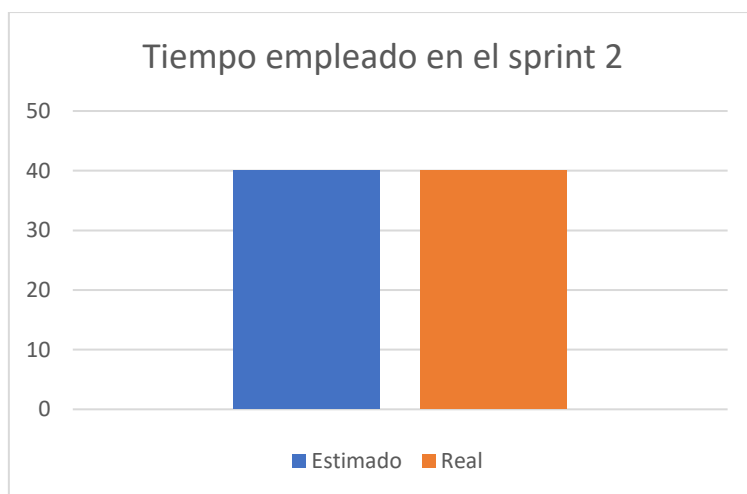


Ilustración 4: Tiempo empleado en el sprint 2.

Sprint 3 (03/12/2018 – 18/12/2018)

Al comienzo de este sprint, se fijó en la reunión el objetivo de crear los servidores que se iban a implementar: Servidor Web, Servidor DHCP y Servidor DNS, la implementación de los servidores DHCP y Web se realizó sin problemas, pero debido a que no dio tiempo a lo largo de este periodo se decidió dejar el DNS sin hacer para poder centrar el proyecto en otros puntos más importantes, como crear una red VLAN y poder hacer la configuración de forma remota con Python.

El tiempo original pensado fue de 10 horas para cada servidor, alargándose en los dos primeros y dejando sin tiempo la opción del Servidor DNS. Quedando al final el tiempo del Issue 6 en 15 horas y del Issue 7 en 12 horas.

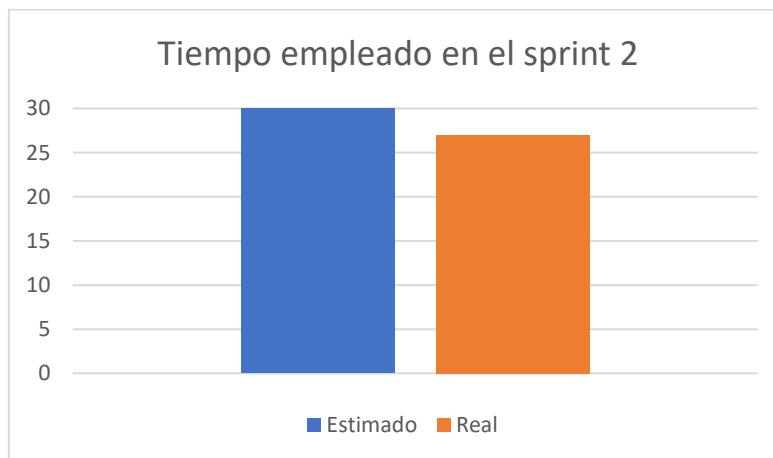


Ilustración 5: Tiempo empleado en el sprint 3.

Sprint 4 (18/12/2018 – 08/01/2019)

En la reunión de comienzo de este sprint, se tuvo en cuenta el periodo festivo de las navidades, en el cual entre semana eran días laborables y los fines de semana festivos, por lo que en este mes fue en el que menos se pudo avanzar en el proyecto. Se fijó como objetivo realizar la red VLAN y empezar a unir todo el proyecto para realizar las pruebas de enrutamiento de forma que se pudiese avanzar al ultimo punto de implementación, la configuración remota.

Se dieron pocos problemas con la red VLAN, lo que corresponde con el Issue 9, y en este paso se añadió el objetivo del enrutamiento, lo cual no se tuvo en cuenta en un inicio.

En ese punto si que hemos tenido varios problemas, debido en gran parte a la red VLAN y su enrutamiento, llevando este sprint a ocupar el mes completo y los primeros días del sprint siguiente.

Por lo que finalmente las 10 horas iniciales se convirtieron en 25 horas.

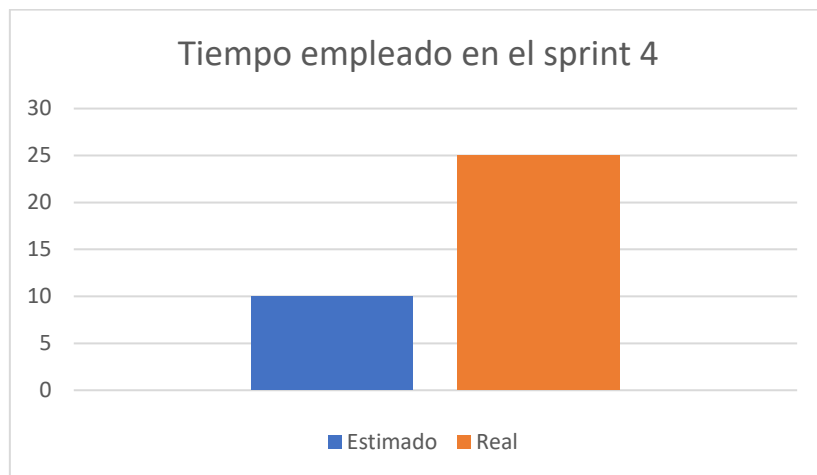


Ilustración 6: Tiempo empleado en el sprint 4.

Sprint 5 (08/01/2019 – 14/02/2019)

Este sprint corresponde al paso de la configuración remota y la preparación de la entrega final, en la reunión se establecieron los requisitos que debía de cumplir el programa a desarrollar en Python.

Decidimos que el programa se iba a encargar de modificar los parámetros de red de Routers y equipos terminales con Ubuntu. Modificando la IP, máscara de red y puerta de enlace en los casos oportunos. Para ellos recibirá los parámetros mediante un documento de texto y lanzará las configuraciones. Como se va a desarrollar en equipos con Ubuntu, el programa mostrará la interfaz mediante terminal mostrando las opciones.

El tiempo que ha llevado este Issue, contando el estudio previo y las pruebas finales ha llegado a las 40 horas. Quedando en 10 horas menos de las previstas inicialmente.

En cuanto a la memoria del proyecto, pese a que se ha ido cumplimentando la información a lo largo de la implementación del proyecto, ha llevado un total de más de 60 horas, entre redacción, formato, correcciones etc. Quedando finalmente en 64 horas

También se ha empleado el tiempo en la realización de las presentaciones y videos, los cuales han llevado su tiempo para poder adaptar todo el contenido del proyecto en los limites de tiempo

establecidos. Se estimó una duración de 40 horas que se quedaron en algo menos de 36 horas, limitados en gran parte por el tiempo restante hasta la fecha de entrega.

Durante este sprint las reuniones se han sustituido por conversaciones mediante email para comprobar los avances con la memoria.

Los últimos días se han dedicado a la confección y grabación de los videos de presentación y demostración del proyecto.

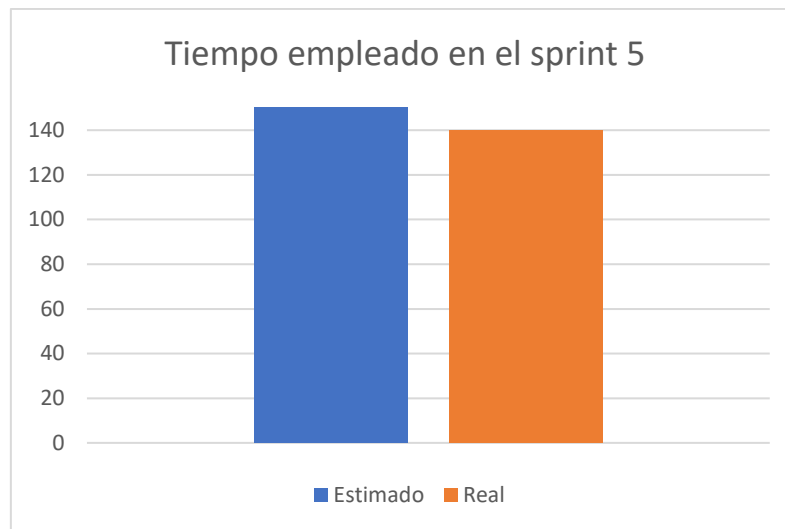


Ilustración 7: Tiempo empleado en el sprint 5.

Resumen final de planificación

Horas estimadas: 300 horas

Horas reales invertidas: 312 horas



Ilustración 8: Comparación final de tiempos.

A.3. Estudio de viabilidad

Viabilidad Económica

- **Costes**

Al tratarse de un proyecto educativo, los costes no han sido reales, pero se realiza una estimación de todos los posibles conceptos económicos que podría implicar el proyecto real:

1. **Coste de salario:**

Se tienen en cuenta 4 meses de trabajo, incluyendo todos los costes de seguridad social:

Salario neto:	1029,05€/mes
Retención IRPF (5%):	63,5€/mes
Seguridad Social (30%):	324,15€/mes
Coste total:	5666,80€

2. **Coste de Hardware:**

Se ha necesitado emplear dos equipos informáticos, para poder comprobar el rendimiento en diferentes procesadores se ha empleado un equipo con procesador Intel y otro con procesador AMD. El coste de cada equipo, teniendo en cuenta una vida útil del equipo de 6 años:

Coste equipo Intel (720€):	40€
Coste equipo AMD (560€):	31,1€
Coste total:	71,1€

3. **Coste de Software:**

Para este proyecto se ha necesitado emplear dos licencias de Windows 10 Pro, dichas licencias van a ser utilizadas a lo largo de los 6 años que se da uso a los equipos:

Coste de cada licencia (245€):	13,6€
Coste total:	26,2€

4. Otros costes:

Alquiler de oficina:	200€/mes
Conexión a Internet:	35€/mes
Total:	940€

Coste total del proyecto: 6704,10€

- **Beneficios**

El proyecto no genera beneficios por sí mismo, ya que no es algo distribuible o que se pueda vender, es más un proyecto de simulación a realizar por petición de una empresa, en la que nos den los parámetros y elementos que quieran añadir a su red. Una vez encargado el proyecto se entregarán los resultados finales. Por lo que sería necesario calcular el precio en función de los beneficios a obtener. Si queremos obtener únicamente de beneficio nuestro sueldo, solo con el coste del proyecto total es suficiente, podríamos pedir al cliente los **6704,10€** de gastos que han surgido, en cambio si fuese para una empresa que tenga beneficios más allá del salario del trabajador, se podría añadir al presupuesto la cantidad que se quiera obtener por la realización. Incluso se podría añadir un presupuesto extra para poner en marcha la red real de la institución, por ejemplo, para obtener un beneficio de 1000€ por el proyecto, el presupuesto final será de: **7704,10€**.

Viabilidad Legal

En cuanto a la legalidad del proyecto, únicamente nos encontramos con las licencias de los productos empleados, y la licencia de uso del proyecto final.

Las únicas licencias de pago empleadas en el proyecto han sido las licencias de Windows 10 Pro, las cuales nos permiten su uso sin restricciones. Para el programa GNS3 se ha utilizado su licencia que

puede obtenerse de forma gratuita desde su página Web con licencia Open Source con versión GPL v3. Se ha utilizado el programa Spyder (gratuito) para desarrollar la aplicación en el lenguaje Python, lo cual no nos limita el uso de la aplicación, ni nos interfiere a la hora de darle licencia. Las licencias utilizadas para las máquinas virtuales son licencias de Ubuntu, gratuitas y sin restricciones de uso, por lo que podemos escoger que licencia va a tener nuestro proyecto.

La licencia que más se ajusta al proyecto es la licencia GNU 3.0, la cual nos permite liberar el código fuente, distribución gratuita sin necesidad de ofrecer ningún tipo de garantía, libertad para realizar cualquier modificación en el proyecto, la posibilidad de uso privado y de patentes. Como el proyecto va a ser puramente educativo, no se va a registrar la licencia y se hace el estudio teórico únicamente.

En cuanto a las imágenes utilizadas en la memoria, se han empleado iconos de aplicaciones e imágenes de uso público sin restricciones. Las capturas de pantalla mostradas son realizadas por el autor de este proyecto en el momento de la realización de este.

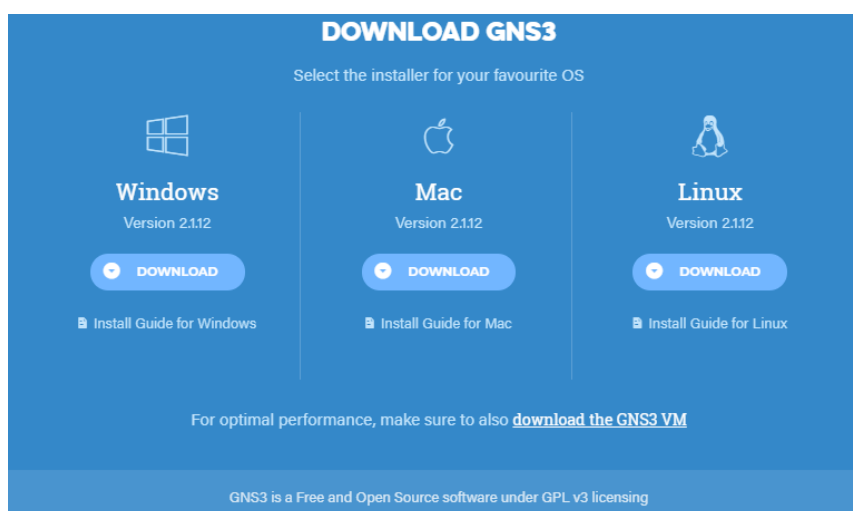


Ilustración 9: Licencia GNS3

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado tratamos el tema del diseño de la red a construir, previamente a comenzar a unir todas partes, se han generado todas ellas por separado, de esta forma hemos podido hacer pruebas más simples sobre ellas y también hemos podido hacer un diseño y estudio de la topología de la red final.

B.2. Diseño de la red

Una vez decididos todos los componentes que van a formar parte de la red, se realiza la topología que seguirá de forma que todo quede conectado y pueda funcionar de forma óptima:

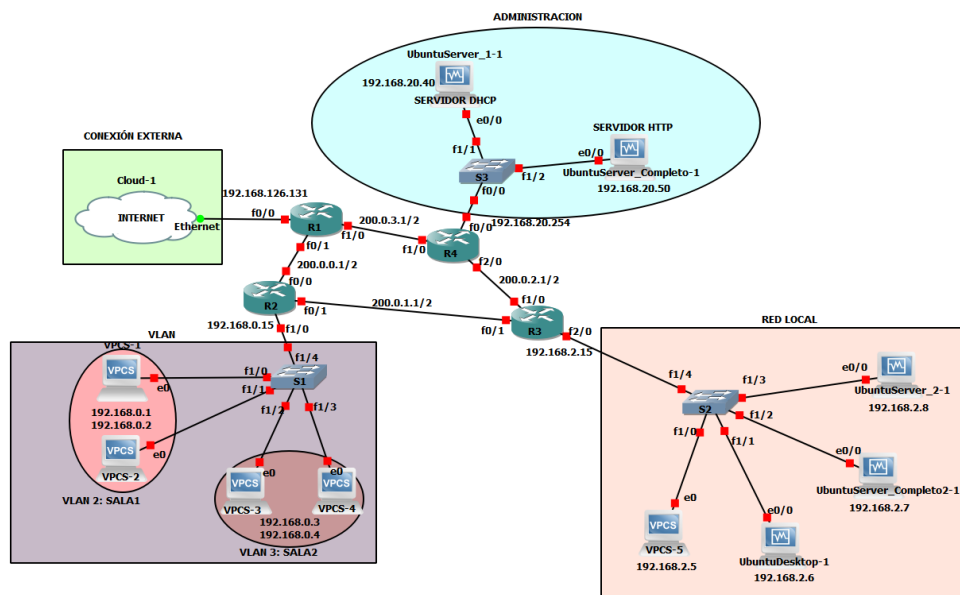


Ilustración 10: Topología de la red final.

B.3. Catálogo de requisitos del proyecto

RF-1 Importar Dockers a GNS3

Se añaden Dockers a la aplicación GNS3, de forma que se pueda comparar el rendimiento entre el uso de estos y de máquinas virtuales, ya que el uso en el proyecto se ha descartado por temas de exportación, pero si podemos hacer un estudio comparativo.

- **RF-1.1 Importar Docker Ubuntu**

Se añade al programa un Docker con las funciones básicas de red que tenemos en Ubuntu.

- **RF-1.2 Importar Docker Python**

Se importa un Docker con base en Ubuntu, que nos permite ejecutar programas escritos en el lenguaje Python sin necesidad de un sistema operativo completo.

RF-2 Conectar GNS3 a Internet

Este caso no permite obtener conexión a Internet utilizando el elemento cloud de GNS3, que se conectará a la red en la que está el equipo anfitrión.

- **RF-2.1 Importar elementos**

Para poder conectarnos a internet vamos a necesitar introducir en la aplicación GNS3 los elemento cloud y un router que se conecte a este.

- **RF-2.2 Configurar equipo anfitrión**

Para poder dar conexión a GNS3 es necesario habilitar parámetros en el equipo local.

- **RF-2.3 Configurar router**

El ultimo paso será configurar el router de forma que obtenga conexión a Internet mediante el elemento cloud.

RF-3 Utilizar GNS3 como Cliente-Servidor

Para este proyecto vamos a crear la posibilidad de ejecutar uno de los equipos contra el otro, de forma que no actúe de servidor y sea el que más carga de trabajo reciba y otro sea un cliente que solo tenga que ejecutar en entorno de GNS3

- **RF-3.1 Configuración de los entornos**

Para poder ejecutar GNS3 en modo cliente sobre un equipo servidor, en ambos entornos debemos instalar un complemento que nos permita dicha conexión para posteriormente ejecutarlo. También debemos tener los dos equipos en red y con conexión entre ellos.

- **RF-3.1 Configuración de los entornos**

Una vez tenemos instalado el complemento, podemos ejecutar GNS3 en el servidor en este modo, y esto nos permitirá que una vez ejecutemos en el cliente la aplicación nos podamos conectar al otro equipo.

RF-4 Implementar Servidor Web

En este requisito vamos a implementar un servidor web que nos permita ver páginas creadas en este desde el resto de los equipos de la red.

- **RF-4.1 Importar elementos**

Para este caso, vamos a emplear una máquina virtual con Ubuntu Server, además de los equipos que vayan a conectarse a este y su correspondiente router.

- **RF-4.2 Configurar elementos**

El siguiente paso es realizar la configuración del servidor para que se pueda acceder desde el resto de los equipos y esté operativo.

RF-5 Implementar Servidor DHCP:

En este requisito vamos a implementar un servidor web que nos permita dar direcciones DHCP al resto de equipos.

- **RF-5.1 Importar elementos**

Para este caso, vamos a emplear una máquina virtual con Ubuntu Server, además de los equipos que vayan a conectarse a este y su correspondiente router.

- **RF-5.2 Configurar elementos**

El siguiente paso es realizar la configuración del servidor para que se pueda acceder desde el resto de los equipos y esté operativo.

RF-6 Implementar Red VLAN:

Un requisito del proyecto será implementar una red VLAN interna que tenga su propia red privada virtual.

- **RF-6.1 Importar elementos**

Necesitamos crear los elementos que van a formar parte de la red, en este caso vamos a emplear VCPS y un switch, este switch tendrá acceso al resto de la red mediante un router.

- **RF-6.2 Configurar elementos**

El siguiente paso es realizar la configuración de todos los elementos para que se pueda conectar la red entre ellos.

RF-7 Configurar enrutamiento de la red:

Este es uno de los pasos más importantes, ya que va a ser el que permita que se puedan comunicar todos los elementos de la red. Para ellos se tendrá que hacer los siguiente

- **RF-7.1 Conectar elementos**

El primer paso es conectar todos los elementos de forma que estén unidos físicamente. Esto lo haremos conectando cada elemento a su router y los routers entre ellos.

- **RF-7.2 Configurar elementos**

En este paso se configuran los routers para utilizar el enrutamiento RIP y en los equipos terminales se configuran las direcciones IP y las puertas de enlace.

RF-8 Programa de configuración remota:

Debido a que este es un programa de simulación de redes, su contenido de código es muy bajo. El programa deberá tener un menú, en el cual nos va a mostrar 3 opciones:

- **RF-8.1 Mostrar el fichero de configuración**

Al seleccionar esta opción, el programa lee el fichero que incluye todos los parámetros de configuración de la red y lo muestra por pantalla, este fichero debe estar en la misma carpeta que el script del programa.

- **RF-8.2 Lanzar el fichero de configuración**

Si entramos en la opción de lanzar la configuración, el programa lee el fichero de configuración y analiza línea por línea el elemento a configurar, teniendo en cuenta en cada caso que configuraciones son necesarias en función del elemento.

- **RF-8.3 Salir del programa**

Al seleccionar esta opción finalizará el programa.

Apéndice C

Documentación de usuario

C.1. Introducción

En este apartado veremos qué necesitamos para poder utilizar el proyecto y poder realizar mejoras, pruebas o cualquier modificación sobre el código entregado.

C.2. Requisitos de usuarios

Los requisitos para poder utilizar el software visto en este proyecto son los siguientes:

- Disponer de al menos dos equipos con sistema operativo Windows.
- Conexión a Internet.
- Un router que nos permita tener los dos equipos en red.
- Espacio suficiente para la instalación de todo el software. Esto depende del número y tipo de máquinas virtuales instaladas, en el caso del proyecto tal cuál, son necesarios al menos 10 GB para garantizar el correcto funcionamiento.

C.3. Instalación

A continuación, se muestra el proceso a seguir para poder tener el proyecto completo con todo el software requerido.

VirtualBox (versión instalada 5.2.18)

En primer lugar, para comenzar con el proyecto se ha instalado el software de virtualización VirtualBox en su versión más reciente a la fecha (5.2.18). Se trata de un programa gratuito desarrollado por Oracle. Gracias a ello podemos ejecutar el sistema operativo Ubuntu Server (en su versión 14.04.4 LTS Server). Podemos cargar estas máquinas más adelante en nuestro proyecto para poder simular elementos de la red. En la instalación inicial se han cargado dos máquinas virtuales de Ubuntu para futuros usos, que según ha ido creciendo el proyecto se han unido más máquinas, todas ellas con Ubuntu:

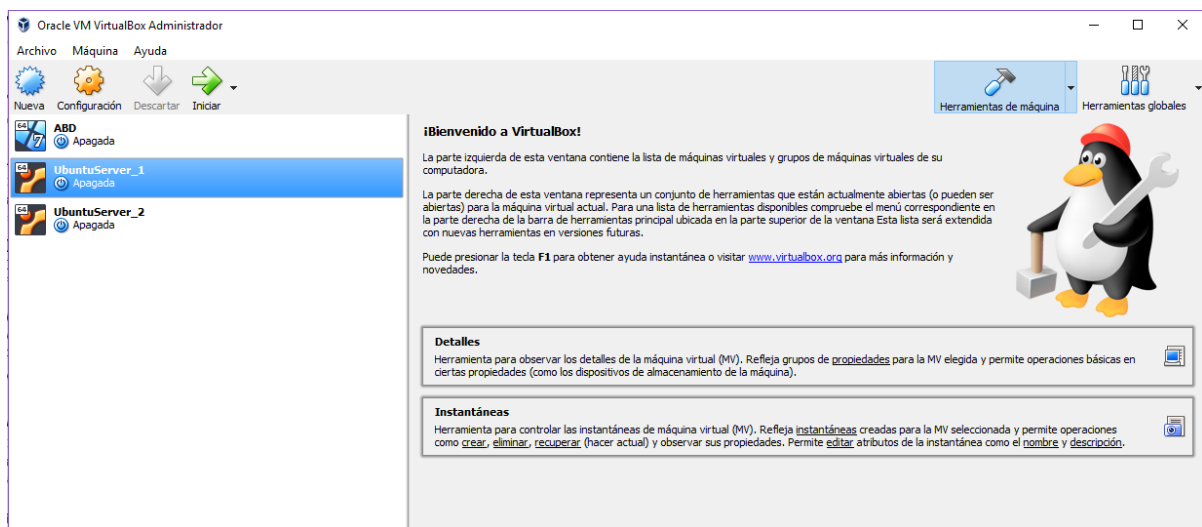


Ilustración 11: Oracle VM VirtualBox

VMWare Workstation 12 player

Otro software imprescindible para realizar el proyecto es VMWare en su versión Workstation 12 player (12.5.8). VMWare pertenece a la empresa EMC Corporation (Dell Inc), nos proporciona este software de virtualización totalmente gratis. Una vez instalado vamos a cargar dentro una virtualización del programa que veremos más adelante y en el cual se basa el proyecto: GNS3. Para poder conectar ambos programas deben estar en la misma versión, que es la versión 2.1.0.

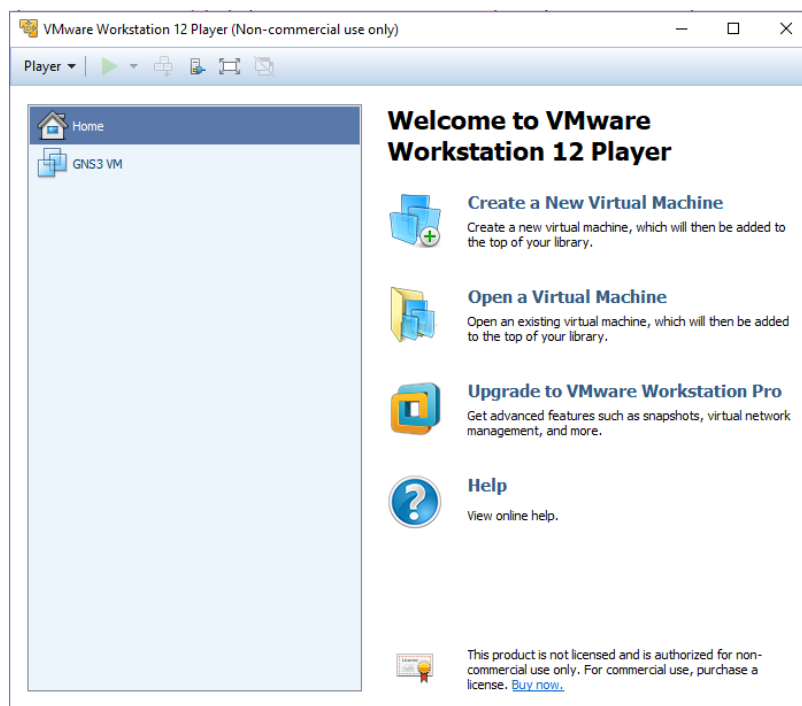


Ilustración 12: VMWare

GNS3 2.1.0

Se trata del programa principal sobre el que se desarrolla este proyecto. Es un software gratuito que permite la simulación gráfica de topologías de red muy complejas y su ejecución. Este programa funciona gracias a Dynamips, se trata de un emulador de sistemas operativos que nos permite ejecutar binarios de Cisco entre otros muchos.

Para su funcionamiento se sincroniza con VMWare y la máquina de GNS3 antes importada sobre este. De esta forma evitamos realizar toda la carga de recursos del programa sobre nuestro propio equipo y lo hacemos sobre el sistema de GNS3 de la máquina virtual. Si se ha realizado correctamente, nada más iniciar GNS3 se nos cargará la máquina de VMWare:

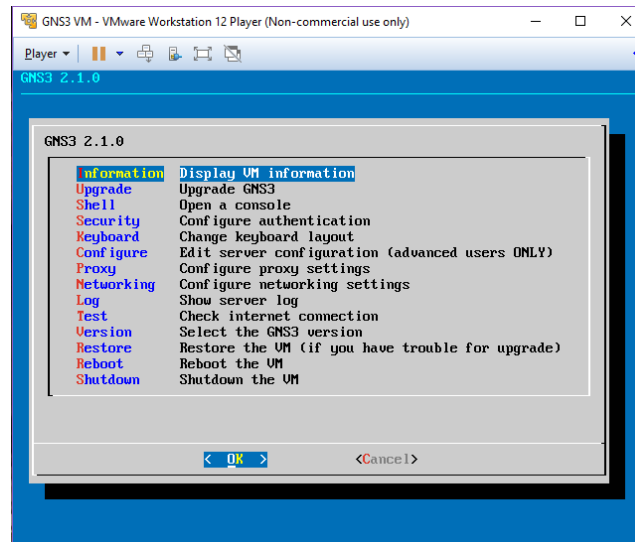


Ilustración 13: GNS3 VM

El entorno que nos encontramos nada más abrir el programa será el siguiente:

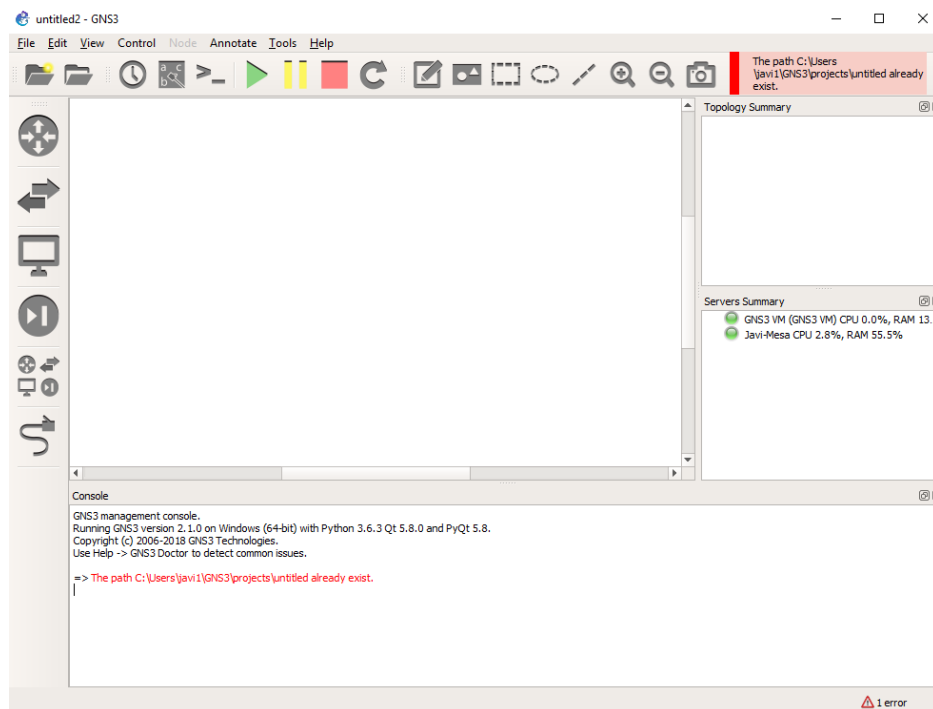


Ilustración 14: Entorno de GNS3

Como podemos ver en la parte derecha ya podemos escoger el sistema sobre el que trabajar, en primer lugar, vemos el sistema sobre GNS3 de VMWare y en segundo lugar tenemos el equipo local. Ya estamos preparados para comenzar a utilizar el programa.

C.4. Manual del usuario

Vemos la realización de todos los objetivos por separado de forma que nos permite llevar un manual de instalación de todos los componentes. A continuación, vemos el proceso a seguir:

Conectar GNS3 a redes externas (Internet)

Para la red queremos tener un punto de acceso a través del cual tener acceso con el exterior y poder conectar a internet. Para ello utilizamos un Router que con una configuración sencilla gracias a la potencia de GNS3 vamos a poder tener conexión a Internet.

Vamos a necesitar únicamente dos componentes, el mencionado Router y un elemento llamado cloud de GNS3.

La configuración necesaria es la que sigue:

En primer lugar, en el equipo vamos a las conexiones de red y entramos a las propiedades de red del adaptador que esté funcionando en nuestro caso, permitimos la conexión de otras redes a través de este equipo, esto nos permitirá escogerlo para nuestra red en GNS3.

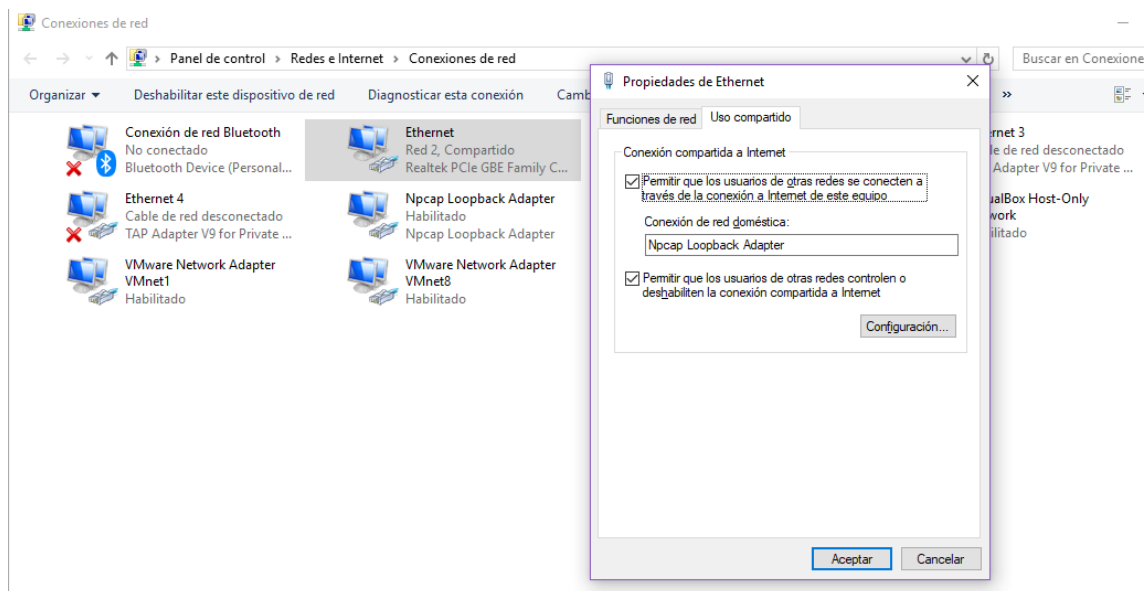
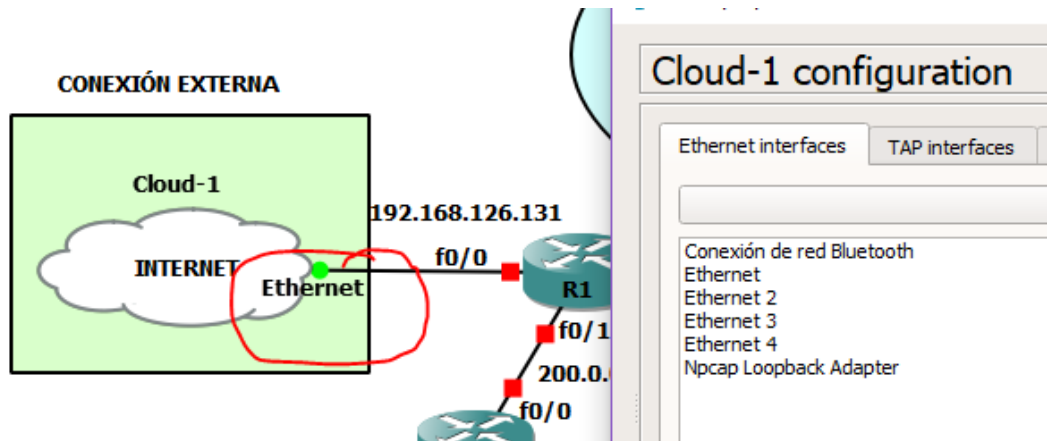


Ilustración 15: Compartir Internet en adaptador de red.

La nube deberemos configurarla para que su conexión Ethernet adquiera las propiedades de la tarjeta de red a través de la cuál nuestro equipo obtiene la conexión a Internet compartida del paso anterior, de

esta forma tendrá acceso al adaptador que ofrece Internet al equipo. En este caso es el adaptador Ethernet.



En el router (una vez lo hemos conectado a la nube y encendido) tan solo debemos activar el enlace utilizado, darle una dirección IP dentro de la subred de nuestro equipo y activar los comandos necesarios:

Damos al puerto asignado (f0/0) una dirección IP cualquiera de la red, para ello podemos utilizar DHCP, así adquiere automáticamente una IP:

```
ip address dhcp
ip domain-lookup
```

Una vez seguidos estos pasos podremos hacer ping a cualquier web:

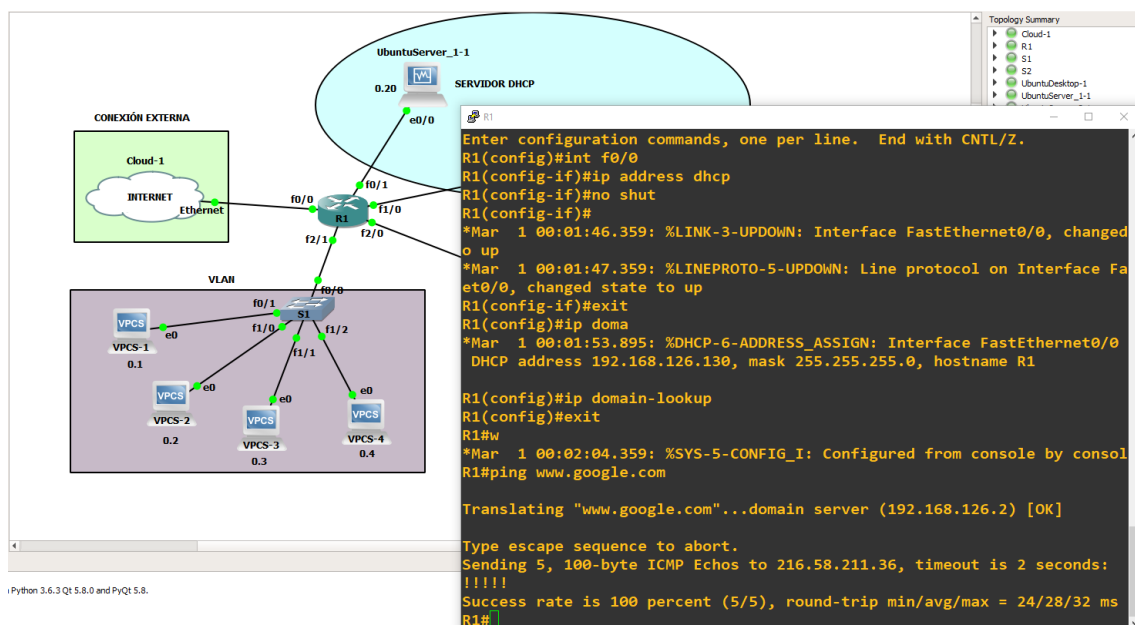


Ilustración 16: Conexión a Internet

Conectar GNS3 contra un servidor remoto

Se ha realizado el proceso de lanzar GNS3 en uno de mis equipos (portátil) contra otro equipo (mesa). Este proceso puede ser muy útil para casos en los que nuestra red empieza a crecer y un equipo carezca de recursos, de esta forma conseguimos que la carga de trabajando sea sobre el servidor y así el equipo en el que estamos trabajando sufra menos de esa carga.

El proceso es sencillo, una vez tenemos la instalación en los dos equipos como hemos visto en los pasos anteriores de instalación, nos fijamos en la IP que tenga cada equipo.

En mi caso en el equipo de mesa (que en este ejemplo hace de servidor):

```
C:\Users\javi>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Ethernet:

    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::5484:3917:b057:4009%10
    Dirección IPv4. . . . . : 192.168.1.11
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . . : 192.168.1.1
```

Ilustración 17: IP equipo Mesa

Y en el portátil:

```
Adaptador de LAN inalámbrica Wi-Fi:

    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::fd75:381f:6e4f:6cba%5
    Dirección IPv4. . . . . : 192.168.1.4
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . . : 192.168.1.1

C:\Users\Javi>
```

Ilustración 18: IP equipo Portátil

Una vez tenemos claro esto, en la instalación GNS3 del portátil, vamos a cambiar la forma de conectar el programa, desde la opción Setup Wizard:

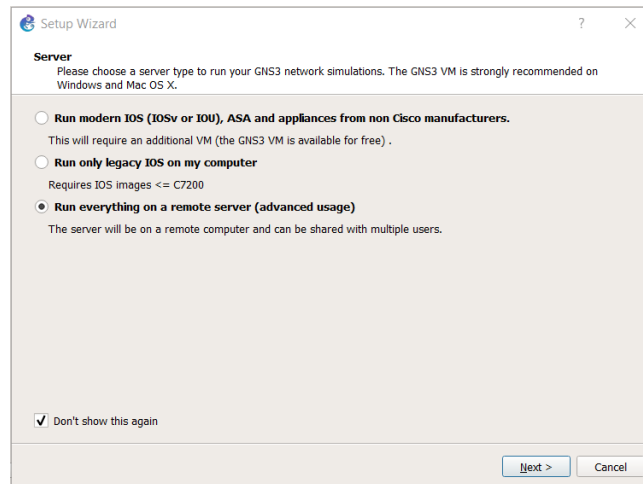


Ilustración 19: GNS3 configuración servidor remoto

Escogemos la opción de ejecutar sobre servidor remoto, como en la imagen.

Ahora debemos poner la dirección del servidor, para ello hemos buscado la IP de cada equipo, de esta forma ahora podemos introducirla, desmarcamos la opción de autenticación ya que nadie más va a tener acceso y el resto lo dejamos con las opciones por defecto.

En este último paso ya solo debemos confirmar que esta todo correcto, el programa conectará al servidor y podremos pulsar en finalizar, el resultado es el siguiente:

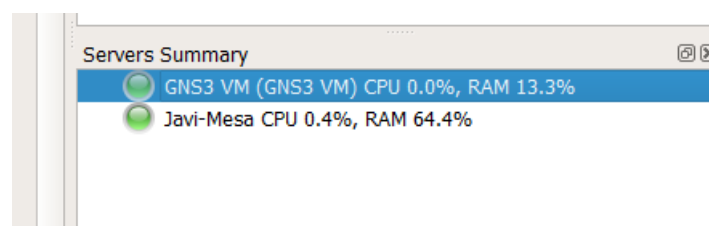


Ilustración 20: Opciones ejecución GNS3

Ya tenemos configurados nuestros equipos para que se ejecute GNS3 en remoto sobre el servidor.

Configuración de redes virtuales VLAN.

Para configurar redes virtuales vamos a utilizar un router con funciones de Switch como el que hemos creado en la instalación del programa. Para ello vamos a dividir la red en dos subredes virtuales:

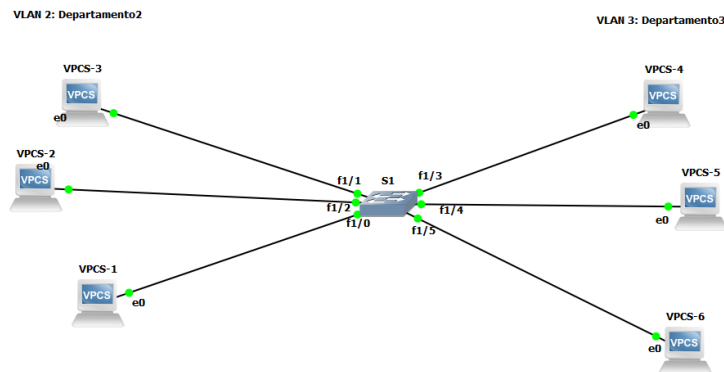


Ilustración 21: Redes VLAN

A partir de esta red, empezamos a configurar los equipos, le damos una IP a cada uno coincidiendo con su nombre, el VPCS-1 tendrá la IP 192.168.0.1, el 2 lo mismo, pero con la IP 192.168.0.2 y así sucesivamente. Esto lo hacemos mediante el comando:

```
Ip 192.168.0.x/24
```

Tras esto pasamos a configurar el Switch, activando todos los puertos a los que se conectan el resto de los equipos, utilizamos para ello:

```
Configure                                     terminal
Interface                                     f1/x
No shutdown
```

Una vez configurados los equipos y activados los enlaces vamos a crear las VLAN:

```
configure terminal
VLAN                                           2
name Departamento2
```

Ahora tendremos que asignar a cada puerto la VLAN a la que queramos que pertenezca:

```
interface f1/x
switchport access vlan 2

interface f1/y
switchport access vlan 3
```


Podemos ver la configuración de cada VLAN y que puertos tiene asignados utilizando:

```
Show vlan-sw
```

En el proyecto completo vamos a tener los elementos:

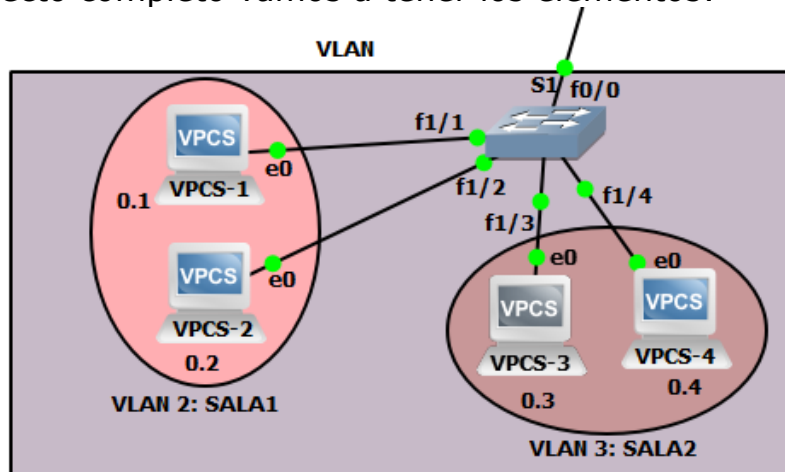


Ilustración 22: VLAN

Y probamos el acceso entre misma VLAN y vemos que falla con la red externa.

```
VPCS-3
VPCS> ping 192.168.0.1
host (192.168.0.1) not reachable

VPCS> ping 192.168.0.4
84 bytes from 192.168.0.4 icmp_seq=1 ttl=64 time=0.131 ms
84 bytes from 192.168.0.4 icmp_seq=2 ttl=64 time=0.208 ms
84 bytes from 192.168.0.4 icmp_seq=3 ttl=64 time=0.150 ms
84 bytes from 192.168.0.4 icmp_seq=4 ttl=64 time=0.178 ms
84 bytes from 192.168.0.4 icmp_seq=5 ttl=64 time=0.153 ms
```

Ilustración 23: Prueba VLAN

Ahora vamos a crear una VLAN de administración para poder gestionar el Switch. En primer lugar, vamos a configurar las conexiones telnet para permitir múltiples conexiones:

```
line vty 0 4
```

Creamos una contraseña de acceso:

```
pass 123456
login
```

Definimos la VLAN para administración y le damos una IP:

```
vlan 99
name admin
interface vlan 99
ip address 192.168.20.20 255.255.255.0
no shutdown
```

Incluimos la interface que vayamos a utilizar del Switch para administración en la VLAN:

```
interface FastEthernet1/15
switchport access vlan 99
```

Una vez tenemos la VLAN de administración lista, vamos a unir a ese puerto un Ubuntu server desde el cual poder conectarnos:

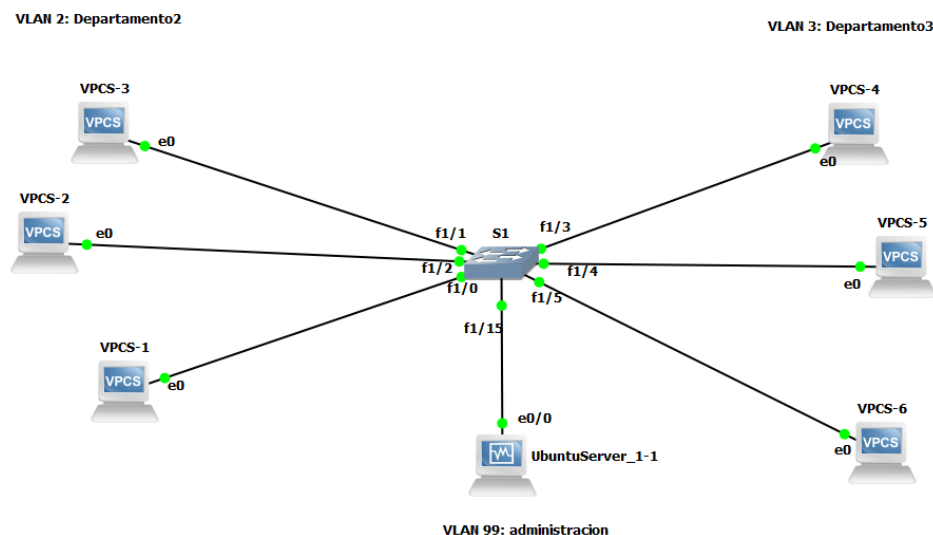


Ilustración 24: Redes VLAN con administración

Servidor Web con apache sobre Ubuntu.

1. Preparación de Ubuntu.

En primer lugar, vamos a configurar un cortafuegos que nos permite bloquear aquellos puertos que no vamos a necesitar, vamos a utilizar UFW. En principio únicamente queremos activarlo para utilizarlo más adelante:

```
Sudo ufw enable
```

2. Actualización de Ubuntu.

A continuación, realizo la actualización de Ubuntu con:

```
Sudo apt update
```

3. Instalación de Apache.

Tras la actualización, pasamos a la instalación de apache2, para ello vamos a ejecutar el comando:

```
sudo apt install apache2
```

esto nos pedirá varias confirmaciones y nos permitirá utilizar apache sobre nuestro Ubuntu.

4. Configurar el cortafuegos.

Vamos a ver las aplicaciones que tenemos:

```
Sudo ufw app list
```

Permitimos la conexión de apache a través del cortafuegos:

```
Sudo ufw allow Apache
```

```
javi@localhost:~$ sudo ufw allow Apache
Rule added
Rule added (v6)
```

Ilustración 25: Apache configuración

Y comprobamos el estado del cortafuegos:

```
Sudo ufw status
```

```
javi@localhost:~$ sudo ufw status
Status: active

To Action From
--
Apache ALLOW Anywhere
Apache (v6) ALLOW Anywhere (v6)
```

Ilustración 26: Apache configuración 2

Una vez tenemos el cortafuegos configurado podemos verificar que el servicio de Apache esta activo:

```
javi@localhost:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2018-11-28 20:23:17 UTC; 2min 43s ago
 Main PID: 3121 (apache2)
    Tasks: 55 (limit: 2317)
   Memory: 5.0M
    CGroup: /system.slice/apache2.service
            └─3121 /usr/sbin/apache2 -k start
              └─3123 /usr/sbin/apache2 -k start
                └─3124 /usr/sbin/apache2 -k start

Nov 28 20:23:17 localhost.localdomain systemd[1]: Starting The Apache HTTP Server...
Nov 28 20:23:17 localhost.localdomain systemd[1]: Started The Apache HTTP Server.
```

Ilustración 27: Apache configuración 3

Vamos a crear un sitio virtual para probar la conexión con el resto de los equipos. Para ello creamos un directorio dentro del bloque que trae por defecto apache:

```
sudo mkdir -p /var/www/ejemplo.com/html
```

Asignamos propietario al directorio:

```
sudo chown -R $USER:$USER /var/www/ejemplo.com/html
```

Cambiamos los permisos sobre el directorio para asegurarnos de que este correcto:

```
sudo chmod -R 755 /var/www/ejemplo.com
```

Vamos a crear una página de ejemplo tfg.html:

```
nano /var/www/ejemplo.com/html/tfg.html
```

Para poder visualizar este contenido lo tenemos que guardar en un espacio virtual con sus directivas. Vamos a crear este espacio:

```
sudo nano/etc/apache2/sites-available/ejemplo.com.conf
```

Y lo creamos con lo siguiente:

```
<VirtualHost *:80>
    ServerAdmin admin@ejemplo.com
    ServerName ejemplo.com
    ServerAlias www.ejemplo.com
    DocumentRoot /var/www/ejemplo.com/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Ahora necesitamos habilitar el archivo que acabamos de guardar:

```
sudo apache2ctl configtest
```

Reiniciamos el servicio apache para que los cambios surtan efecto:

```
sudo systemctl restart apache2
```

Ahora configuramos una red simple para hacer las pruebas y tratamos de llegar desde el cliente al servidor http. Accedemos utilizando el navegador links:

```
links http://192.168.0.2
```

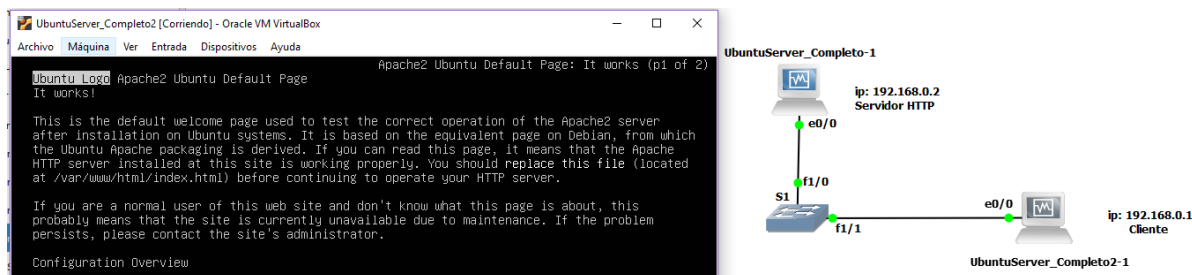


Ilustración 28: Prueba Apache

Servidor DHCP.

Vamos a configurar una máquina virtual con Ubuntu de forma que pueda utilizarse como un servidor DHCP.

En primer lugar, vamos a permitir la conexión a Internet de la máquina virtual, para ello vamos a la configuración de la máquina en VirtualBox y cambiamos el adaptador a NAT y habilitamos la opción cable conectado.

Una vez se tiene acceso a Internet, instalamos el servidor DHCP con el comando:

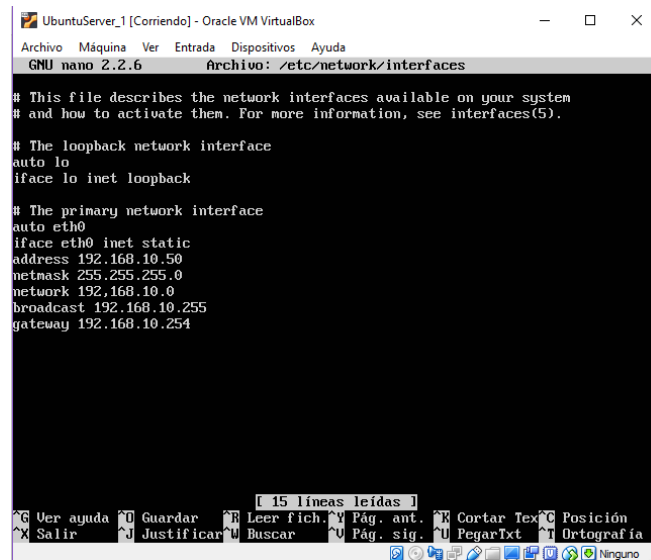
```
sudo apt-get install isc-dhcp-server
```

Tras la instalación quitamos los cambios del adaptador de red, sino nos dará error al tratar de utilizar la máquina en GNS3.

Ahora vamos a darle al servidor una dirección IP:

```
sudo nano /etc/network/interfaces
```

Y dejamos el fichero de la siguiente forma:



```
GNU nano 2.2.6 Archivo: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

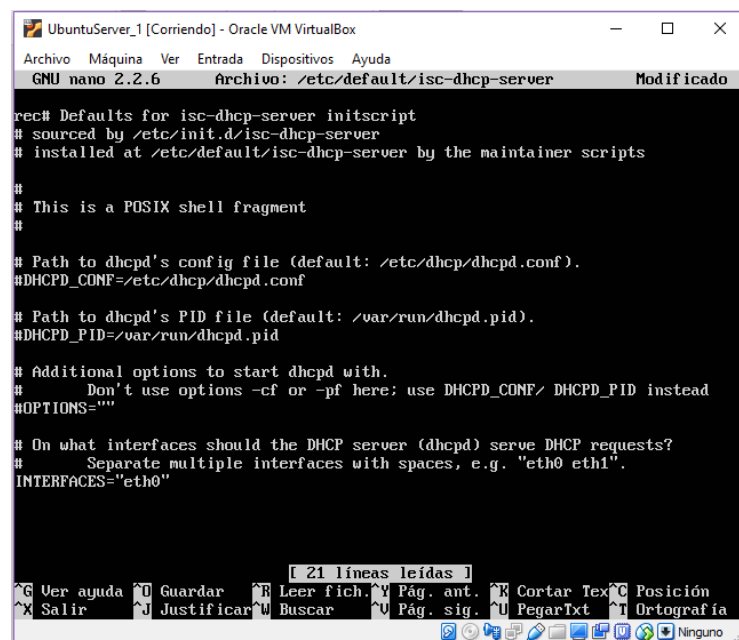
# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.10.50
netmask 255.255.255.0
network 192.168.10.0
broadcast 192.168.10.255
gateway 192.168.10.254
```

Ilustración 29: Fichero configuración DHCP

Como el servidor no puede asignarse una IP a sí mismo, por eso le damos la dirección estática a mano fuera del rango que dará a los clientes.

Ahora modificamos las interfaces por las cuales va a poder recibir peticiones. Modificamos el archivo:

```
sudo nano /etc/default/isc-dhcp-server
```



```
GNU nano 2.2.6 Archivo: /etc/default/isc-dhcp-server Modificado

rec# Defaults for isc-dhcp-server initscript
# sourced by /etc/init.d/isc-dhcp-server
# installed at /etc/default/isc-dhcp-server by the maintainer scripts

#
# This is a POSIX shell fragment
#

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPD_CONF=/etc/dhcp/dhcpd.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPD_PID=/var/run/dhcpd.pid

# Additional options to start dhcpd with.
# Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth0"
```

Ilustración 30: Fichero configuración DHCP 2

Ahora vamos a configurar el servidor DHCP en el archivo:

```
sudo nano /etc/dhcp/dhcpd.conf
```

```

GNU nano 2.2.6 Archivo: /etc/dhcp/dhcp.conf

#Configuration for an internal subnet.
subnet 192.168.10.0 netmask 255.255.255.0 {
    range 192.168.10.10 192.168.10.20;
    option routers 192.168.10.254;
    option broadcast address 192.168.10.255;
    default-lease-time 600;
    max-lease-time 7200;
}

[ 8 líneas leídas ]
Ver ayuda  Guardar  Leer fich.  Pág. ant.  Cortar Text  Posición
Salir  Justificar  Buscar  Pág. sig.  PegarText  Ortografía

```

Ilustración 31: Fichero configuración DHCP3

Con esto solo nos queda probar, creamos la red que se muestra a continuación:

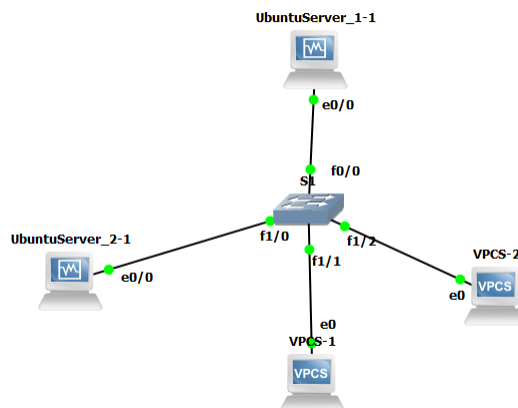


Ilustración 32: Topología DHCP

En los equipos con Ubuntu, para obtener la dirección mediante el servidor DHCP tenemos que configurar la interfaz, al igual que en el servidor, modificando el fichero `/etc/network/interfaces`:

```

GNU nano 2.2.6 Archivo: /etc/network/interfaces

# The primary network interface
auto eth0
iface eth0 inet dhcp

```

Ilustración 33: Fichero configuración DHCP 4

Para liberar la dirección se usará el comando:

```
sudo dhclient -r eth0
```

Y para cambiar la actual por otra:

```
sudo dhclient eth0
```

En los VPCs configurados el proceso es más sencillo aún, tan solo debemos escribir por consola:

```
dhcp
```

Para liberar la conexión:

```
dhcp -x
```

Para renovar la dirección:

```
dhcp -r
```


Servidor DHCP. Configurar retransmisor

Si nos encontramos en el caso de que el servidor DHCP no se encuentra en la misma subred que los equipos conectados, es necesario el uso de routers retransmisores, si no hacemos esto, los routers no reenvían los mensajes de difusión.

Esta es la razón por la que vamos a configurar un router como retransmisor DHCP o DHCP relay. Para que un router se comporte como DHCP relay vamos a especificar la interfaz de la subred en la que se encuentran los equipos y configurarlo para que la interfaz encuentre la IP del servidor DHCP.

Esto es muy sencillo, tan solo vamos a añadir el router entre la red y el servidor de la siguiente forma:

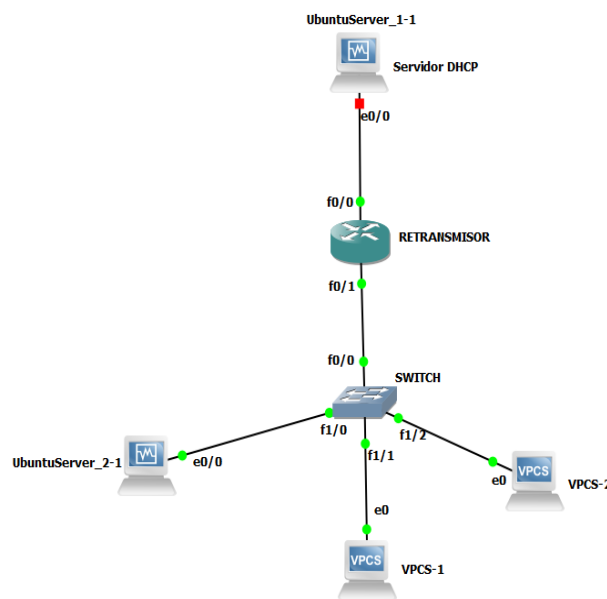


Ilustración 34: Retransmisor DHCP

En el router retransmisor vamos a hacer la configuración con los comandos:

```
enable
configure terminal
interface FastEthernet0/0
ip helper-address 192.168.0.20
```

Enrutamiento dinámico

Una vez tenemos la red final configurada con todos los elementos, debemos unirlos a través de un router que establezca los caminos. Para este caso vamos a emplear el enrutamiento dinámico RIP, el cual es capaz de establecer las rutas óptimas indicando todas las subredes conectadas.

Para ello necesitamos eliminar cualquier enrutamiento estático que se haya configurado previamente, podemos ver todas las rutas y después eliminarlas con los parámetros del router:

```
show ip route
configure terminal
no ip route 145.66.0.0 255.255.0.0
```

Vemos ahora las redes que debemos conectar a nuestro router:

192.168.0.0

192.168.2.0

192.168.10.0

192.168.20.0

Una vez las tenemos empleamos los comandos para activar el enrutamiento RIP (Donde x va a ser sustituido por cada subred que tenemos en nuestra red:

```
router rip
network 192.168.x.0
```

Configuración remota

Para realizar la configuración remota de los dispositivos de la red, vamos a utilizar Python, este lenguaje de programación nos permite utilizar Telnet, de forma que establecemos una conexión con el elemento a configurar. Lo lanzamos desde uno de los terminales Ubuntu con Python preinstalado y se lanzará por la red la llamada a la IP de cada equipo, la cual leerá del fichero de configuración

El fichero de configuración tendrá el formato siguiente:

Tipo de equipo,nombre del equipo,flag que indique si se usa dhcp o no,nombre de la interfaz,IP antigua,máscara antigua,puerta de enlace antigua,IP nueva, máscara nueva,puerta de enlace nueva

Por ejemplo, dos de las líneas podrían ser:

PC,UbuntuDesktop-1,e0/0/0,192.168.2.6,255.255.255.0,
192.168.2.15,192.168.2.16,255.255.255.0,192.168.2.15

y

Router,R1,f0/0/0,192.168.0.254,255.255.255.0, ,192.168.10.254,255.
.255.255.0,

Una vez tenemos el fichero de configuración listo, lo que hace el programa es leer dicho fichero y guardar los datos que contiene en un array de arrays, donde cada array será una de las líneas leídas en el programa y que nos va a permitir leer fácilmente todos los campos de dicha fila.

```
archivo = open("fichero.txt")
archivo.seek(0)
datos = archivo.read()
archivo.close()

#Esta función nos permite guardar los datos leídos del fichero en un array de arrays.
array = np.genfromtxt(StringIO(datos), delimiter=",", dtype="|U20", autostrip=True)
#Numero de interfaces a configurar
numConfigs = len(array)
```

Ilustración 35: Leer fichero desde Python

El programa tiene una función interna que realiza la conexión mediante telnet pasando como parámetro la IP sobre la que lanzar la conexión y el número de fila del fichero. Una vez recibida dicha IP y número de fila, comprueba si es DHCP o no, en caso de serlo avisa y no lanza

telnet, en caso de no ser DHCP, lanza las configuraciones oportunas comprobando si se trata de un PC o de un Router.

```
'''
telnet(ip,numFila)
Funcion que recibe una dirección IP y el número de fila del fichero en que se encuentra la ejecución
y realiza una conexión mediante telnet para configurar las interfaces del componente.
Autor: Javier García González
'''

def telnet(ip,numFila):
    wait = 2
    con = telnetlib.Telnet(ip, 23, 5)

    #Se comprueba si es DHCP, en caso afirmativo no se cambia nada.
    if array[numFila][3] == 1:
        print("Es DHCP, no se configura" + "\n")
    else:
        #Se comprueba el tipo de equipo que es, Router o PC
        if array[numFila][0] == 'Router':

            con.write("configure terminal" + "\n")
            time.sleep(wait)
            con.write("int " + array[numFila][2] + "\n")
            time.sleep(wait)
            con.write("ip address " + array[numFila][7] + array[numFila][8] + "\n")
            time.sleep(wait)
            con.write("no shutdown" + "\n")
            time.sleep(wait)
            con.write("exit" + "\n")
            time.sleep(wait)

        if array[numFila][0] == 'PC':

            con.write("ifconfig " + array[numFila][2] + " " + array[numFila][7] + " netmask "
                    + array[numFila][8] + " broadcast " + array[numFila][9] + " up" + "\n")
            time.sleep(wait)
```

Ilustración 36: Función telnet.

Para lanzar estas configuraciones, el programa dispone de un menú que nos permite escoger entre tres opciones:

- Mostrar el contenido del fichero.
- Lanzar la configuración.
- Salir del programa.

```

'''
menu()
Funciona que nos muestra una serie de opciones para ver leer el fichero de configuración
o lanzar las configuraciones.

Autor: Javier García González
'''
def menu():

    os.system('cls')

    print("Opciones")
    print("\t1 - Leer configuración")
    print("\t2 - Lanzar Configuración")
    print("\t9 - Salir")

while True:
    menu()
    opcionMenu = input("Inserta numero -> ")
    if opcionMenu == "1":

        print("ifconfig " + array[6][2] + " " + array[6][7] + " netmask " + array[6][8] +
            " broadcast " + array[6][9] + " up" + "\n")

        break

    elif opcionMenu == "2":
        print("")
        input("Lanzando configuración...\n pulsa enter para continuar")
        j=0
        while j <= numConfigs-1:
            telnet(array[j][4],j)
            j+=1
        print("Configuración realizada correctamente.")

        break

    elif opcionMenu == "9":
        print("Cerrando conexión...")
        break

    else:
        print("")
        input("No has pulsado ninguna opción correcta...\npulsa una tecla para volver al menú")

```

Ilustración 37: Función menú.



Esta obra está bajo una licencia Creative Commons Reconocimiento 4.0 Internacional ([CC- BY-4.0](https://creativecommons.org/licenses/by/4.0/)).