UNIVERSIDAD DEL BÍO-BÍO

Facultad de Ciencias Empresariales

"*Interfaz gráfica para la creación de redes en el emulador Mininet*"

Sebastián Nicolás Acuña González
Profesor Guía: Patricio Galdames Sepúlveda

Proyecto de título para optar al Título de
Ingeniero Civil en Informática

Febrero 27, 2015
Concepción - Chile

Agradecimientos

Agradezco a dios a mi familia, seres amados, docentes y compañeros de estudio por el apoyo brindado durante todo el proceso universitario de estos últimos años. Todo y cada uno de ellos fueron de suma importancia academica y emotiva para finalizar este proceso.

Resumen

Este proyecto se presenta para dar conformidad a los requisitos exigidos por la Universidad del Bío-Bío, en el proceso de titulación para la carrera de Ingeniería Civil Informática.

"Mininet" es un software que permite emular redes en un simple ordenador, a pesar de ser una herramienta potente en el entorno de redes, posee una escasa documentación, tanto en su utilización como en su API.

El proyecto titulado *"Interfaz gráfica para la creación de redes en el emulador Mininet"* brinda la generación de script por medio de una interfaz gráfica funcional en donde apoya la creación y simulación de redes para el software "Mininet".

Abstract

This Project present a conformation for the requirements of the University of Bio – Bio, in the process to obtain a certification in Informations an Civic Engineering.

"Mininet" is a software that allow networks to be emulated on a computer, to also be used as potencial tool in the environment of networks, is also has little documentation in the use of the software and his API"

The project tittled "Graphic interface for the creation of networks in Mininet emulator" provides a generation of scripts through of a functional graphic interface that supports the creation and simulation of networks for the software "Mininet".

Índice General

ÍNDICE DE TABLAS

Índice de Figuras

1 INTRODUCCIÓN

Las redes tal como están configuradas actualmente se presentan un poco obsoletas a los requerimientos de las telecomunicaciones actuales, debido a limitaciones como: depencia de los provedores de internet, difícil escalabilidad (cientos de miles de nodos que deben ser configurados y gestionados), políticas inconsistentes (por ejemplo la configuración de una maquina virtual puede llevar horas o en algunas casos días reconfigurando lista de accesos en toda la red), complejidad, imposibilidad de asignar calidad al trafico de acuerdo a su importancia, falta de cooperacion entre redes y nuevas tendencias tecnológicas tienen como resultado que internet se haya convertido en un cuello de botella para satisfacer las necesidades actuales. Por ejemplo, en cloud computing los datos de una empresa no residen en ella y esto hace que la información sea vulnerable a la copia de terceros, la centralización de aplicaciones y almacenamiento de datos genera una dependencia de los proveedores de servicios, la información debe recorrer diferentes nodos (cada uno un foco de vulnerabilidad) antes de llegar a su destino y la utilización de protocolos de seguridad como HTTPS disminuyen la velocidad de la red debido a la sobrecarga que requieren.

Para lograr que los proveedores de internet y las empresas de telecomunicaciones brinden servicios de mayor calidad, flexibilidad y control, es necesario migrar a un paradigma. SDN (Redes definidas por software)[15] es un nuevo paradigma que une un conjunto de técnicas del área de redes computacionales, cuyo objeto es facilitar la implementación e implantación de servicios de red, evitando al administrador de la red gestionar dichos servicios a bajo nivel, todo esto se consigue mediante la separación del plano de control (inteligencia de un elemento de red, ejemplo: el software responsable por definir los procesos de enrutamiento, la política de seguridad y la ingeniería de tráfico.) del plano de datos (responsable del envío de paquetes).

SDN se crea en respuesta a data center de gran tamaño y redes que presentan patrones de trafico impredecibles en un corto lapso de tiempo, estas requerien una alta demanda de recursos particulares que con la infraestructura de red actual no se puede satisfacer. Google ha desarrollado una implementación de SDN en donde a partir de las arquitecturas compartidas de Google SDN crea la ilusión de aplicaciones y servicios ejecutadas en las propias redes de los usuarios [20].

Bob Lantz desarrolla el software "Mininet", capaz de emular rápidamente toda una red en un ordenador, dando la base para la creación de prototipos de redes definidas por software [16].

El proyecto de título denominado "*Interfaz gráfica para la creación de redes en el emulador Mininet*" presenta el desarrollo de una interfaz gráfica para el software "Mininet" que permite la generación de scripts para la creación de redes en "Mininet".

El presente documento está organizado mediante secciones: la sección número 2, describe la institución para la cual está desarrollado el software; la sección número 2.3, describe la problemática y el porqué del generador; la sección número 3, describe una serie de conocimientos previos explicando brevemente que es "Mininet" (Características y Limitaciones), que es "OpenFlow" y la explicación de la API de "Mininet"; la sección número 4, describe la definición del proyecto; la sección número 5, define la especificación de requerimientos del software; la sección número 6, describe el análisis del proyecto de

software (Casos de Uso, Especificación y Modelamiento de datos); la sección número 7, especifica el diseño de interfaces, navegación y menús; la sección número 8, especifica la sección de pruebas del proyecto; la sección número 9, presenta los casos de pruebas en donde el generador es puesto a prueba; la sección número 10, presenta las conclusiones del proyecto de título; la sección número 11, enuncia la bibliografía utilizada para el proyecto; la sección número 13 y 14, describen los anexos en donde es presentada la Carta Gantt y los manuales de usuario.

2 DEFINICIÓN DE LA INSTITUCIÓN

2.1 Descripción de la Institución

2.1.1 Antecedentes de la Institución

Nombre:	**Universidad del Bío - Bío**
Rol:	60911006-6
Rubro:	Educación Superior
Dirección	Concepción: Avda. Collao 1202
Teléfono:	(56-41)3111200
Sitio Web:	www.ubb.cl
Email:	ubb@ubiobio.cl
Representante Legal:	Héctor Guillermo Gaete Feres

Tabla 1 Antecedentes de la Institución

2.1.2 Misión

La Universidad del Bío - Bío es una institución de educación superior, pública, estatal y autónoma, de carácter regional, que se ha propuesto por misión:

Formar profesionales de excelencia capaces de dar respuesta a los desafíos de futuro, con un modelo educativo cuyo propósito es la formación integral del estudiante a partir de su realidad y sus potencialidades, promoviendo la movilidad social y la realización personal.

Fomentar la generación de conocimiento avanzado mediante la realización y la integración de actividades de formación de postgrado e investigación fundamental, aplicada y de desarrollo, vinculadas con el sector productivo, orientadas a áreas estratégicas regionales y nacionales Contribuir al desarrollo armónico y sustentable de la Región del Biobío, a través de la aplicación del conocimiento, formación continua y extensión, contribuyendo a la innovación, productividad y competitividad de organizaciones, ampliando el capital cultural de las personas, actuando de manera interactiva con el entorno y procurando la igualdad de oportunidades. Desarrollar una gestión académica y administrativa moderna, eficiente, eficaz y oportuna, centrada en el estudiante, con estándares de calidad certificada que le permiten destacarse a nivel nacional y avanzar en la internacionalización

2.1.3 Visión

Ser reconocida a nivel nacional como una Universidad estatal, pública, regional, autónoma, compleja e innovadora con énfasis en la formación de capital humano, vinculada al desarrollo sustentable de la Región del Biobío y que aporta a la sociedad del conocimiento y al desarrollo armónico del país.

2.1.4 Estructura Organizativa

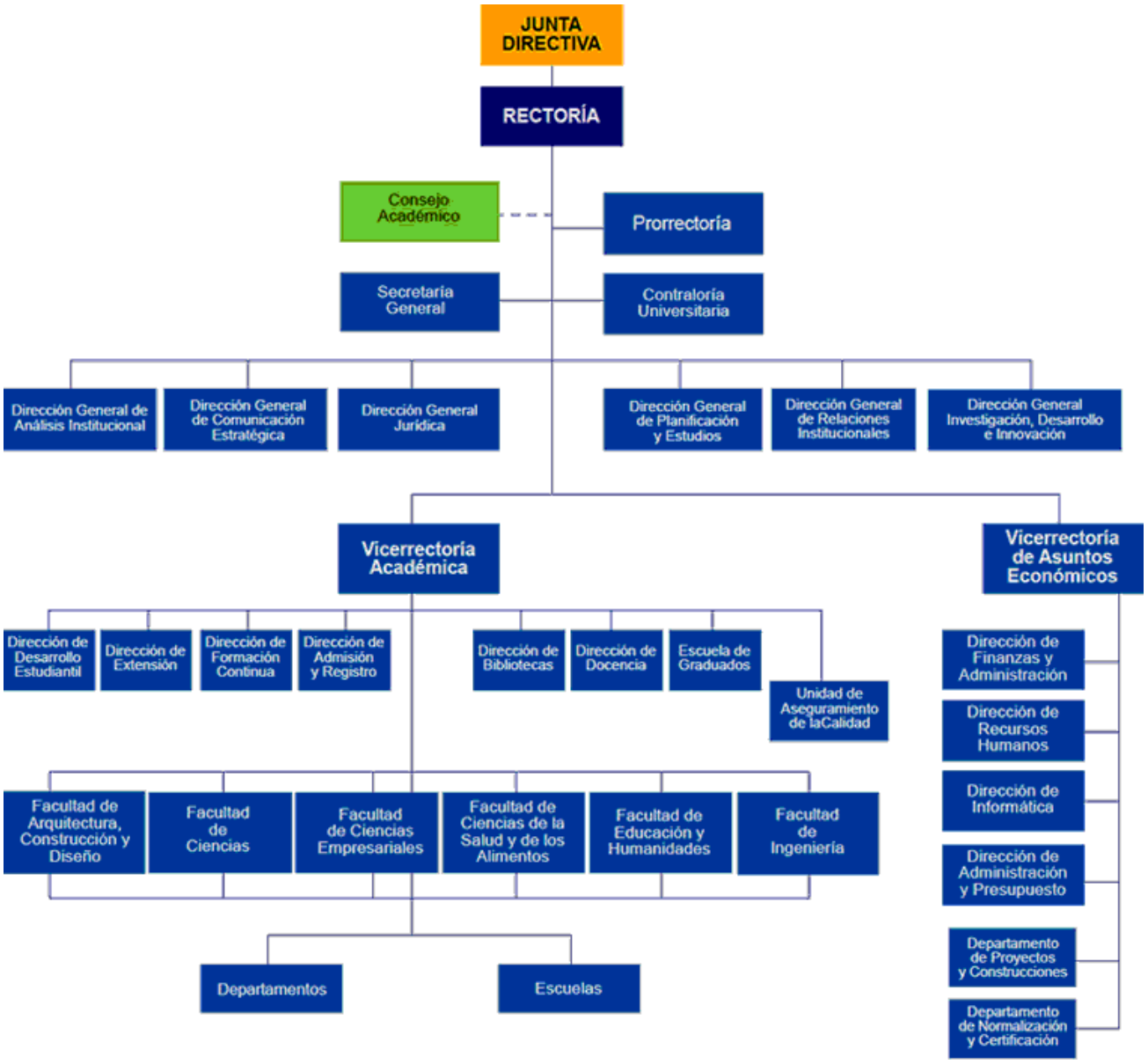


Figura 1. Organigrama "*Universidad del Bío - Bío*"

2.1.5 Objetivos de la Institución

2.1.5.1 Objetivo General

Formar profesionales competentes y proactivos en didáctica de la lengua materna, desde una perspectiva innovadora, analítico-crítica, creativa, indagadora y resolutiva en cuanto a las problemáticas del aprendizaje de las competencias comunicativas (discursivas, literarias, multimodales), en diferentes contextos, situaciones y escenarios sociales y educativos, con una visión integradora, transversal e interdisciplinaria.

2.1.5.2 Objetivos Específicos

- Generar información científica en investigaciones referidas a las problemáticas del aprendizaje de la lengua materna en diversos contextos y situaciones educativas, con un riguroso análisis crítico.

- Profundizar en el estudio de nudos críticos del aprendizaje de las competencias comunicativas en diferentes escenarios, con un posicionamiento teórico y proactividad en la toma de decisiones educacionales.

2.2 Descripción del área de estudio

2.2.1 Área

El software está destinado a ser utilizado en el "*Departamento de Sistema de Información"* perteneciente a la *"Facultad de Ciencias Empresariales"* de la "*Universidad del Bio – Bio"* para fines académicos e investigación.

2.2.2 Objetivos del Área

- Desarrollo académico de la Ciencias de la Computación e Informática y la participación activa en la formación, capacitación y asistencia técnica de profesionales en informática, particularmente en las áreas de la informática aplicada a la gestión.

- Creciente actividad de investigación relevante, buscando la formación de equipos en áreas donde se pueda destacar.

- Permanente perfeccionamiento de sus cuadros académicos y capacitación del personal administrativo, para el mejoramiento de la calidad.

- Establecimiento de relaciones con el entorno, que permita una mayor pertinencia y contribución a su quehacer.

2.2.3 Estructura organizacional

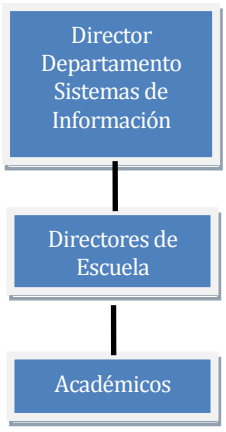


Figura 2. Organigrama Departamento Sistemas de Información

2.3 Descripción de la problemática

Si bien "Mininet" dispone de un editor basado en su API en Python, su utilización se torna compleja al momento de realizar simulaciones y recuperación de datos simulados sobre los nodos creados en la red debido a la necesidad de programación y un entendimiento de la API poco documentada de "Mininet". Además, no posee ciertas características y módulos implementados como: Ediciones de mac y máscaras en sus nodos, creación de vlans, validación de campos (que al no validarlos podrían generar problemas en los scripts creados) y generación de directorios persistentes y temporales. Dados estos inconvenientes, resulta adecuado contar con un editor que permita generar los scripts de forma más didáctica y rápida. También, se espera que este proyecto permita mejorar la forma de presentación del contenido en un curso de redes avanzado, disminuir la complejidad de enseñar el paradigma de SDN con la herramienta actual y facilitar la investigación de nuevas aplicaciones a desarrollar en redes definidas por software.

Lo anterior se debe a que las redes creadas en "Mininet" son compatibles con SDN (Redes Definidas por Software), dichas redes permiten que el plano de control (inteligencia de un elemento de red, ejemplo: el software responsable por definir los procesos de enrutamiento, la política de seguridad y la ingeniería de tráfico) este separado del plano de datos (responsable del envío de paquetes) brindando inteligencia programable, otorgando beneficios como: reducción de costos debido a la posibilidad de reutilización de hardware existentes, permite a los desarrolladores enfocarse al desarrollo de las aplicaciones y no a las configuraciones especificas de cada proveedor de servicios e incentiva la innovación en las redes con el desarrollo de nuevas aplicaciones.

2.3.1 Desafios de la problemática

"Mininet" es una herramienta relativamente nueva (Su primera versión fue lanzada el 14 de Junio del 2012[18]), su documentación pública y oficial actual data de comienzos del 10 de Mayo del 2013 [19], por lo que presenta algunos desafios como:

- Dificultad al profundizar en la utilización de la herramienta, esto se debe a la poca documentación que hay sobre "Mininet" obligando a entrar al código fuente.

- A finales de 2014 en la documentación aun muchos métodos de las clases no indican tipo de dato en los parámetros de entrada y retorno.

- Escases de ejemplos avanzados en la utilización de la herramienta.

- Poca información de errores o "bugs" de la herramienta (hoy en día el único soporte eficaz y semi-oficial existente es el mailist de la "Stanford University" sobre"Mininet", en donde las problemáticas presentadas no siempre son respondidas).

3 CONOCIMIENTOS PREVIOS

3.1 ¿Qué es Mininet?

Mininet es un software que permite simular toda una red de computadoras en un simple kernel en Linux, permitiendo la creación de host, switches, enlaces y controladores en un simple ordenador. Lo anterior se logra por medio de técnicas de virtualización y procesos como espacios de nombres, Veth (Dispositivos Ethernet Virtuales) y pipes (Comunicación entre Procesos en Linux), esto permite que cada nodo posea sus propias interfaces de redes, tablas de ruteo y tablas ARP separadas, el cual simula un host virtual como si fuera uno real proporcionando la posibilidad de ejecutar cualquier aplicación o código sobre los nodos. [12]

Actualmente "Mininet" está en pleno desarrollo y es utilizado en ámbitos como la investigación, el desarrollo, el aprendizaje, el prototipado, las pruebas y depuración junto con OpenFlow.

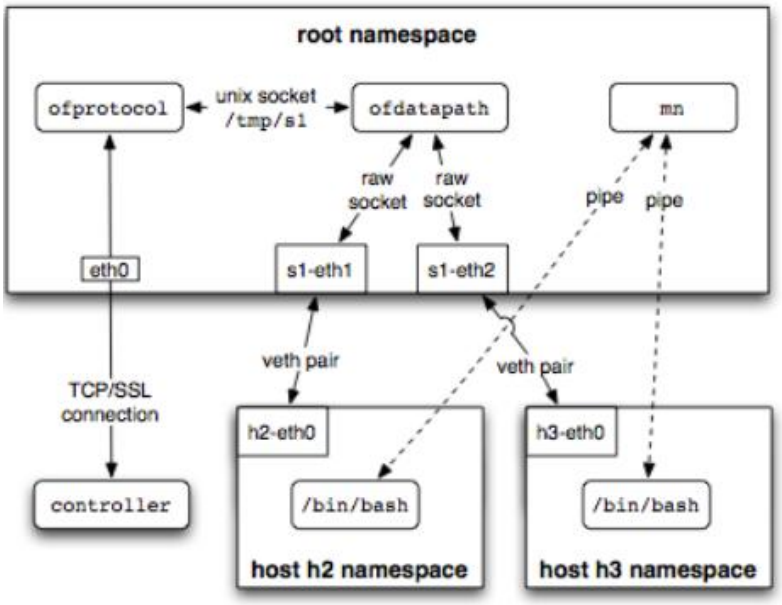


Figura 3. Esquema Mininet [10]

3.2 ¿Qué es OpenFlow?

OpenFlow es un protocolo abierto de comunicaciones, este es simplemente un elemento que forma parte de la arquitectura de SDN, siendo el mismo un protocolo que permite al plano de control (servidores SDN programables) interactuar con la red en el plano de datos para especificar cómo el tráfico fluye por la red por medio de tablas de flujo. [17]

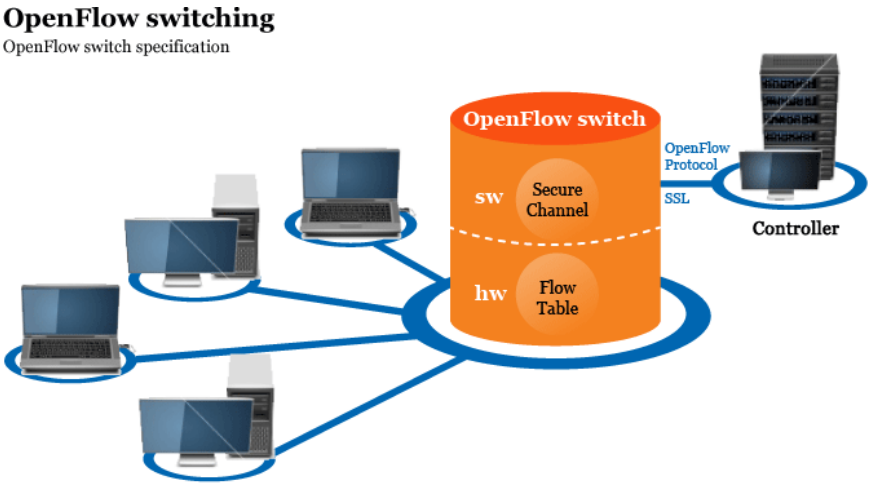


Figura 4. Esquema OpenFlow [9]

3.3 Características

"Mininet" posee las siguientes características principales:

- Proporciona una manera simple y económica para el desarrollo y testeo de redes de ordenadores o redes OpenFlow.
- Permite que múltiples personas puedan trabajar de forma concurrente sobre una misma topología.
- Soporta pruebas de regresión a nivel de sistema que son fácilmente implementadas
- Permite realizar pruebas sobre una topología de red completa sin tener la necesidad de cablear una red física.
- Incluye una línea de comandos que permite interactuar con los nodos de la red.
- Permite crear topologías personalizadas de forma arbitraria e incluye un conjunto básico de topologías parametrizables.
- Es posible crear redes sencillas pero no personalizables sin necesidad de programación por medio de una línea de comandos.
- Posee una API en Python para la creación y experimentación de redes.

3.4 Limitaciones

- Al ejecutarse "Mininet" en un solo sistema se dan limitaciones en los recursos, esto quiere decir que cada host y switch virtual compartirá los recursos del ordenador de forma equivalente.
- Al utilizar solo un núcleo de Linux para todos los host virtuales no se permite la ejecución de software que depende de BSD(Berkeley Software Distribution) o Windows u otro núcleo de algún sistema operativo. No obstante se puede ejecutar Mininet dentro de una máquina virtual con una distribución en Linux.
- Actualmente los host de "Mininet" no hacen NAT (Network Address Translation) por defecto. Eso quiere decir que las redes creadas en Mininet son aisladas y no tienen comunicación con internet, a menos que se proporcione un medio para que lo hagan.
- Todos los host de "Mininet" comparten el sistema de archivos del ordenador y el espacio PID (Identificador de Procesos), esto quiere decir que se debe tener sumo cuidado en la ejecución de procesos demonios que requieran configuración en el directorio /etc de Linux.
- A diferencia de un simulador, "Mininet" no tiene una fuerte noción de tiempo virtual lo que significa que las medidas de temporización son basadas en tiempo real, esto implica que redes de 100Gbps no pueden ser fácilmente emuladas.

- Los enlaces de "Mininet" solo se dan por medio de switch virtuales, esto significa que no se pueden realizar conexiones punto a punto entre host o controladores.

- "Mininet" no programa los controladores, estos deben ser programados de forma separada.

- La modificación manual de IPs en los switch de "Mininet" provocan inconsistencias en las redes creadas, por ende su modificación solo se debe realizar si es estrictamente necesario.

- Actualmente en la ejecución de los scripts la recuperación de datos solo se puede dar por medio del comando ">" de una terminal en Linux.

- Herramientas como traceroute o ethtool no funcionan en las interfaces virtuales de "Mininet".

- Comandos como siege (usado para generar ataques de denegación de servicio) no son permitidos por errores de conexión de sockets

3.5 API Mininet

El siguiente ítem explica la API en Python de Mininet. Se presentara el nombre del archivo, sus clases, constructores, parámetros, métodos y algunos ejemplos a las clases más importantes con observaciones cuando sea pertinente.

3.5.1 Niveles API Mininet

Mininet posee clases como: Topo, Mininet, Host, Switch y Link y estas sus respectivas subclases, es conveniente dividir estas clases en niveles para un mejor entendimiento.

- **API Nivel bajo:** La API de bajo nivel consiste en las clases bases como node y link (Host, Switch y Link y sus subclases). Aquí se utilizan instancias individuales para crear una red.

Ejemplo:

```
h1 = Host( 'h1' )
h2 = Host( 'h2' )
s1 = OVSSwitch( 's1', inNamespace =False )
c0 = Controller( 'c0',inNamespace =False )
Link( h1, s1 )
Link( h2, s1 )
h1.setIP( '10.1/8' )
h2.setIP( '10.2/8' )
c0.start()
s1.start( [ c0 ] )
print h1.cmd( 'ping -c1', h2.IP() )
s1.stop()
c0.stop()
```

- **API Nivel medio:** La API de mediano nivel añade los objetos Mininet que sirven como contenedores para los nodos y sus enlaces. Esta proporciona una serie de métodos (AddHost(), AddSwitch(), addLink()) para agregar nodos y enlaces a la red, además ofrece una serie de métodos de configuración (en particular start(), stop() entre otros). La API de mediano nivel posee la mayor maniobrabilidad entre todos los niveles.

Ejemplo:

```
net = Mininet()
h1 = net.addHost( 'h1' )
h2 = net.addHost( 'h2' )
s1 = net.addSwitch( 's1' )
c0 = net.addController( 'c0' )
net.addLink( h1, s1 )
net.addLink( h2, s1 )
```

```
net.start()
print h1.cmd( 'ping -c1', h2.IP() )
CLI( net )
net.stop()
```

- **API de Alto Nivel:** La API de alto nivel añade plantillas de una topología, la clase Topo proporciona la capacidad de crear dichas plantillas reutilizables y parametrisadas, luego pueden ejecutarse en la consola de Linux con el comando mn (mn –custom topo.py).

Ejemplo:

```
class SingleSwitchTopo( Topo ):
    "Single Switch Topology"
    def __init__( self, count=1, **params ):
        Topo.__init__( self, **params )
        hosts = [ self.addHost( 'h%d' % i )
                  for i in range( 1, count + 1 ) ]
        s1 = self.addSwitch( 's1' )
        for h in hosts:
            self.addLink( h, s1 )

net = Mininet( topo=SingleSwitchTopo( 3 ) )
net.start()
CLI( net )
net.stop()
```

Como se puede observar la API de nivel medio es mas flexible ya que no requiere la creación de una clase topología (API alto nivel) y es mas potente al no poseer la rigidez que caracteriza a la API de bajo nivel.

3.5.2 Archivo net.py

3.5.2.1 Clase Mininet

3.5.2.1.1 Diagrama de Herencia

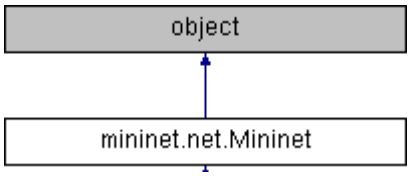


Figura 5. Herencia Clase Mininet

3.5.2.1.2 Constructor

```
def mininet.net.Mininet.__init__     (self,
                    topo = None,
                    switch = OVSKernelSwitch,
                    host = Host,
                    controller = DefaultController,
                    link = Link,
                    intf = Intf,
                    build = True,
                    xterms = False,
                    cleanup = False,
                    ipBase = '10.0.0.0/8',
                    inNamespace = False,
                    autoSetMacs = False,
                    autoStaticArp = False,
                    autoPinCpus = False,
                    listenPort = None,
                    waitConnected = False
      )
```

Descripción: Constructor de la clase Mininet, este crea la base de la red para la posterior creación de nodos.

Parámetros	Tipo	Opcional	Descripción
Topo	Topo	SI	Indica si la red posee una topología dentro de su construcción que puede ser de tipo None (ninguno) o bien de la clase Topo .
switch	Switch	SI	Indica el switch por default que tendrá la topología, por defecto es el switch OVSKernelSwitch.
Host	Host	SI	Indica la clase por defecto de los host utilizados en la red.
controller	Controller	SI	Indica el tipo de controlador por defecto en la red.
Link	Link	SI	Indica el tipo de links por defecto que tendrá la red.
Intf	Intf	SI	Indica las Interfaces por defecto que posera la red.
build	Booleano	SI	Indica si la red se construirá a partir de alguna topología.
xterms	Booleano	SI	Indica que si anteriormente es construida la red, esta debe o no abrir terminales externas xterm en cada nodo de la red generada.
cleanup	Booleano	SI	Indica si limpia o no las terminales una vez construida la red
ipBase	String	SI	Indica la ip y mascara base de la red
autoSetMacs	Booleano	SI	Indica si las direcciones MAC de los host serán de forma aleatoria o de forma ordenada (00:00:00:00:01, 00:00:00:00:00:02 etc)
autoStaticArp	Booleano	SI	Si es verdadero, automáticamente las tablas ARP se generaran en cada host
listenPort	Int	SI	Puerto Base de escucha para la comunicación con los switchs
waitConnected	Booleano	SI	Atributo que si es verdadero espera 5 segundos a que cada switch se conecte a un controlador
autoPinCpus	Booleano	SI	Atributo no implementado aun, indicara si los host realizan un acoplamiento a las cpu reales del ordenador.

Tabla 2. Descripción de parámetros constructor clase Mininet

Ejemplo:

```
net=
Mininet(topo=None,build=False,ipBase='10.0.0.0/8',waitConnected=True,au
toSetMacs=True)
```

El siguiente ejemplo es la primera línea que debe poseer cualquier script de "Mininet", indicando las características generales de la red.

Este ejemplo indica que la red no tendrá una topología definida, que no se construirá, tendrá una ip base 10.0.0.0 con mascara 8 y esperara un tiempo antes de que los switch conecten a los controladores de la red y por ultimo las direcciones MAC de los hosts se generara de forma ordenada.

3.5.2.1.3 Métodos

3.5.2.1.3.1 Añadir Host

```
def mininet.net.Mininet.addHost (self,
                                 name,
                                 cls = None,
                                 ip,
                                 mac,
                                 cores,
                                 defaultRoute
      )
```

Retorna: Node

Descripción: Método utilizado para añadir un host a la red.

Parámetros	Tipo	Opcional	Descripción
Name	String	NO	Nombre e identificador del host.
Cls	Node	SI	Atributo que indica si el host recibirá un tipo de host personalizado (HostWithPrivateDirs host con directorios privador o CPULimitedHost host con limitaciones en su CPU).
Ip	String	SI	Atributo que indica la ip y la máscara del host ("10.0.0.1/5").
Mac	String	SI	Atributo que indica la dirección MAC del host ("00:00:00:00:00:01").
Cores	Int	SI	Atributo que indica la cantidad de núcleos reales que le serán asignados a un host.
defaulRoute	String	SI	Atributo que indica la dirección ip de un router por defecto ("192.168.0.3").

Tabla 3. Descripción de parámetros método Añadir Host

Ejemplo:

```
h1 = net.addHost( 'h1', cls = CPULimitedHost, ip = '10.0.0.1/8' )
```

El siguiente ejemplo muestra como añadir un host a la red, en donde almacena en el objeto tipo Node h1 un host que tendrá como nombre e identificador h1, además indica que tendrá limitaciones en su CPU que podrán ser modificadas posteriormente, se le asigna una ip 10.0.0.1 y una máscara 255.0.0.0 (/8).

Observaciones

Los métodos opcionales si no son agregados, "Mininet" automáticamente ingresara valores por defecto.

3.5.2.1.3.2 Conseguir Ítem

```
def mininet.net.Mininet.__getitem__  ( self,
                                       key
    )
```

Retorna: **Node**

Descripción: Método que retorna un nodo dando por parámetro su identificador.

Parámetros	Tipo	Opcional	Descripción
Key	String	NO	Atributo que indica el identificador del nodo para ser recuperado.

Tabla 4. Descripción de parámetros método Conseguir Item

Ejemplo

```
haux = net.__getitem__( 'h1' )
print haux.name
```

El siguiente ejemplo almacena en una variable el host h1 conseguido con el método __getitem__ y luego imprimiría el nombre del host.

3.5.2.1.3.3 Tamaño

```
def mininet.net.Mininet.__len__()
```

Retorna: Int

Descripción: Método que retorna el número de nodos presentes en una red ya creada

Ejemplo

```
print net.__len__()
```

Asumiendo que la red ya esta creada, el siguiente ejemplo imprimirá la cantidad de nodos (hosts, switches y controladores) en la red.

3.5.2.1.3.4 Añadir Switch

```
def mininet.net.Mininet.addSwitch  ( self,
                                     name,
                                     cls = None,
                                     listenPort,
                                     inNamespace
)
```

Retorna: Switch

Descripción: Método utilizado para añadir un switch a la red.

Parámetros	Tipo	Opcional	Descripción
Name	String	NO	Nombre e identificador del Switch.
Cls	Switch	SI	Atributo que indica si el switch recibirá un tipo de switch personalizado como UserSwitch, OVSSwitch, IVSSwitch ().
listenPort	Int	SI	Atributo que indica el puerto de escucha OpenFlow que tendrá el switch.
inNamespace	Booleano	SI	Atributo que indica si el Switch estará o no dentro de un espacio de nombre de red

Tabla 5. Descripción de parámetros método Añadir Switch

Ejemplo:

```
s0 = net.addSwitch ( 's0' ,cls = OVSKernelSwitch)
```

El siguiente ejemplo añade un switch a la red con nombre e identificado s0 e indica que será un switch del tipo OVSKernelSwitch.

3.5.2.1.3.5 Lista de nodos

```
def keys ( self )
```

Retorna: Lista

Descripción: Método que retorna la lista de nodos actuales en la red.

Ejemplo

```
a = net.keys()
print a
```

Suponiendo que la red ya este creada (por ejemplo 2 host conectados a un switch y este switch a un controlador). El siguiente ejemplo retornaría una lista con todos los identificadores a la variable a, el resultado sería el siguiente ['h1', 'h2', 's0', 'c0']

3.5.2.1.3.6 Añadir Link

```
def mininet.net.Mininet.addLink( self,
                                 node1,
                                 node2,
                                 port1 = None,
                                 port2 = None,
                                 cls = None,
                                 addr1,
                                 addr2,
                                 )
```

Retorna: Link

Descripción: Método que añade un link a la red.

Parámetros	Tipo	Opcional	Descripción
node1	Node	NO	Indica el nodo con que se realizara el link.
node2	Node	NO	Indica el nodo con que se realizara el link.
port1	Int	SI	Indica el puerto de conexión de la interfaz de red del primer nodo pasado por parámetro.
port2	Int	SI	Indica el puerto de conexión de la interfaz de red del segundo nodo pasado por parámetro.
Cls	Link	SI	Atributo que indica si será un tipo de link especial como TCLink que permite que el enlace sea personalizado (Ancho de banda, Jitter, porcentaje de perdida entre otros)
addr1	String	SI	Indica la dirección MAC del host
addr2	String	SI	Indica la dirección MAC del switch

Tabla 6. Descripción parámetros Añadir Link

Ejemplo

```
net.addLink(h1, s0,port1 =7000,port2 =7000,addr1 ='00:00:00:00:00:01',
addr2 ='00:00:00:00:00:02')
```

El siguiente ejemplo muestra como añadir un link entre dos nodos (suponiendo que el host h1 y el switch s0 estén ya declarados) en donde indica que el host h1 se comunicara al switch por medio del puerto 7000 y viceversa, además indica que la dirección MAC del host h1 será '00:00:00:00:00:01' y la dirección MAC del enlace con el host en el switch será de '00:00:00:00:00:02'.

Observación

Si la dirección MAC es agregada al momento de configurar el host esta no será modificada por el parámetro addr1 de addLink.

3.5.2.1.3.7 Esperar Conexión de Switchs

```
def mininet.net.Mininet.waitConnected ( self,
                                        timeout = None,
                                        delay = .5 )
```

Retorna: Booleano

Descripción: Método utilizado para esperar que cada switch se conecte a un controlador. Retorna verdadero si todos los switch han sido conectados exitosamente.

Parámetros	Tipo	Opcional	Descripción
Timeout	Int	SI	Atributo que indica el tiempo de espera del método, si se define en None el tiempo será indefinido.
Delay	Float	SI	Atributo que indica el tiempo de espera que tendrá cada iteración

Tabla 7. Descripción Atributos Esperar Conexión de Switchs

Ejemplo

```
if net.waitConnected( timeout = 2000 )
     print 'conexión efectuada correctamente
```

El siguiente ejemplo indica que si todos los Switchs estan correctamente conectados almenos a un controlador, se imprimira el mensaje que la conexión ha sido efectuada exitosamente.

3.5.2.1.3.8 Añadir Host con NAT

```
def mininet.net.Mininet.addNAT      ( self,
                                      name = 'nat0',
                                      connect = True,
                                      inNamespace = False,
                                      cls = None,
                                      ip,
                                      mac,
                                      cores,
                                      defaultRoute
```

Retorna: Host

Descripción: El siguiente método añade un host pero con NAT, es decir un host capaz de realizar traducción de direcciones de red y tener conexión a internet.

Parámetros	Tipo	Opcional	Descripción
Name	Node	SI	Atributo que indica el nombre e identificador del host
Connect	Booleano	SI	Atributo que indica si el host estará conectado o no a internet
inNamespace	Booleano	SI	Atributo que indica si el host estará o no dentro de un espacio de nombres
Ip	Int	SI	Atributo que indica la dirección de la puerta de enlace predeterminada del host
Cls	Node	SI	Atributo que indica si el host recibirá un tipo de host personalizado (HostWithPrivateDirs host con directorios privador o CPULimitedHost host con limitaciones en su CPU).
Mac	String	SI	Atributo que indica la dirección MAC del host (“00:00:00:00:00:01”).
Cores	Int	SI	Atributo que indica la cantidad de núcleos reales que le serán asignados a un host.
defaulRoute	String	SI	Atributo que indica la dirección ip de un router por defecto (“192.168.0.3”).

Tabla 8. Descripción Parámetros Agregar Host con NAT

Ejemplo

```
hnat = net.addNat(name = ’nat1’ )
```

El siguiente ejemplo añade un host con acceso a internet con nombre nat1.

Observaciones

Mininet aun esta en pleno desarrollo, por este motivo este método aun posee ciertos problemas y no es recomendable su uso. No obstante, si estubiera presente en los script de los usuario es recomendable agregar por parámetro el nombre (a pesar de que sea opcional) para no tener conflicto de identificadores, de igual forma es recomendable que el atributo cls no contenga una clase personalizada de host.

3.5.2.1.3.9 Conseguir Nodo por Nombre

```
def mininet.net.Mininet.getNodeByName( self, *args )
```

Retorna: Node o una Lista de Node

Descripción: Método que retorna uno o mas nodos pasando por parámetros sus nombres.

Parámetros	Tipo	Opcional	Descripción
***args**	String	NO	Atributo que indica el nombre o los nombres e identificadores de los host.

Tabla 9. Descripción parámetros Conseguir Nodo por Nombre

Ejemplo

```
a = net.getNodeByName( 'h1' , 'h2' )
b = net.getNodeByName( 'h2' )
```

El siguiente ejemplo muestra cómo conseguir dos host, el primero retorna una lista de Node que se almacena en a, mientras tanto el segundo retorna un Node que se almacenara en b

3.5.2.1.3.10 Conseguir Nodo por Nombre II

```
def mininet.net.Mininet.get( self, *args )
```

Retorna: Node o una Lista de Node

Descripción: Método que retorna uno o mas nodos pasando por parámetros sus nombres.

Parámetros	Tipo	Opcional	Descripción
***args**	String	NO	Atributo que indica el nombre o los nombres e identificadores de los host.

Tabla 10. Descripción parámetros Conseguir Nodo por Nombre II

Ejemplo

```
a = net.get( 'h1' , 'h2' )
b = net.get( 'h2' )
```

El siguiente ejemplo muestra cómo conseguir dos host, el primero retorna una lista de Node que se almacena en a, mientras tanto el segundo retorna un Node que se almacenara en b.

3.5.2.1.3.11 Nodo Contenido

```
def mininet.net.Mininet.__contains__ ( self, item )
```

Retorna: Booleano

Descripción: Método que retorna verdadero o falso si es que el nodo se encuentra en la red, pasando por parámetro su nombre.

Parámetros	Tipo	Opcional	Descripción
Item	String	NO	Atributo que indica el nombre e identificador del host

Tabla 11. Descripción parámetros Conseguir Nodo por Nombre II

Ejemplo

```
a = net.__contains__( 'h1' )
print a
```

El siguiente ejemplo almacena un valor Booleano en la variable a e indica si el nodo h1 esta contenido o no en la red.

3.5.2.1.3.12 Lista de Claves

```
def mininet.net.Mininet keys ( self )
```

Retorna: Lista de String

Descripción: Método que retorna una lista de String que contiene todas las claves de los nodos presentes en la red

Ejemplo

```
a = net.keys()
print a
```

El siguiente ejemplo almacena una lista de String con todas las claves de los nodos presentes en la red, suponiendo que hubiera una red con 2 host un switch y un controlador el resultado sería el siguiente ['h1', 'h2', 's0', 'c0']

3.5.2.1.3.13 Lista de Claves II

```
def mininet.net.Mininet.values ( self )
```

Retorna: Lista de String

Descripción: Método que retorna una lista de String que contiene todas las claves de los nodos presentes en la red

Ejemplo

```
a = net.values()
print a
```

El siguiente ejemplo almacena una lista de String con todas las claves de los nodos presentes en la red, suponiendo que hubiera una red con 2 host un switch y un controlador el resultado sería el siguiente ['h1', 'h2', 's0', 'c0']

3.5.2.1.3.14 Información Nodos

```
def mininet.net.Mininet. items ( self )
```

Retorna: Tupla Clave Valor(String, String)

Descripción: Método que retorna una tupla clave valor con el identificador del host y en valor presenta el tipo de nodo, nombre, interfaz, ip y el número del proceso que ocupa.

Ejemplo

```
a = net.items()
print a
```

Suponiendo que hubiera una red con 2 host un switch y un controlador el resultado sería el siguiente: [('h1', <Host h1: h1-eth7000:10.0.0.1 pid=19716>), ('h2', <Host h2: h2-eth0:10.0.0.2 pid=19721>), ('s0', <OVSSwitch s0: lo:127.0.0.1,s0-eth8000:None,s0-eth8001:None pid=19711>), ('c0', <Controller c0: 127.0.0.1:6633 pid=19702>)]

3.5.2.1.3.15 Iniciar Terminales

```
def mininet.net.Mininet.startTerms ( self )
```

Retorna: Void

Descripción: Método que inicia terminales Xterms en cada uno de los nodos de la red.

Ejemplo

```
net.startTerms()
```

Suponiendo que hubiera una red con 2 host un switch y un controlador el script iniciaría un total de 4 terminales Xterm que simularan a cada uno de los nodos virtuales

3.5.2.1.3.16 Cerrar Terminales

```
def mininet.net.Mininet.stopTerms ( self )
```

Retorna: Void

Descripción: Método que detiene todas las terminales si es que se han iniciado.

Ejemplo:

```
net.stopXterms()
```

Suponiendo que hubiera una red con 2 host un switch y un controlador y hubiera terminales Xterm iniciadas el script cerraría automáticamente todas las terminales.

3.5.2.1.3.17 Static Arp

```
def mininet.net.Mininet.staticArp ( self )
```

Retorna: Void

Descripción: Método que inicia las tablas ARP en cada uno de los host.

Ejemplo:

```
net.staticArp()
```

Suponiendo que hubiera una red con 2 host un switch, un controlador y la red ya esta construida se iniciaran tablas ARP dentro de cada uno de los host.

3.5.2.1.3.18 Iniciar Controladores y Switches

```
def mininet.net.Mininet.start ( self )
```

Retorna: Void

Descripción: Método que inicia todo los controladores y switches de la red

Ejemplo:

```
net.start()
```

Suponiendo que la red ya esta creada y los switch y controladores aun no se han iniciado, este método los iniciara.

3.5.2.1.3.19 Detener Red

```
def mininet.net.Mininet.stop( self )
```

Retorna: Void

Descripción: Método que detiene host, switchs y controladores en la red. (Este método es el final para todo script en Mininet).

Ejemplo:

#Final del código de red en Mininet

```
net.stop()
```

Este ejemplo indica que el método stop detendrá a todos los nodos existentes en la red.

3.5.2.1.3.20 Ping

```
def mininet.net.Mininet.ping( self, hosts = None , timeout = None )
```

Retorna: String

Descripción: Método utilizado para hacer ping entre los primeros host o bien entre los host indicados por medio de una lista.

Parámetros	Tipo	Opcional	Descripción
hosts	Lista	SI	Atributo que indica los host que se involucraran en el ping. En caso de pasar este atributo al método se hará el ping entre los 2 primeros hosts.
timeout	String	SI	Atributo que indica cuando tiempo (mili segundos) esperara para que se entregue una respuesta.

Tabla 12. Descripción Parámetros Ping

Ejemplo

```
a = net.get( 'h1' , 'h2' )
print net.ping(hosts = a)
```

El siguiente ejemplo muestra como realizar un ping entre dos host por medio del método ping de la clase "Mininet", como bien se observa es necesario conseguir antes la lista de nodos a los que se realizara el ping antes de utilizar el método si es que se quiere enviar por parámetro los host.

El resultado sería como este:

*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)

3.5.2.1.3.21 Parse Ping

```
def mininet.net.Mininet._parsePing ( pingOutput )
```

Retorna: String y String

Descripción: Método estático no utilizado directamente en los script creados por el usuario pero si utilizado por el método ping de la clase "Mininet" para acortar el resultado de un comando ping.

Parámetros	Tipo	Opcional	Descripción
pingOutput	String	NO	Atributo que indica el resultado de un ping entre 2 host.

Tabla 13. Descripción Atributos Parse Ping

3.5.2.1.3.22 Parse Ping Full

```
def mininet.net.Mininet._parsePingFull ( pingOutput )
```

Retorna: String y String

Descripción: Método estático no utilizado directamente en los script creados por el usuario pero si utilizado por el método pingFull de la clase "Mininet" para acortar el resultado de un comando ping.

Parámetros	Tipo	Opcional	Descripción
pingOutput	String	NO	Atributo que indica el resultado de un ping entre 2 host.

Tabla 14. Descripción Parámetros Parse Ping Full

3.5.2.1.3.23 Ping Full

```
def mininet.net.Mininet.pingFull( self, hosts = None , timeout = None )
```

Retorna: String

Descripción: Método utilizada para hacer ping entre los primeros host o bien entre los host indicados por medio de una lista.

Parámetros	Tipo	Opcional	Descripción
hosts	Lista	SI	Atributo que indica los host que se involucrara en el ping. En caso de pasar este atributo al método se hará el ping entre los 2 primeros hosts
timeout	String	SI	Atributo que indica cuanto tiempo (mili segundos) esperara para que se entregue una respuesta.

Tabla 15. Descripción parámetros Ping Full

Ejemplo

```
a = net.get( 'h1' , 'h2' )
print net.pingFull(hosts=a)
```

El siguiente ejemplo muestra como realizar un ping entre dos host por medio del método pingFull de la clase "Mininet", como bien se observa es necesario conseguir antes la lista de nodos a los que se realizara el ping antes de utilizar el método si es que se quiere enviar por parámetro los host.

El resultado sería como este:

*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results:
h1->h2: 1/1, rtt min/avg/max/mdev 0.334/0.334/0.334/0.000 ms
h2->h1: 1/1, rtt min/avg/max/mdev 0.365/0.365/0.365/0.000 ms

Observación

A diferencia del método ping, este además de entregar el resultado debidamente parseado, entrega el tiempo de ejecución en tiempo real de cada uno de los ping.

3.5.2.1.3.24 Ping All

```
def mininet.net.Mininet.pingAll( timeout = None )
```

Retorna: String

Descripción: Método utilizada para hacer ping entre todos los host de la red, los resultados serán entregados con el formato de entrega del método ping de la clase Mininet.

Parámetros	Tipo	Opcional	Descripción
timeout	String	SI	Atributo que indica cuanto tiempo (mili segundos) esperara para que se entregue una respuesta.

Tabla 16. Descripción Parámetros Ping All

Ejemplo

```
print net.pingAll(timeout = '2000')
```

El siguiente ejemplo muestra cómo realizar un ping entre todos los host de la red. Suponiendo que hubieran 3 host en la red el resultado sería como este:

*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)

3.5.2.1.3.25 Ping Par

```
def mininet.net.Mininet.pingPair( self )
```

Retorna: Void

Descripción: Método utilizado para hacer ping entre los dos primeros host de la red

Ejemplo

```
print net.pingPair()
```

El siguiente ejemplo muestra cómo realizar un ping entre los 2 primeros host de la red el resultado sería como este:
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)

3.5.2.1.3.26 Ping Par Full

```
def mininet.net.Mininet.pingPairFull( self )
```

Retorna: Void

Descripción: Método utilizado para hacer ping entre los dos primeros host de la red.

Ejemplo

```
print net.pingPairFull()
```

El siguiente ejemplo muestra cómo realizar un ping entre los 2 primeros host de la red el resultado sería como este:

```
h1 -> h2
h2 -> h1
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 1003.802/1003.802/1003.802/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 2.605/2.605/2.605/0.000 ms
```

Observación

Nótese que el formato de entrega del ping es como pingFull() esto es debido a que es utilizado el parseado de este método para entregar el resultado.

3.5.2.1.3.27 Parse Iperf

```
def mininet.net.Mininet.parseIperf ( pingOutput )
```

Retorna: String

Descripción: Método estático no utilizado directamente en los script creados por el usuario, pero si utilizado por el método pingFull de la clase Mininet para acortar el resultado de un comando iperf.

Parámetros	Tipo	Opcional	Descripción
pingOutput	String	NO	Atributo que indica el resultado de un Iperf entre 2 host.

3.5.2.1.3.28 Ping All Full

```
def mininet.net.Mininet.pingAllFull( self )
```

Retorna: Void

Descripción: Método utilizado para hacer ping entre todos los host de la red.

Ejemplo

```
print net.pingALLFull()
```

El siguiente ejemplo muestra cómo realizar un ping entre todos los host de la red. Suponiendo que hubieran 3 host creados el resultado sería veria como este:

*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results:
h1->h2: 1/1, rtt min/avg/max/mdev 5.886/5.886/5.886/0.000 ms
h1->h3: 1/1, rtt min/avg/max/mdev 4.012/4.012/4.012/0.000 ms
h2->h1: 1/1, rtt min/avg/max/mdev 1.990/1.990/1.990/0.000 ms
h2->h3: 1/1, rtt min/avg/max/mdev 3.485/3.485/3.485/0.000 ms
h3->h1: 1/1, rtt min/avg/max/mdev 1.954/1.954/1.954/0.000 ms
h3->h2: 1/1, rtt min/avg/max/mdev 1.934/1.934/1.934/0.000 ms

Observación

Nótese que el formato de entrega del ping es como pingFull(), esto es debido a que el metodo utiliza el mismo metodo de parseado que pingFulll().

3.5.2.1.3.29 Iperf

```
def mininet.net.Mininet.iperf ( self,
                                hosts = None,
                                l4Type = 'TCP',
                                udpBw = '10M',
                                format = None,
      )
```

Retorna: String

Descripción: Método utilizado para realizar una prueba iperf entre dos host.

Parámetros	Tipo	Opcional	Descripción
hosts	Lista de Host	SI	Atributo que indica los host que serán ingresados a la prueba iperf.
l4Type	String	SI	Atributo que inda el tipo de prueba que se realizara, por defecto es TCP pero también puede ser UDP.
udpBw	String	SI	Atributo que indica el tamaño de ancho de banda de la prueba UDP (en el caso que l4Type sea UDP).
format	String	SI	Atributo que indica el formato en que se realizara la prueba por defecto esta en Mega Bytes (tambin puede ser Kilo Bits, Mega Bits, Kilo Bytes y Mega Bytes)

Tabla 17. Descripción parámetros Iperf

Ejemplo

 # 3 host ingresados h1 , h2 y h3

```
print net.iperf()
print net.iperf((h1,h2))
```

El siguiente ejemplo muestra un test de ancho de banda entre 2 host, en el caso de no pasar parámetros el método hará la prueba entre los host extremos (h1 y h3). El resultado se vería como esto:

*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['7.42 Gbits/sec', '7.43 Gbits/sec']
['7.42 Gbits/sec', '7.43 Gbits/sec']

*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['7.08 Gbits/sec', '7.09 Gbits/sec']
['7.08 Gbits/sec', '7.09 Gbits/sec']

3.5.2.1.3.30 Test Límite de CPU

```
def mininet.net.Mininet.runCpuLimitTest( self, cpu, duration = 5 )
```

Retorna: Lista de Floats

Descripción: Método utilizado para realizar pruebas de límites en la CPU de los host.

Parámetros	Tipo	Opcional	Descripción
cpu	Float	NO	Atributo que indica la fracción de cpu con que serán probados los host
duration	Int	SI	Atributo que indica la duración en segundos que tendrá la prueba

Tabla 18. Descripción Parámetros Test Límite de CPU

Ejemplo

```
print net.runCpuLimitTest( cpu = .3 )
```

El siguiente ejemplo indica que a todos los host se le hara un test de CPU, utilizando un 30% en cada uno de ellos. El resultado se veria como esto.

Results: [23.890075, 24.38935385, 24.4470097, 24.735528600000002, 24.63887425, 23.9017149, 24.16578115, 24.78575595, 24.5121782, 24.6731175, 42.286314000000004, 40.40795705, 33.8532268, 36.753131350000004, 37.7672988]

Observación

Para realizar esta prueba todo los host deben ser del tipo CPULimitedHost.

3.5.2.1.3.31 Configurar estado del Link

```
def mininet.net.defconfigLinkStatus( self, src, dst, status )
```

Retorna: Void

Descripción: Método utilizado para activar y desactivar la conexión de un link entre dos nodos.

Parámetros	Tipo	Opcional	Descripción
Src	String	NO	Atributo que indica el identificador origen del link.
Dst	String	NO	Atributo que indica el identificador destino del link .
Status	String	NO	Atributo que indica si el nodo estará activo o inactivo (up or down).

Tabla 19. Descripción Parámetros Configurar estado del Link

Ejemplo

```
print net.defconfigLinkStatus( 'h1' , 's0', 'down' )
```

El siguiente ejemplo indica que el link entre el host h1 y el switch s0 se desactivara.

3.5.2.1.3.32 Construir a partir de Topología

```
def mininet.net.buildFromTopo ( self, topo = None )
```

Retorna: Void

Descripción: Método utilizado para crear una red a partir de una topología predefinida, el método construirá e iniciara los switch, hosts, enlaces y controladores de la red.

Parámetros	Tipo	Opcional	Descripción
Topo	Topo	Si	Atributo que indica el tipo de topología a introducir (LinearTopo, SingleSwitchReversedTopo y SingleSwitchTopo).

Tabla 20. Descripción Parámetros Configurar estado del Link

Ejemplo

```
net  = Mininet(topo = None,ipBase ='10.0.0.0/8')
net.buildFromTopo(topo = LinearTopo(4) )
```

El siguiente ejemplo indica cómo crear una red a partir del método net.buildFromTopo, en donde el atributo topo indica que tendrá una topología lineal de 4 (4 host conectados cada uno a su respectivo switch y cada switch conectado a un controlador)

Observación

El atributo topo también pudo haber sido entregado como parámetro al declarar el constructor Mininet, sin necesidad de llamar al método buildFromTopo.

```
net  = Mininet (topo = LinearTopo(4),build=True, ipBase = '10.0.0.0/8')
```

3.5.2.1.3.33 Construir

```
def mininet.net.build ( self )
```

Retorna: Void

Descripción: Método utilizado por el constructor de Mininet para construir la red cuando esta es pasada por parámetro.

3.5.2.1.3.34 Configurar

```
def mininet.net.configHosts ( self )
```

Retorna: Void

Descripción: Método utilizado por el constructor de Mininet para configurar los host cuando se construyen a partir de una topología

3.5.2.1.3.35 Iterador

```
def mininet.net.__iter__( self )
```

Retorna: Iterador

Descripción: Método utilizado internamente por Mininet, retorna un iterador sobre los nodos según su nombre.

3.5.2.1.3.36 Inicializar

```
def mininet.net.init( self )
```

Retorna: Void

Descripción: Método utilizado internamente por el constructor de Mininet para inicializar y asegurar que el script este corriendo como root.

3.5.2.1.3.37 Interactuar

```
def mininet.net.interact( self )
```

Retorna: Void

Descripción: Método utilizado para iniciar la red y correr la consola de comandos CLI en Mininet.

Ejemplo:

```
net.iteract()
```

El siguiente ejemplo indica que después de haber creado la red (Agregación de nodos, links etc) esta inicia y se lanza dentro de ella la consola de comandos CLI para poder interactuar con los nodos de la red.

3.5.3 Archivo node.py

3.5.3.1 Clase Node

3.5.3.1.1 Diagrama de Herencia

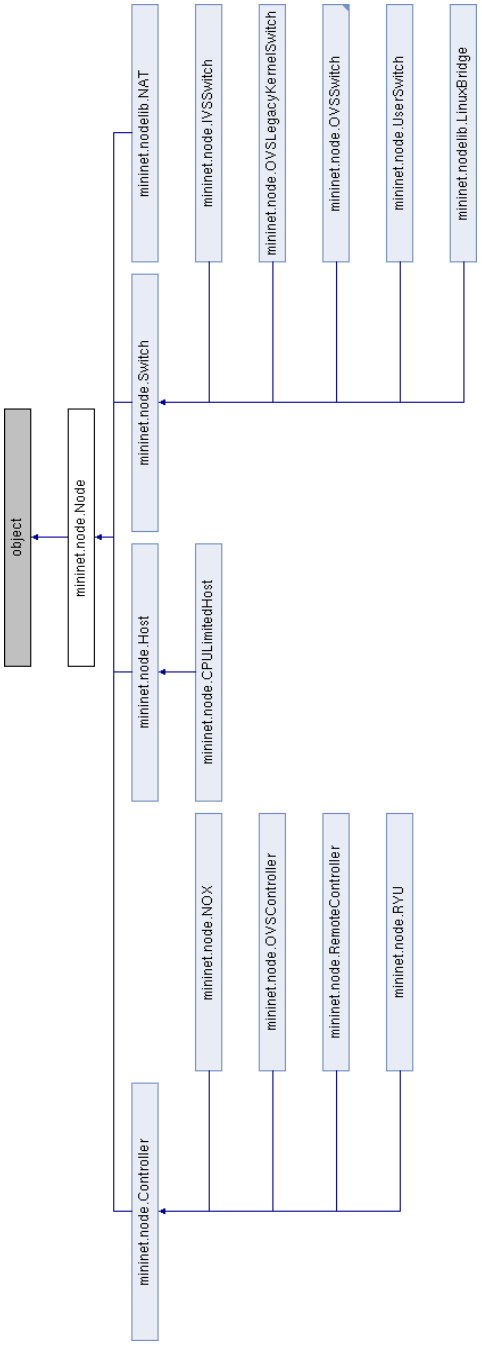


Figura 6. Diagrama de Herencia Clase Node

3.5.3.1.2 Constructor

```
def mininet.node.Node.__init__( self,
                                name,
                                inNamespace = True,
                               mac = None,
                                ip = None,
                                defaultRoute = None,
                                lo = 'up',
      )
```

Descripción: Constructor base para la creación de un host, switch o controlador.

Parámetros	Tipo	Opcional	Descripción
name	String	NO	Nombre e identificador del nodo.
ip	String	SI	Atributo que indica la ip y la máscara del nodo ("10.0.0.1/5").
mac	String	SI	Atributo que indica la dirección MAC del nodo ("00:00:00:00:00:01").
defaulRoute	String	SI	Atributo que indica la dirección ip de la ruta por defecto.
lo	String	SI	Atributo que indica si el nodo estará o no disponible.

Figura 7. Descripción Parámetros Constructor Node

3.5.3.1.3 Métodos

3.5.3.1.3.1 Conseguir Nodo Archivo Descriptor

```
def mininet.node.Node.fdToNode( cls, fd )
```

Retorna: Node

Descripción: Método de clase que retorna un Node pasando por parámetro un descriptor de archivos

Parámetros	Tipo	Opcional	Descripción
fd	File Descriptor	NO	Atributo que indica un archivo descriptor

Figura 8. Descripción parámetros Conseguir Nodo Archivo Descriptor

3.5.3.1.3.2 Iniciar Consola

```
def mininet.node.Node.startShell ( cls , mnopts = None )
```

Retorna: Node

Descripción: Método que inicia una consola de procesos sobre los nodos. Este método no es utilizado por el usuario, sino utilizado por el constructor de la clase Node para iniciar las Shell dentro de cada nodo virtual.

Parámetros	Tipo	Opcional	Descripción
mnopts	String	SI	Atributo que indica las opciones de comando que tendrá la consola de procesos.

Figura 9. Descripción Parámetros Start Shell

3.5.3.1.3.3 Limpiar Node

```
def mininet.node.Node.cleanup ( self)
```

Retorna: Void

Descripción: Método interno de Python que ayuda a recolectar memoria basura.

3.5.3.1.3.4 Escribir Data

```
def mininet.node.Node.write ( self, data)
```

Retorna: Void

Descripción: Método de uso interno utilizado para escribir información en los nodos

Parámetros	Tipo	Opcional	Descripción
Data	String	SI	Atributo que indica que se escribirá dentro del nodo.

Tabla 21. Descripción Parámetros Escribir Data

3.5.3.1.3.5 Cmd

```
def mininet.node.Node.cmd( self, **args, **kwargs)
```

Retorna: String

Descripción: Método utilizado para ejecutar comandos de consola sobre los nodos.

Parámetros	Tipo	Opcional	Descripción
****args**	String	NO	Atributo que indica el comando a ejecutar en el nodo.
****kwargs**	String	SI	Atributo que indica el complemento del comando a ejecutar en el nodo.

Tabla 22. Descripción Parámetros Cmd

Ejemplo

```
print h1.cmd('ping -c 3 10.0.0.2')
print h2.cmd('ping' , '-c', ' 3', ' 10.0.0.1')
```

El siguiente ejemplo muestra 2 hosts haciendo ping uno al otro. Cabe ver que es posible mandar de una sola vez el String junto con el comando, o bien ingresar los parametros de forma separada como en el host h2.

3.5.3.1.3.6 Enviar Cmd

```
def mininet.node.Node.sendCmd( self,**args, **kwargs)
```

Retorna: Void

Descripción: Método interno utilizado ejecutar comandos de consola sobre los nodos (método interno utilizado por el método cmd ()).

Parámetros	Tipo	Opcional	Descripción
****args**	String	NO	Atributo que indica el comando a ejecutar en el nodo.
****kwargs**	String	SI	Atributo que indica el complemento del comando a ejecutar en el nodo.

Tabla 23. Descripción Parámetros Enviar Cmd

3.5.3.1.3.7 Leer

```
def mininet.node.Node.read( self, maxbytes = 1024)
```

Retorna: Void

Descripción: Método interno utilizado para leer el buffer de un nodo.

Parámetros	Tipo	Opcional	Descripción
maxbytes	Int	SI	Atributo que indica el número máximo de bytes a retornar.

Tabla 24. Descripción Parámetros Leer

3.5.3.1.3.8 Leer Línea

```
def mininet.node.Node.readline( self)
```

Retorna: String

Descripción: Método interno utilizado para leer una línea del buffer del nodo.

3.5.3.1.3.9 Terminar

```
def mininet.node.Node.terminate( self)
```

Retorna: Void

Descripción: Método interno utilizado para destruir un nodo y luego recolectar su basura.

3.5.3.1.3.10 Detener

```
def mininet.node.Node.stop( self)
```

Retorna: Void

Descripción: Método interno que utiliza a al metodo terminate() para destruir un nodo y luego recolectar su basura.

3.5.3.1.3.11 Esperar Lectura

```
def mininet.node.Node.waitReadable( self, timeoutms = None )
```

Retorna: Void

Descripción: Método interno utilizado para esperar la salida de un nodo.

Parámetros	Tipo	Opcional	Descripción
timeoutms	Int	SI	Atributo que indica el tiempo de espera en milisegundos.

Tabla 25. Descripción Parámetros Esperar Lectura

3.5.3.1.3.12 Enviar Interrupción

```
def mininet.node.Node.sendInt ( self, intr = chr( 3 ) )
```

Retorna: Void

Descripción: Método interno utilizado para esperar la salida de un nodo

Parámetros	Tipo	Opcional	Descripción
Intr	String	SI	Atributo que indica por defecto el final de un carácter.

Tabla 26. Descripción Parámetros Enviar Interrupción

3.5.3.1.3.13 Monitor

```
def mininet.node.Node.monitor (self, timeoutms = None, findPid = True )
```

Retorna: Void

Descripción: Método interno utilizado para esperar la salida de un nodo.

Parámetros	Tipo	Opcional	Descripción
timeoutms	Int	SI	Atributo que indica el tiempo de espera del monitor, si es None esperara indefinidamente.
findPid	Booleano	SI	Atributo que indica si el metodo realiza o no la búsqueda de los procesos.

Tabla 27. Descripción Parámetros Monitor

3.5.3.1.3.14 Esperar Salida

```
def mininet.node.Node.waitOutput (self, verbose = False,
                                  findPid = True )
```

Retorna: String

Descripción: Método interno utilizado para esperar que un comando se complete.

Parámetros	Tipo	Opcional	Descripción
verbose	Booleano	SI	Atributo que indica si la salida será o no interactiva.
findPid	Booleano	SI	Atributo que indica si el metodo realiza o no la búsqueda de los procesos.

Tabla 28. Descripción Parámetros Esperar Salida

3.5.3.1.3.15 Nuevo Puerto

```
def mininet.node.Node.newPort ( self )
```

Retorna: Int

Descripción: Método interno utilizado para retornar el próximo puerto a asignar.

3.5.3.1.3.16 Añadir Interface

```
def mininet.node.Node.addIntf( self,
                               intf,
                               port = None,
                               moveIntfFn = moveIntf )
```

Retorna: Intf

Descripción: Método interno utilizado para añadir interfaces a la red.

Parámetros	Tipo	Opcional	Descripción
intf	Intf	NO	Interfaz de red.
port	Int	SI	Atributo que indica el puerto de comunicación de la interfaz.
moveIntfFn	Function()	SI	Función que indica si la interfaz se moverá a otro nodo.

Tabla 29. Descripción Parámetros Añadir Interface

3.5.3.1.3.17 Interface Default

```
def mininet.node.Node.defaultIntf ( self )
```

Retorna: Intf

Descripción: Método utilizado para retornar una interfaz para el puerto mas bajo de un nodo.

Observación

En caso que no existiera una interface, se mostrara una advertencia en la ejecución indicando que el nodo no posee interfaces.

3.5.3.1.3.18 Conseguir Interface

```
def mininet.node.Node.intf ( self, intf = '' )
```

Retorna: Intf

Descripción: Método utilizado para retornar una interface pasando por parámetro el nombre de la interfas,

Parámetros	Tipo	Opcional	Descripción
intf	String	SI	Atributo que indica el nombre de la interfaz (ejemplo: h1-eth0)

Tabla 30. Descripción Parámetros Conseguir Interface

Ejemplo

 # switch s0 , host h1

```
intf1 = s0.intf( intf  = 's0-eth1')
```

El siguiente ejemplo muestra cómo conseguir la interfaz del switch conectada al host h1 y almacenarla en la variable intf1.

Observación

En caso de no pasar por parámetro el nombre de la interface se retornara la interface existente para el nodo. Este método es muy utilizado para conseguir las interfaces de los Swichts.

3.5.3.1.3.19 Conectados a

```
def mininet.node.Node.connectionsTo ( self, node)
```

Retorna: Tupla

Descripción: Método utilizado para conseguir las interfaces de dos nodos.

Parámetros	Tipo	Opcional	Descripción
node	Node	NO	Atributo que indica el nodo que buscara la interfaz.

Tabla 31. Descripción Parámetros Conectados a

Ejemplo

 # switch s0 , host h1

```
print s0.connectionsTo(h1)
```

El siguiente ejemplo muestra cómo conseguir las interfaces del switch s 0. El resultado seria de la siguiente forma:

(interfaz del switch conectado al host y la interfaz del host conectado al switch)
[(<Intf s0-eth1>, <Intf h1-eth0>)]

3.5.3.1.3.20 Borrar Interfaces

```
def mininet.node.Node.deleteIntfs ( self, checkname = True)
```

Retorna: Void

Descripción: Método utilizado para eliminar todas las interfaces creadas en un nodo.

Parámetros	Tipo	Opcional	Descripción
checkname	Booleano	SI	Atributo que indica que se borraran solo las interfaces que contienen un nombre.

Tabla 32. Descripción Borrar Interfaces

3.5.3.1.3.21 Set ARP

```
def mininet.node.Node.setARP ( self, ip, mac )
```

Retorna: String

Descripción: Método utilizado para añadir elementos a la tabla ARP dentro de un nodo.

Parámetros	Tipo	Opcional	Descripción
ip	String	NO	Atributo que indica la ip del nodo destino.
mac	String	NO	Atributo que indica la dirección MAC del nodo destino.

Tabla 33. Descripción Parámetros Set ARP

Ejemplo

 # host h1 ip:10.0.0.1 mac:00:00:00:00.00:1 y host h2 ip: 10.0.0.2 mac:00:00:00:00:00:2

```
h1.setARP('10.0.0.2', mac = '00:00:00:00:00:02')
```

El siguiente ejemplo muestra cómo añadir un elemento a la tabla ARP del host h1.

3.5.3.1.3.22 Set Host Route

```
def mininet.node.Node.setHostRoute ( self, ip, intf )
```

Retorna: String

Descripción: Método utilizado para añadir elementos a tabla de ruteo dentro de un nodo.

Parámetros	Tipo	Opcional	Descripción
ip	String	NO	Atributo que indica la ip del nodo destino.
intf	String	NO	Atributo que indica el nombre de la interfaz

Tabla 34. Descripción Parámetros Set Host Route

Ejemplo

 # host h1 ip:10.0.0.1 mac:00:00:00:00.00:1

```
h1.setHostRoute(intf = 'h1-eth0’)
```

El siguiente ejemplo muestra cómo añadir un elemento a la tabla de ruteo del host h1.

3.5.3.1.3.23 Set Default Route

```
def mininet.node.Node.setDefaultRoute ( self, intf = None )
```

Retorna: Void

Descripción: Método utilizado para añadir la ruta por defecto a una tabla de ruteo dentro de un nodo, esta se dirigirá a través de la interfaz presente en el nodo

Parámetros	Tipo	Opcional	Descripción
intf	String o Intf	NO	Atributo que indica el nombre de la interfaz o un objeto tipo Intf.

Tabla 35. Descripción Parámetros Set Default Route

Ejemplo

host h1 ip:10.0.0.1 mac:00:00:00:00.00:1

```
h1.setDefaultRoute( intf = 'h1-eth0’ )
```

El siguiente ejemplo muestra cómo añadir una ruta por defecto a la tabla de ruteo del host h1, indicando la interfaz.

3.5.3.1.3.24 Set MAC

```
def mininet.node.Node.setMAC ( self, mac , intf = None )
```

Retorna: Función

Descripción: Método utilizado para modificar la dirección MAC de un nodo.

Parámetros	Tipo	Opcional	Descripción
intf	String o Intf	SI	Atributo que indica el nombre de la interfaz o un objeto tipo Intf.
mac	String	NO	Atributo que indica la nueva dirección mac a añadir.

Tabla 36. Descripción Parámetros Set MAC

Ejemplo

host h1 ip:10.0.0.1 mac:00:00:00:00.00:1, switch s0 interfaz: s0-eth1

```
h1.setMAC(mac= ‘00:00:00:00:00:10’)
s0.setMAC(mac= ‘00:00:00:00:00:10’, intf = 's0-eth1’)
```

El siguiente ejemplo muestra cómo modificar la dirección mac de un nodo, en este caso de un host y un switch.

Observación

Si se modifica la dirección MAC de un host no es necesario pasar por parámetro el nombre de la interfaz, no obstante si se modifica un switch si es necesario pasar por parámetro el nombre de la interfaz, si la interfaz no es entregada por parámetro al switch no se modificara ninguna interfaz de este.

3.5.3.1.3.25 Set IP

```
def mininet.node.Node.setIP(self, ip, prefixLen = 8,intf = None )
```

Retorna: String

Descripción: Método utilizado para modificar la IP de un nodo.

Parámetros	Tipo	Opcional	Descripción
intf	String o Intf	SI	Atributo que indica el nombre de la interfaz o un objeto tipo Intf.
ip	String	NO	Atributo que indica la nueva dirección ip del nodo.
prefixLen	Int	SI	Atributo que indica la máscara de la IP (por defecto es /8)

Tabla 37. Descripción Parámetros Set IP

Ejemplo

host h1 ip:10.0.0.1, mascara: /8 e interfaz: h1-eth0

```
h1.setIP('10.0.0.100',9,'h1-eth0')
```

El siguiente ejemplo muestra cómo modificar la dirección ip de un nodo, en este caso de un host.

Observación

Si se modifica la dirección IP de un host no es necesario pasar por parámetro el nombre de la interfaz y la máscara, no obstante si se modifica un switch si es necesario pasar por parámetro el nombre de la interfaz, si la interfaz no es entregada por parámetro al switch no se modificara ninguna IP en sus interfaces.

3.5.3.1.3.26 Obtener IP

```
def mininet.node.Node.IP (self, intf = None )
```

Retorna: String

Descripción: Método utilizado para obtener la IP de un nodo entregando por parámetro su interfaz.

Parámetros	Tipo	Opcional	Descripción
intf	String o Intf	SI	Atributo que indica el nombre de la interfaz o un objeto tipo Intf

Tabla 38. Descripción Parámetros Obtener IP

Ejemplo

host h1 ip:10.0.0.1, mascara: /8 e interfaz: s0-eth0

```
print h1.IP( 'h1-eth0' )
print s0.IP( '10.0.0.9' , 'S0-eth1')
```

El siguiente ejemplo muestra cómo obtener la dirección ip de un nodo, en este caso de un host y un switch.

Observación

Si se obtiene la dirección IP de un host no es necesario pasar por parámetro el nombre de la interfaz, no obstante si se obtiene la dirección ip de alguna de las interfaces de un switch si es necesario pasar por parámetro el nombre de la interfaz, si la interfaz no es entregada por parámetro al switch el método no se entregara ninguna IP de sus interfaces.

3.5.3.1.3.27 Obtener MAC

```
def mininet.node.Node.MAC (self, intf = None )
```

Retorna: String

Descripción: Método utilizado para obtener la dirección MAC de un nodo, entregando opcionalmente por parámetro su interfaz.

Parámetros	Tipo	Opcional	Descripción
intf	String o Intf	SI	Atributo que indica el nombre de la interfaz.

Tabla 39. Descripción Parámetros Obtener MAC

Ejemplo

host h1 ip:10.0.0.1, mascara: /8 y switch s0 interfaz: s0-eth0

```
print h1.MAC()
print s0.MAC( 'S0-eth1' )
```

El siguiente ejemplo muestra cómo obtener la dirección MAC de un nodo en este caso de un host y un switch.

Observación

Si se obtiene la dirección MAC de un host no es necesario pasar por parámetro el nombre de la interfaz, no obstante si se obtiene la dirección MAC de alguna de las interfaces de un switch si es necesario pasar por parámetro el nombre de la interfaz, si la interfaz no es entregada por parámetro al switch no se obtendrá ninguna dirección MAC de sus interfaces.

3.5.3.1.3.28 Chequear Interfaz

```
def mininet.node.Node.intfIsUp (self, intf = None )
```

Retorna: Booleano

Descripción: Método utilizado para activar y desactivar las interfaces de un nodo.

Parámetros	Tipo	Opcional	Descripción
intf	String o Intf	SI	Atributo que indica el nombre de la interfaz

Tabla 40. Descripción Parámetros Chequear Interfaz

Observación

Este método aun esta en desarrollo y posee ciertos problemas. Debido a lo anterior el resultado de este método siempre será False a pesar de que las interfaces estén activas.

3.5.3.1.3.29 Establecer Parámetros

```
def mininet.node.Node.setParam (self, result, method, **params )
```

Retorna: Diccionario

Descripción: Método utilizado internamente por Mininet para configurar los parámetros de los nodos.

Parámetros	Tipo	Opcional	Descripción
result	Diccionario	NO	Atributo que indica el resultado que debe actualizar.
method	String	NO	Atributo que indica el nombre del método.
****params**	Variable, Lista o Diccionario	SI	Atributo que indica los parámetros opcionales que generalmente son utilizados por las subclases de la clase Node.

Tabla 41.Descripción Parámetros Establecer Parámetros

3.5.3.1.3.30 Configurar

```
def mininet.node.Node.config ( self,
                               mac = None,
                               ip = None,
                               defaultRoute = None,
                               lo = 'up',
)
```

Retorna: Diccionario

Descripción: Método utilizado internamente por Mininet para configurar los parámetros de los nodos.

Parámetros	Tipo	Opcional	Descripción
mac	String	SI	Atributo que indica la dirección MAC a establecer dentro del nodo.
ip	String	SI	Atributo que indica la dirección IP a establecer dentro del nodo.
defaultRoute	String	SI	Atributo que indica la dirección por defecto de la tabla de enrutamiento del nodo.
lo	String	SI	Atributo que indica si el nodo estará o no disponible

Tabla 42. Descripción Parámetros Configurar

Observación

Este método es utilizado en conjunto con setParam() para configurar los parámetros de los host.

3.5.3.1.3.31 Configurar por Defecto

```
def mininet.node.Node.configDefault()
```

Retorna: Void

Descripción: Método utilizado para configurar los nodos con sus atributos por defecto.

Observación

En el caso de haber establecido las direcciones automáticas de las MAC a los host en el constructor de la clase Mininet estas direcciones se mantendrán.

3.5.3.1.3.32 Lista de Interfaces Ordenadas por Puerto

```
def mininet.node.Node.intfList()
```

Retorna: Lista

Descripción: Método que retorna una lista de interfaces ordenada ascendentemente por número de puerto.

Ejemplo

switch s0 interfaz: s0-eth1, s0-eth2, lo

```
print s0.intfList()
```

El siguiente ejemplo muestra como obtener una lista de interfaces del switch s0, para luego mostrarlas por pantallas. El resultado se vería como esto:

[<Intf lo>, <Intf s0-eth1>, <Intf s0-eth2>]

3.5.3.1.3.33 Lista de Interfaces

```
def mininet.node.Node.intfNames()
```

Retorna: Lista

Descripción: Método que retorna una lista con los nombres de las interfaces de un nodo.

Ejemplo

switch s0 interfaz: s0-eth1, s0-eth2, lo

```
print s0.intfNames()
```

El siguiente ejemplo muestra cómo obtener una lista de interfaces del switch s0, para luego mostrarlas por pantalla. El resultado se vería como esto:

['lo', 's0-eth1', 's0-eth2']

3.5.3.1.3.34 Información representativa

```
def mininet.node.Node.__repr__()
```

Retorna: String

Descripción: Método que retorna una lista con información representativa de las interfaces de un nodo.

Ejemplo

switch s0 interfaz: s0-eth1, s0-eth2, lo

```
print s0.__repr__()
```

El siguiente ejemplo muestra cómo obtener información representativa de un nodo. El resultado se vería como esto:

<OVSSwitch s0: lo:127.0.0.1,s0-eth1:None,s0-eth2:None pid=6562>

3.5.3.1.3.35 Obtener nombre del Nodo

```
def mininet.node.Node.__str__()
```

Retorna: String

Descripción: Método que retorna el nombre del nodo.

Ejemplo

switch s0 interfaz: s0-eth1, s0-eth2, lo

```
print s0.__strn__()
```

El siguiente ejemplo muestra cómo obtener el nombre del switch s0. El resultado veria como esto:

s0

3.5.3.1.3.36 Verificar Instalación

```
def mininet.node.Node.checkSetup()
```

Retorna: Void

Descripción: Método de clase de uso interno en Mininet para asegurar que las subclases y súper clases han sido debidamente establecidas.

3.5.3.1.3.37 Instalación

```
def mininet.node.Node.setup()
```

Retorna: Void

Descripción: Método de clase de uso interno en Mininet para asegurar que las dependencias de la clase estén disponibles.

3.5.3.2 Clase Host

Un host simplemente es un nodo.

3.5.3.3 Diagrama de Herencia

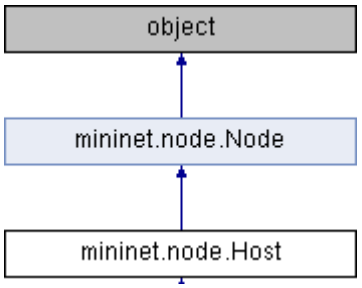


Figura 10. Diagrama de Herencia Clase Host

3.5.3.4 Clase CPULimitedHost

Host personalizado para el uso de limitaciones de CPU

3.5.3.4.1 Diagrama de Herencia

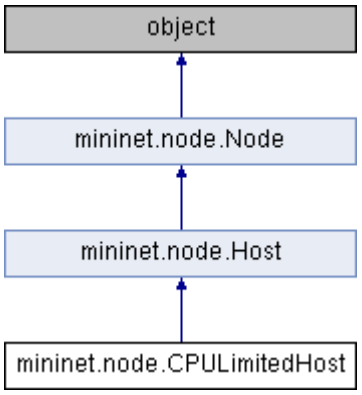


Figura 11. Diagrama de Herencia Clase CPULimitedHost

3.5.3.4.2 Constructor

```
def mininet.node.CPULimitedHost.__init__ (  self,
                                            name,
                                            sched = 'cfs'
                                            cpu = None,
                                            cores = None,
                                            ip = None,
                                            mac = None,
                                            defaultRoute = None
                                            lo = 'up'

        )
```

Descripción: Constructor de la clase CPULimitedHost utilizado para crear un host con restricciones de CPU.

Parámetros	Tipo	Opcional	Descripción
name	String	NO	Nombre e identificador del host.
sched	String	SI	Atributo que indica el tipo de administrador que tendrá la limitación de cpu (cfs, host y rt, en la versión actual es solo funcional cfs)
ip	String	SI	Atributo que indica la ip y la máscara del host ("10.0.0.1/5").
mac	String	SI	Atributo que indica la dirección MAC del host ("00:00:00:00:00:01").
cores	Int	SI	Atributo que indica la cantidad de núcleos reales que le serán asignados a un host.
defaulRoute	String	SI	Atributo que indica la dirección ip de un router por defecto ("192.168.0.3").
cpu	Float	SI	Atributo que indica el porcentaje(0.3,0.65 etc) de cpu que será asignado al host.
lo	Booleano	SI	Atributo que indica si el host estará o no disponible.

Figura 12. Descripción Parámetros Constructor CPULimitedHost

3.5.3.4.3 Métodos

3.5.3.4.3.1 Establecer Fracción de CPU

```
def mininet.node.CPULimitedHost.setCPUFrac (self, f = -1 ,sched = None )
```

Retorna: Void

Descripción: Método utilizado para establecer la fracción de CPU que tendrá el host.

Parámetros	Tipo	Opcional	Descripción
f	Float	SI	Atributo que indica la fracción de CPU que recibirá el host (0.3, 0.9 etc) .
sched	String	SI	Atributo que indica el tipo de administrador que tendrá la limitación de cpu (cfs, host y rt)

Figura 13.Descripción Parámetros Establecer Fracción de CPU

Ejemplo

Red ya creada, host h1 y h2 de tipo CPULimitedHost

```
h1.setCPUFrac(0.3)
h2.setCPUFrac(0.7)
```

El siguiente ejemplo indica que el host h1 tendrá una asignación del 30% de la CPU, mientras tanto el host h2 tendrá una asignación del 70%

Observación

Si no se entregan parámetros al método, la asignación por defecto será ilimitada y el administrador será cfs. Cabe destacar que en la versión actual de Mininet cfs es el único administrador funcional.

3.5.3.4.3.2 Establecer Cgroup

```
def mininet.node.CPULimitedHost.cgroupSet(   self,
                                             param,
                                             value,
                                             resource = 'cpu' )
```

Retorna: Int

Descripción: Método interno utilizado por setCPUfrac() para establecer fracciones de cpu a un host.

Parámetros	Tipo	Opcional	Descripción
param	String	NO	Atributo que indica el parámetro a establecer de cgset (Establecedor de parámetros de cgroup).
value	Int	NO	Atributo que indica el valor del parámetro a establecer.
resource	String	SI	Atributo que indica el recurso a modificar.

Figura 14. Descripción Parámetros Establecer Cgroup

Observación

Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [2]

3.5.3.4.3.3 Inicializar

```
def mininet.node.CPULimitedHost.init ( cls )
```

Retorna: Void

Descripción: Método de clase utilizado para inicializar la clase y asegurar que el sistema de archivos de cgroup estén montados.

Observación

Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [2].

3.5.3.4.3.4 Obtener Cgroup

```
def mininet.node.CPULimitedHost.cgroupGet(  self,
                                            param ,
                                            resource = 'cpu' )
```

Retorna: Int

Descripción: Método interno utilizado para obtener el valor del parámetro del cgroup del host, si es que este fue establecido.

Parámetros	Tipo	Opcional	Descripción
param	String	NO	Atributo que indica el parámetro a obtener del cgroup.
resource	String	SI	Atributo que indica el recurso a modificar.

Figura 15. Figura 13. Descripción Parámetros Obtener Cgroup

Observación

Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [3]

3.5.3.4.3.5 Eliminar Cgroup

```
def mininet.node.CPULimitedHost.cgroupDel ( self )
```

Retorna: Booleano

Descripción: Método interno utilizado para borrar un cgroup establecido dentro de un host

Observación

Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [3]

3.5.3.4.3.6 Información RT

```
def mininet.node.CPULimitedHost.rtInfo ( self , f )
```

Retorna: Tupla

Descripción: Método interno utilizado para retornar los parámetros de un cgroup entregando la fracción de CPU por parámetro.

Parámetros	Tipo	Opcional	Descripción
f	Float	NO	Atributo que indica la fracción de CPU que recibirá el host (0.3, 0.9 etc) .

Tabla 43. Descripción Parámetros RT

Observación

Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [5]

3.5.3.4.3.7 Información CFS

```
def mininet.node.CPULimitedHost.cfsInfo ( self , f )
```

Retorna: Tupla

Descripción: Método interno utilizado para retornar los parámetros de un cgroup entregando la fracción de CPU por parámetro.

Parámetros	Tipo	Opcional	Descripción
f	Float	NO	Atributo que indica la fracción de CPU que recibirá el host (0.3, 0.9 etc)

Tabla 44. Descripción Parámetros Información CFS

Observación

Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [4]

3.5.3.4.3.8 Establecer Administrador de Prioridad de Tiempo Real

```
def mininet.node.CPULimitedHost.chrt ( self )
```

Retorna: String

Descripción: Método interno utilizado establecer prioridad en tiempo real a los hosts de la red.

Observación

-Método en desarrollo, actualmente solo disponible en kernels en Linux que posee la característica RT_GROUP_SCHED.

-Prioridad por defecto de 20 (rango del 1 al 90, siendo los rangos menores de mayor prioridad).

-Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [5].

3.5.3.4.3.9 Establecer Administrador de Prioridad de Tiempo Real

```
def mininet.node.CPULimitedHost.chrt ( self )
```

Retorna: String

Descripción: Método interno utilizado para establecer prioridad en tiempo real a los hosts de la red.

Observación

-Método en desarrollo, actualmente solo disponible en kernels en Linux que posee la característica RT_GROUP_SCHED.

-Prioridad por defecto de 20 (rango del 1 al 90, siendo los rangos menoress de mayor prioridad).

-Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [5].

3.5.3.4.3.10 Establecer CPU

```
def mininet.node.CPULimitedHost.setCPUs ( self, cores, mems = 0 )
```

Retorna: Void

Descripción: Método utilizado para asignar núcleos reales, que el cgroup del hosts administrara.

Parámetros	Tipo	Opcional	Descripción
cores	Int o Lista	NO	Atributo que indica la cantidad de núcleos reales que se asignara al host
mems	Int	SI	Atributo que indica la cantidad de memoria

Tabla 45. Descripción Parámetros Establecer CPU

Ejemplo

```
# red creada, h1 host de tipo CPUlimitedHost y suponiendo que tenemos
una maquina de 4 núcleos

h1.setCPUs( 2 )
```

El siguiente ejemplo indica que al host h1 se le asignaran 2 núcleos del ordenador para que sean gestionados por el cgroup del nodo.

Observación

-El atributo cores opcionalmente puede ser una Lista, pero es recomendable entregar el parámetro como un entero.

-El atributo mems no es relevante para los desarrolladores de Mininet, no obstante esta especificado de igual manera.

-Mininet establece fracciones de cpu a cada uno de sus nodos virtuales por medio de cgroups. [2].

3.5.3.4.3.11 Configurar

```
def mininet.node.CPULimitedHost.config ( self,
                                         cpu = None,
                                         cores = None,
                                         mac = None
                                         ip = None
                                         defaultRoute = None
                                         lo = 'up'
                                       )
```

Retorna: Diccionario

Descripción: Método utilizado para configurar los parámetros dentro de un Host tipo CPULimitedHost.

Parámetros	Tipo	Opcional	Descripción
mac	String	SI	Atributo que indica la dirección MAC a establecer dentro del nodo.
ip	String	SI	Atributo que indica la dirección IP a establecer dentro del nodo.
defaultRoute	String	SI	Atributo que indica la dirección por defecto de la tabla de enrutamiento del nodo.
lo	String	SI	Atributo que indica si el nodo estará o no disponible.
cores	Int o Lista	SI	Atributo que indica la cantidad de núcleos reales que se asignara al host.
cpu	Float	SI	Atributo que indica la fracción de CPU que recibirá el host (0.3, 0.9 etc).

Tabla 46. Descripción Parámetros Configurar

Ejemplo

```
# red creada, h1 host de tipo CPUlimitedHost

h1.config(ip = '10.0.0.7' , cpu = 0.6)
```

El siguiente ejemplo indica que al host h1 se le asignaran 2 núcleos del ordenador para que sean establecidos por el cgroup del nodo

Observación

El método retorna un diccionario con sus respectivos parámetros, no obstante estos están vacíos.

3.5.4 Archivo link.py

3.5.4.1 Clase Intf

Interfaz de red básica

3.5.4.1.1 Diagrama de Herencia

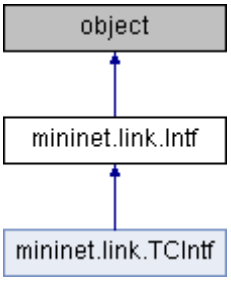


Figura 16. Diagrama de Herencia Clase Intf

3.5.4.1.2 Constructor

```
def mininet.link.Intf.__init__(   self,
                                  name,
                                  ip = None,
                                  node = None,
                                  port = None,
                                  link = None,
                                  mac = None,
                                  ifconfig = None
    )
```

Descripción: Constructor de la Clase Intf, utilizado para crear interfaces de red sobre los nodos.

Parámetros	Tipo	Opcional	Descripción
name	String	NO	Atributo que indica el nombre de la interfaz
node	Node	NO	Atributo que indica el nodo en donde se creara la interfaz.
port	Int	SI	Atributo que indica el puerto al cual será conectada la interfaz.
link	Link	SI	Atributo que indica si el link tendrá una clase especial (TCLink).
mac	String	SI	Atributo que indica la dirección mac de la interfaz.
ifconfig	String	SI	Atributo usado para enviar configuraciones personalizadas .
ip	String	SI	Atributo que indica la dirección ip de la interfaz

Tabla 47. Descripción Parámetros Constructor Intf

Ejemplo

```
# red creada, switch s0

c = Intf( 'eth0' , s0 )
```

El siguiente ejemplo muestra como agregar una interfaz a un switch, en donde el switch s0 se le asignara la interfaz eth0 perteneciente al ordenador.

Observación

Al agregar una interfaz a un nodo, esta debe existir en el sistema.

3.5.4.1.3 Métodos

3.5.4.1.3.1 Cmd

```
def mininet.link.Intf.cmd( self,**args, **kwargs)
```

Retorna: String

Descripción: Método utilizado ejecutar comandos de consola sobre los nodos

Parámetros	Tipo	Opcional	Descripción
****args**	String	NO	Atributo que indica el comando a ejecutar en el nodo.
****kwargs**	String	SI	Atributo que indica el complemento del comando a ejecutar en el nodo.

Tabla 48.Descripción Parámetros Cmd

3.5.4.1.3.2 Ifconfig

```
def mininet.link.Intf.ifconfig( self,**args)
```

Retorna: String

Descripción: Método interno utilizado para enviar comandos personalizados tipo ifconfig

Parámetros	Tipo	Opcional	Descripción
**args	String	NO	Atributo que indica los atributos de ifconfig

Tabla 49. Descripción Parámetros Ifconfig

Ejemplo

```
# red creada, switch s0

c = Intf( 'eth0' , s0 )
c.ifconfig( '192.168.2.2' )
```

El siguiente ejemplo muestra cómo utilizar el comando ifconfig en donde se le entrega por parámetro una dirección IP, esto indica que se le asignara una ip 192.168.2.2 a la interfaz eth0.

Observación

El comando en su totalidad es ifconfig s0 192.168.2.2
[6]

3.5.4.1.3.3 Establecer IP

```
def mininet.link.Intf.setIP( self, ipstr, prefixLen = None)
```

Retorna: String

Descripción: Método utilizado para establecer la dirección IP de una interfaz.

Parámetros	Tipo	Opcional	Descripción
ipstr	String	NO	Atributo que indica la dirección ip a establecer.
prefixLen	String	SI	Atributo que indica la máscara a establecer.

Tabla 50. Descripción Parámetros Establecer IP

Ejemplo

```
# red creada, switch s0

c = Intf( 'eth0' , s0)
c.setIP( '192.168.2.2', ’8’ )
```

El siguiente ejemplo muestra como establecer una IP en una interfaz ya creada (eth0)

Observación

El comando en su totalidad es ifconfig s0 192.168.2.2
[6]

3.5.4.1.3.4 Establecer MAC

```
def mininet.link.Intf.setMAC( self, macstr)
```

Retorna: String

Descripción: Método utilizado para establecer la dirección MAC de una interfaz

Parámetros	Tipo	Opcional	Descripción
macstr	String	NO	Atributo que indica la dirección ip a establecer

Tabla 51. Descripción Parámetros Establecer MAC

Ejemplo

```
# red creada, switch s0

c = Intf( 'eth0' , s0 )
c.setMAC( '00:00:00:00:00:11' )
```

El siguiente ejemplo muestra como establecer una dirección MAC en una interfaz ya creada

3.5.4.1.3.5 Actualizar IP

```
def mininet.link.Intf.updateIP( self)
```

Retorna: String

Descripción: Método utilizado para actualizar la dirección IP de una interfaz.

Ejemplo

```
# red creada, switch s0

c = Intf( 'eth0' , s0)
c.updateIP( )
```

El siguiente ejemplo muestra como actualizar dirección ip en una interfaz ya creada (eth0).

3.5.4.1.3.6 Actualizar MAC

```
def mininet.link.Intf.updateMAC( self )
```

Retorna: String

Descripción: Método utilizado para actualizar la dirección MAC de una interfaz.

Ejemplo

```
# red creada, switch s0

c = Intf( 'eth0' , s0 )
c.setMAC( '00:00:00:00:00:11' )
```

El siguiente ejemplo muestra como actualizar una dirección MAC en una interfaz ya creada.

Observación

Método actualmente con problemas en la versión actual de Mininet, ya que al ser ejecutado no actualiza la dirección MAC de las interfaces.

3.5.4.1.3.7 Actualizar Direcciones

```
def mininet.link.Intf.updateAddr( self )
```

Retorna: Tupla

Descripción: Método utilizado para actualizar la dirección MAC e IP de una interfaz.

Ejemplo

```
# red creada, switch s0

c = Intf( 'eth0' , s0 )
c.setIP( '192.168.2.2', ‘8’ )
c.setMAC( '00:00:00:00:00:11' )
c.updateAddr()
```

El siguiente ejemplo muestra como actualizar las direcciones MAC e IP de una interfaz ya creada.

Observación

Método actualmente con problemas en la versión actual de Mininet, ya que al ser ejecutado no actualiza la dirección MAC de la interfaz

3.5.4.1.3.8 Obtener IP

```
def mininet.link.Intf.IP( self )
```

Retorna: String

Descripción: Método utilizado para obtener la dirección IP de una interfaz.

Ejemplo

```
c = Intf('eth0', s0 )
c.setIP( '192.168.2.2', ‘8’ )
print c.IP()
```

El siguiente ejemplo muestra cómo obtener la dirección IP de una interfaz ya creada.

3.5.4.1.3.9 Obtener MAC

```
def mininet.link.Intf.MAC( self )
```

Retorna: String

Descripción: Método utilizado para obtener la dirección MAC de una interfaz.

Ejemplo

```
c = Intf( 'eth0' , s0 )
c.setMAC( '00:00:00:00:00:06')
print c.MAC()
```

El siguiente ejemplo muestra cómo obtener la dirección MAC de una interfaz ya creada.

3.5.4.1.3.10 Disponibilidad

```
def mininet.link.Intf.IsUp( self, setUp = False )
```

Retorna: Booleano

Descripción: Método utilizado para verificar la disponibilidad de una interfaz.

Parámetros	Tipo	Opcional	Descripción
setUp	Booleano	SI	Atributo que en el caso de ser verdadero establecerá disponibilidad en la interfaz.

Tabla 52. Descripción Parámetros Establecer Disponibilidad

Ejemplo

```
c = Intf( 'eth0' , s0 )
c.isUp( True )
```

El siguiente ejemplo muestra como verificar si una interfaz esta disponible o no.

3.5.4.1.3.11 Renombrar

```
def mininet.link.Intf.rename( self , newname )
```

Retorna: String

Descripción: Método utilizado para renombrar una interfaz de un nodo.

Parámetros	Tipo	Opcional	Descripción
newname	String	NO	Atributo que indica el nuevo nombre que tendrá la interfaz del nodo.

Tabla 53. Descripción Parámetros Renombrar

Ejemplo

```
c = Intf( 'eth0' , s0 )
c.rename( ‘interfaz1’ )
```

El siguiente ejemplo muestra como renombrar una interfaz ya creada.

3.5.4.1.3.12 Configurar

```
def mininet.link.Intf.config(      self,
                                   name,
                                   ip = None
                                   node = None,
                                   port = None,
                                   link = None,
                                   mac = None,
                                   ifconfig = None
                                   up = True
                               )
```

Retorna: Diccionario

Descripción: Método utilizado para configurar los parámetros de una interfaz.

Parámetros	Tipo	Opcional	Descripción
name	String	NO	Atributo que indica el nombre de la interfaz
node	Node	NO	Atributo que indica el nodo en que se creara la interfaz
port	Int	SI	Atributo que indica el puerto al cual será conectada la interfaz.
link	Link	SI	Atributo que indica si el link tendrá una clase especial (TCLink)
mac	String	SI	Atributo que indica la dirección mac de la interface
ifconfig	String	SI	Usado para enviar configuraciones personalizadas

Tabla 54. Descripción Parámetros Configurar

Ejemplo

```
c = Intf( 'eth0' , s0)
c.config( name = 'eth0', ip = '192.0.0.0.0', port = 200, node = s0,
mac = '00:00:00:00:00:09' )
```

El siguiente ejemplo muestra como configurar los parámetros de una interfaz de un switch.

3.5.4.1.3.13 Eliminar Interfaz

```
def mininet.link.Intf.delete( self )
```

Retorna: Void

Descripción: Método utilizado para eliminar una interfaz.

Ejemplo

```
c = Intf( 'eth0' , s0 )
c.delete()
```

El siguiente ejemplo muestra cómo eliminar una interfaz ya creada en un nodo.

3.5.4.1.3.14 Estado

```
def mininet.link.Intf.status( self )
```

Retorna: Void

Descripción: Método utilizado verificar el estado de una interfaz.

Ejemplo

```
c = Intf( 'eth0' , s0 )
print c.status()
```

El siguiente ejemplo muestra el estado de la interfaz ya creada.

3.5.4.1.3.15 Nombre de Clase e Interfaz

```
def mininet.link.Intf.__repr__( self )
```

Retorna: String

Descripción: Método utilizado para retornar el nombre del objeto y el nombre de la interfaz.

Ejemplo

```
c = Intf( 'eth0' , s0 )
print c.__repr__()
```

El siguiente ejemplo muestra el nombre del objeto y el nombre de la interfaz ya creada.

3.5.4.1.3.16 Nombre de Clase e Interfaz

```
def mininet.link.Intf.__str__( self )
```

Retorna: String

Descripción: Método utilizado para retornar el nombre de la interfaz.

Ejemplo

```
c = Intf( 'eth0' , s0 )
print c.__str__()
```

El siguiente ejemplo muestra el nombre de la interfaz ya creada.

3.5.4.2 Clase TCIntf

Interfaz personalizada para control de tráfico, permite especificaciones como: Ancho de banda, retraso, pérdida de paquetes entre otros.

3.5.4.2.1 Diagrama de Herencia

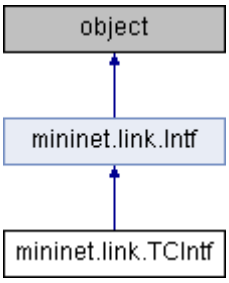


Figura 17. Diagrama de Herencia Clase TCIntf

3.5.4.2.2 Constructor

```
def mininet.link.Intf.__init__(   self,
                                  name,
                                  ip = None,
                                  node = None,
                                  port = None,
                                  link = None,
                                  mac = None,
                                  ifconfig = None
    )
```

Parámetros	Tipo	Opcional	Descripción
name	String	NO	Atributo que indica el nombre de la interfaz
node	Node	NO	Atributo que indica el nodo en donde se creara la interfaz.
port	Int	SI	Atributo que indica el puerto al cual será conectada la interfaz.
link	Link	SI	Atributo que indica si el link sera personalizado (TCLink).
mac	String	SI	Atributo que indica la dirección mac de la interfaz.
ifconfig	String	SI	Atributo usado para enviar configuraciones personalizadas .
ip	String	SI	Atributo que indica la dirección ip de la interfaz

Tabla 55. Descripción Constructor Clase TCIntf

Ejemplo

```
# red creada, switch s0

c = TCIntf( 'eth0' , s0 )
```

El siguiente ejemplo muestra como agregar una interfaz a un switch, en donde el switch s0 se le asignara la interfaz eth0 perteneciente al ordenador.

Observación

- Al agregar una interfaz a un nodo, esta debe existir en el sistema.
- El constructor de la clase TCIntf es el constructor heredado de la Clase Intf.

3.5.4.2.3 Métodos

3.5.4.2.3.1 Configurar

```
def mininet.link.TCIntf.config(    self,
                                   bw = None,
                                   delay = None,
                                   jitter = None,
                                   loss = None,
                                   speedup = 0,
                                   max_queue_size = None
                                   disable_gro = True,
                                   latency_ms = None,
                                   use_hfsc = False,
                                   use_tbf = False,
                                   enable_ecn = False,
                                   enable_red = False,

                               )
```

Retorna: Diccionario

Descripción: Método utilizado para configurar una interfaz personalizada.

Parámetros	Tipo	Opcional	Descripción
bw	Int	SI	Atributo que indica el ancho de banda que tendrá la interfaz (rango 0 a 1000).
delay	String	SI	Atributo que indica el retardo que tendrá la interfaz.
loss	Int	SI	Atributo que indica el porcentaje de perdida que sufrirán los paquetes de la interfaz.
jitter	String	SI	Atributo que indica la cantidad de señales de ruido que tendrá la interfaz.
max_queue_size	Int	SI	Atributo que indica el tamaño del buffer de los paquetes.
speedup	Int	SI	Atributo que indica la aceleración de ancho de banda que tendrá la interfaz.
disable_gro	Booleano	SI	Atributo de uso interno
latency_ms	Int	SI	Atributo de uso interno
use_hfsc	Booleano	SI	Atributo de uso interno
Use_tbf	Booleano	SI	Atributo de uso interno
enable_ecn	Booleano	SI	Atributo de uso interno
enable_red	Booleano	SI	Atributo de uso interno

Tabla 56.Descripción Parámetros Configurar

Observación

El método config() es utilizado para la personalización de un par de interfaces de un link, es un error hacer uso de esta función de forma individual.

3.5.4.3 Clase Link

La clase link proporciona conexión entre dos nodos.

3.5.4.3.1 Diagrama de Herencia

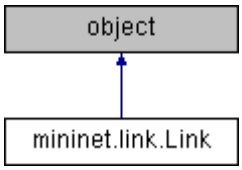


Figura 18. Diagrama de Herencia Clase Link

3.5.4.3.2 Constructor

```
def mininet.link.Link.__init__ (  self,
                                  node1,
                                  node2,
                                  port1 = None,
                                  port2 = None,
                                  intfName1 = None,
                                  intfName2 = None,
                                  addr1 = None,
                                  addr2 = None,
                                  intf = Intf,
                                  cls1 = None,
                                  cls2 = None,
                                  params1 = None,
                                  params2 = None
      )
```

Descripción: Constructor de la clase Link, utilizado para crear un link entre dos nodos.

Parámetros	Tipo	Opcional	Descripción
node1	Node	NO	Atributo que indica el primer nodo para generar el enlace.
node2	Node	NO	Atributo que indica el segundo nodo para generar el enlace.
port1	Int	SI	Atributo que indica el puerto de comunicación de la primera interfaz.
Port2	Int	SI	Atributo que indica el puerto de comunicación de la segunda interfaz.
intfName1	String	SI	Atributo que indica el nombre de la primera interfaz
intfName2	String	SI	Atributo que indica el nombre de la segunda interfaz.
addr1	String	SI	Atributo que indica la dirección MAC de la primera interfaz
Addr2	String	SI	Atributo que indica la dirección MAC de la segunda interfaz
intf	Intf	SI	Atributo que indica si el enlace será personalizado (TCLink).
params1	Diccionario	SI	Parámetros para la configuración personalizada de la primera interfaz (name, node,port,link,mac,ifconfig y ip)
Params2	Diccionario	SI	Parámetros para la configuración personalizada de la segunda interfaz (name, node,port,link,mac,ifconfig y ip)

Tabla 57. Descripción Parámetros Constructor Clase Link

Ejemplo

Red creada host h1, h2, switch s0.

```
h1s1 = {'bw':1000 }
h2s2 = {'bw':1000 }
link = Link(node1=h1,node2=s0,intf=TCIntf,params1=h1s1,params2=h2s2)
link2= Link(node1=h2,node2=s0,intf=TCIntf,params1=h1s1,params2=h2s2)
```

El siguiente ejemplo muestra como agregar un link entre dos nodos utilizando la clase Link, en un principio se crean 2 diccionarios para entregarlos por parámetros a los link personalizados.

Observación

No es recomendable agregar los link de esta manera a menos que sea estrictamente necesario, esto es debido a que se puede causar conflictos con el envió de parámetros y mas de un atributo puede provocar que el script no se ejecute en su totalidad (como es el caso si agregamos el atributo addr1 o intfName) lo mejor es utilizar el método addLink() de la clase Mininet.

3.5.4.3.3 Métodos

3.5.4.3.3.1 Generar nombre de Interfaz

```
def mininet.link.Link.intfName ( self, node , n )
```

Retorna: String

Descripción: Método interno que construye un nombre de interfaz canónico (Nombre del nodo-nombre de interfaz)

Parámetros	Tipo	Opcional	Descripción
node	Node	NO	Nodo entregado por parámetro.
n	Intf	NO	Interfaz entregadada por parámetro.

Tabla 58. Descripción Constructor Clase TCIntf

3.5.4.3.3.2 Eliminar Link

```
def mininet.link.Link.delete ( self )
```

Retorna: Void

Descripción: Método utilizado para eliminar un Link entre dos nodos.

Ejemplo

```
# Red creada host h1, h2, switch s0.

link = Link(node1=h1,node2=s0)
link2 = Link(node1=h2,node2=s0)
link.delete()
```

El siguiente ejemplo muestra como agregar un link entre dos nodos utilizando la clase Link, en un principio se crean 2 diccionarios para entregarlos por parámetros a los link personalizados.

3.5.4.3.3.3 Estado

```
def mininet.link.Link.status ( self )
```

Retorna: String

Descripción: Método utilizado para ver el estado de las interfaces (Estado del Link).

Ejemplo

```
# Red creada host h1, h2, switch s0.

link = Link(node1=h1,node2=s0)
link2 = Link(node1=h2,node2=s0)
link.delete()
print 'Link 1 ' +link.status()
print 'Link 2 ' +link2.status()
```

El siguiente ejemplo muestra la creación de 2 links y la impresión de sus estados. El resultado se vería como esto:

```
Link 1 (MISSING MISSING)
Link 2 (OK OK)
```

3.5.4.3.3.4 Obtener nombre

```
def mininet.link.Link.__str__ ( self )
```

Retorna: String

Descripción: Método utilizado para obtener el nombre del link

Ejemplo

```
# Red creada host h1, h2, switch s0.

link = Link(node1=h1,node2=s0)
link2 = Link(node1=h2,node2=s0)
print link.__str__()
```

El siguiente ejemplo muestra la creación de 2 links y la impresión del nombre del primer link. El resultado se vería como esto:

h1-eth0<->s0-eth1

3.5.4.4 Clase TCLink

La clase TCLink proporciona la capacidad de crear un Link con características personalizadas entre dos nodos, estos nodos poseerán interfaces simétricas en cuanto a su personalización.

3.5.4.4.1 Diagrama de Herencia

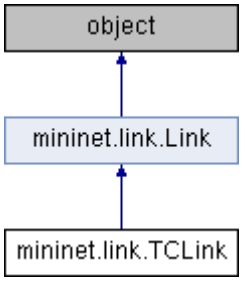


Figura 19. Diagrama de Herencia Clase TCLink

3.5.4.4.2 Constructor

```
def mininet.link.TCLink.__init__ ( self,
                                   node1,
                                   node2,
                                   port1 = None,
                                   port2 = None,
                                   intfName1 = None,
                                   intfName2 = None,
                                   addr1 = None,
                                   addr2 = None,
                                   intf = Intf,
                                   cls1 = None,
                                   cls2 = None,
                                   bw = None,
                                   delay = None,
                                   jitter = None,
                                   loss = None,
                                   speedup = 0,
                                   max_queue_size = None
        )
```

Descripción: Constructor de la clase TCLink permite crear un Link entre dos nodos con personalizaciones simétricas en sus interfaces.

Parámetros	Tipo	Opcional	Descripción
node1	Node	NO	Atributo que indica el primero nodo para generar el enlace.
node2	Node	NO	Atributo que indica el segundo nodo para generar el enlace.
port1	Int	SI	Atributo que indica el puerto de comunicación de la primera interfaz.
Port2	Int	SI	Atributo que indica el puerto de comunicación de la segunda interfaz.
intfName1	String	SI	Atributo que indica el nombre de la primera interfaz.
intfName2	String	SI	Atributo que indica el nombre de la segunda interfaz.
addr1	String	SI	Atributo que indica la dirección MAC de la primera interfaz.
Addr2	String	SI	Atributo que indica la dirección MAC de la segunda interfaz.
Intf	Intf	SI	Atributo que indica si el enlace será personalizado (TCLink).
Bw	Int	SI	Atributo que indica el ancho de banda que tendrá la interfaz (rango 0 a 1000).
Delay	String	SI	Atributo que indica el retardo que tendrá la interfaz.
Loss	Int	SI	Atributo que indica el porcentaje de perdida que sufrirán los paquetes de la interfaz.
Jitter	String	SI	Atributo que indica la cantidad de señales de ruido que tendrá la interfaz.
max_queue_size	Int	SI	Atributo que indica el tamaño del buffer de los paquetes.
speedup	Int	SI	Atributo que indica la aceleración de ancho de banda que tendrá la interfaz.

Tabla 59. Descripción Parámetros Constructor TCLink

Ejemplo

```
# Red creada host h1, h2, switch s0.

link = TCLink(node1 = h1,node2 = s0,bw = 5000,delay = '3000', loss = 40 )
link2 = Link(node1 = h2, node2 = s0 )
```

El siguiente ejemplo muestra la creación de 2 links, uno de ellos (link) tiene parámetros de personalización especificando que tendrá un ancho de bando de 5 GBytes por segundo, un retraso de 3 segundos y un porcentaje de pérdida del 40%.

Observación

Este modo de ingreso es solo efectivo cuando es utilizado a bajo nivel [Ver Seccion 3.5.1 API Nivel Bajo]

3.5.5 Archivo cli.py

3.5.5.1 Clase CLI

3.5.5.1.1 Diagrama de Herencia

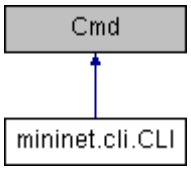


Figura 20. Diagrama de Herencia Clase CLI

3.5.5.1.2 Constructor

```
def mininet.cli.CLI.__init__ (   self,
                                 mininet,
                                 stdin = sys.stdin,
                                 script = None
)
```

Descripción: Constructor de la clase CLI, permite crear una consola de comandos para interactuar con los nodos de la red.

Parámetros	Tipo	Opcional	Descripción
mininet	Mininet	NO	Red mininet ya creada.
stdin	File	SI	Atributo de uso interno.
script	String	SI	Atributo de uso interno .

Tabla 60. Descripción Parámetros Constructor Clase CLI

Ejemplo

```
net = Mininet(topo = None,build = False,ipBase = '10.0.0.0/8' )
# Red ya creada e iniciada
CLI(net)
```

Observación

La creación de una consola de comandos es solo efectiva en nivel medio y alto de la API Mininet[Ver Seccion 3.5.1 API Nivel medio, API de alto Nivel]

3.5.6 Archivo topo.py

3.5.6.1 Clase Topo

Clase utilizada para la creación de topologías personalizadas

3.5.6.1.1 Diagrama de Herencia

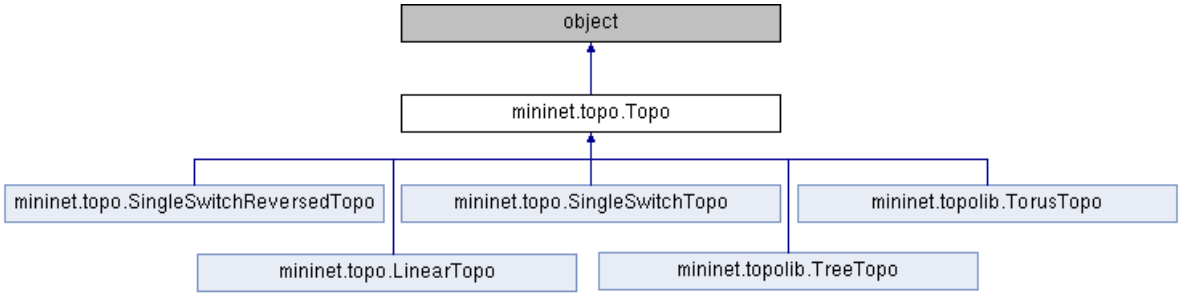


Figura 21. Diagrama de Herencia Clase Topo

3.5.6.1.2 Constructor

```
def mininet.topo.Topo.__init__ ( self,
                                 hopts = None,
                                 sopts = None,
                                 lopts = None
)
```

Descripción: Constructor de la clase Topo que permite crear topologías personalizadaz.

Parámetros	Tipo	Opcional	Descripción
hopts	Diccionario	SI	Parámetros generales de los host (cpu,cores etc)
sopts	Diccionario	SI	Parámetros generales de los switchs.
lopts	Diccionario	SI	Parámetros generales de los links (ancho de banda, retardo entre otros)

Tabla 61. Descripción Parámetros Constructor Clase Topo

Observación
En la actual versión de Mininet el atributo hopts no es funcional

Ejemplo

```
a = {'loss': 10 , 'bw': 500 }
b = { 'cpu' = 0.5 }
topos = Topo( lopts = a , hopts = b )
```

El siguiente ejemplo muestra cómo crear la base de una topología personalizada, indicando por parámetros que los enlaces entre nodos tendrán un 10% de perdida en sus paquetes y su máximo de ancho de banda es de 500 Mega Bytes por segundo, además cada host tendrá limitaciones en el uso de cpu (50%).

3.5.6.1.3 Métodos

3.5.6.1.3.1 Añadir Nodo

```
def mininet.topo.Topo.addNode (  self, name, **opts )
```

Return : String

Descripción: Método utilizado para añadir un nodo a la topología, retorna el nombre del nodo.

Parámetros	Tipo	Opcional	Descripción
name	String	NO	Atributo que indica el nombre nodo.
Opts	Diccionario	SI	Parámetros generales del nodo (configuraciones personalizadas host, switch y controlador).

Tabla 62. Descripción Parámetros Añadir Nodo

Ejemplo

```
a = {'loss': 10 , 'bw': 500 }
b = { 'cpu' = 0.5 }
topos = Topo( lopts = a , hopts = b )
topos.addNode( 'h1' )
topos.addNode( 'h2' )
```

El siguiente ejemplo muestra cómo crear la base de una topología personalizada, indicando por parámetros que los enlaces entre nodos tendrán un 10% de perdida en sus paquetes y su máximo de ancho de banda es de 500 Mega Bytes por segundo, además cada host tendrá limitaciones en el uso de cpu (50%), luego se procede añadir dos nodos a la topologia indicando sus nombres.

3.5.6.1.3.2 Añadir Host

```
def mininet.topo.Topo.addHost (  self, name, **opts )
```

Return : String

Descripción: Método utilizado para añadir un host a una topología, retorna el nombre del host

Parámetros	Tipo	Opcional	Descripción
name	String	SI	Atributo que indica el nombre del host.
Opts	Diccionario	SI	Parámetros generales del host (cpu,cores etc).

Tabla 63. Descripción Parámetros Añadir Host

Ejemplo

```
a = {'loss': 10 , 'bw': 500 }
b = { 'cpu' = 0.5 }
topos = Topo( lopts = a , hopts = b )
topos.addHost( 'h1' )
topos.addHost( 'h2' )
```

El siguiente ejemplo muestra cómo crear la base de una topología personalizada, indicando por parámetros que los enlaces entre nodos tendrán un 10% de perdida en sus paquetes y su máximo de ancho de banda es de 500 Mega Bytes por segundo, además cada host tendrá limitaciones en el uso de cpu (50%), luego se procede añadir dos nodos a la topologia indicando sus nombres.

3.5.6.1.3.3 Añadir Switch

```
def mininet.topo.Topo.addSwitch (  self, name, **opts )
```

Return : String

Descripción: Método utilizado para añadir un switch a una topología, retorna el nombre del switch.

Parámetros	Tipo	Opcional	Descripción
name	String	SI	Atributo que indica el nombre del switch
Opts	Diccionario	SI	Parámetros generales del switch

Tabla 64. Descripción Parámetros Añadir Host

Ejemplo

```
topos = Topo()
topos.addSwitch( 's0' )
```

El siguiente ejemplo muestra cómo agregar un switch a una topología.

3.5.6.1.3.4 Añadir Link

```
def mininet.net.Topo.addLink(     self,
                                  node1,
                                  node2,
                                  port1 = None,
                                  port2 = None,
                                  cls = None,
                                  addr1,
                                  addr2,
                             )
```

Retorna: Link

Descripción: Método que añade un link a una topología.

Parámetros	Tipo	Opcional	Descripción
node1	Node	NO	Atributo que indica el nodo con que se realizara el link.
node2	Node	NO	Atributo que indica el nodo con que se realizara el link.
port1	Int	SI	Atributo que indica el puerto de conexión de la interfaz de red del primer nodo pasado por parámetro.
port2	Int	SI	Atributo que indica el puerto de conexión de la interfaz de red del segundo nodo pasado por parámetro.
Cls	Link	SI	Atributo que indica si será un tipo de link especial como TCLink que permite que el enlace sea personalizado (Ancho de banda, Jitter, porcentaje de perdida entre otros).
addr1	String	SI	Atributo que indica la dirección MAC del host.
addr2	String	SI	Atributo que indica la dirección MAC del switch.

Tabla 65. Descripción Parámetros Añadir Link

Ejemplo

```
topos = Topo()
topos.addHost('h1')
topos.addSwitch( 's0' )
topos.addLink('h1','s0',port1=7000,port2=7000,addr1='00:00:00:00:00:01',
addr2 ='00:00:00:00:00:02')
```

El siguiente ejemplo muestra como añadir un link entre dos nodos, el host h1 se comunicara con el switch por medio del puerto 7000 y viceversa, además indica que la dirección MAC del host h1 será '00:00:00:00:00:01' y la dirección MAC del enlace con el host en el switch será de '00:00:00:00:00:02'.

Observación

Si la dirección MAC es agregada al momento de configurar el host esta no será modificada por el parámetro addr1 de addLink.

3.5.6.1.3.5 Obtener nodos

```
def mininet.net.Topo.nodes( self , sort = True )
```

Retorna: Lista

Descripción: Método utilizado para obtener los nombres de los nodos presentes en la topología.

Parámetros	Tipo	Opcional	Descripción
sort	Booleano	Si	Atributo que indica si la lista sera ordenada alfabéticamente.

Tabla 66. . Descripción Parámetros Obtener Nodos

Ejemplo

```
# Topología ya creada (topo)

print topo.nodes()
```

El siguiente ejemplo muestra como imprimir una lista de nodos presentes en una topología.

3.5.6.1.3.6 Verificar Switch

```
def mininet.net.Topo.isSwitch( self , n )
```

Retorna: Booleano

Descripción: Método utilizado para verificar si un nodo en una la topología es un switch.

Parámetros	Tipo	Opcional	Descripción
n	String	Si	Atributo que indica si la lista sera ordenada alfabéticamente.

Tabla 67. Descripción Parámetros Verificar Switch

Ejemplo

```
# Topología ya creada (topo), nombre del switch s0

print topo.isSwitch( 's0' )
```

El siguiente ejemplo indica si el nodo s0 es un switch o no.

3.5.6.1.3.7 Obtener Switchs

```
def mininet.net.Topo.switches( self , sort = True )
```

Retorna: Lista

Descripción: Método utilizado para obtener los nombres de los switchs presentes en una topología.

Parámetros	Tipo	Opcional	Descripción
sort	Booleano	Si	Atributo que indica si la lista debe ser ordenada alfabéticamente.

Tabla 68. Descripción Parámetros Obtener Switchs

Ejemplo

```
# Topología ya creada (topo)

print topo.switches()
```

El siguiente ejemplo muestra como imprimir una lista con los switchs presentes en una topología.

3.5.6.1.3.8 Obtener Hosts

```
def mininet.net.Topo.hosts( self , sort = True )
```

Retorna: Lista

Descripción: Método utilizado para obtener los nombres de los hosts presentes en una topología.

Parámetros	Tipo	Opcional	Descripción
sort	Booleano	Si	Atributo que indica si la lista debe ser ordenada alfabéticamente.

Tabla 69. Descripción Parámetros Obtener Hosts

Ejemplo

```
# Topología ya creada (topo)

print topo.hosts()
```

El siguiente ejemplo muestra como imprimir una lista con los hosts presentes en una topología

3.5.6.1.3.9 Obtener Links

```
def mininet.net.Topo.links( self , sort = True )
```

Retorna: Lista

Descripción: Método utilizado para obtener los nombres de los links presentes en una topología.

Parámetros	Tipo	Opcional	Descripción
sort	Booleano	Si	Atributo que indica si la lista debe ser ordenada alfabéticamente.

Tabla 70. Descripción Parámetros Obtener Links

Ejemplo

```
# Topología ya creada (topo)
print topo.links()
```

El siguiente ejemplo muestra como imprimir una lista con los links presentes en una topología.

3.5.6.2 Clase SingleSwitchTopo

3.5.6.2.1 Diagrama de Herencia

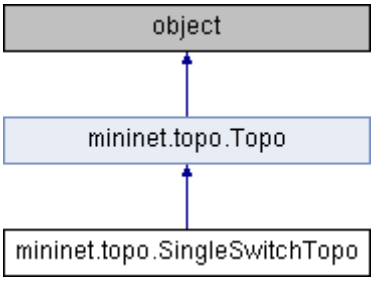


Figura 22. Diagrama de Herencia Clase SingleSwitchTopo

3.5.6.2.2 Constructor

```
def mininet.topo.SingleSwitchTopo.__init__ ( self,
                                              k = 2,
                                              hopts = None,
                                              sopts = None,
                                              lopts = None
                                            )
```

Descripción: Constructor de la clase SingleSwitchTopo, permite crear una topología personalizada (K hosts conectados a un switch).

Parámetros	Tipo	Opcional	Descripción
K	Int	SI	Atributo que indica la cantidad de host que estarán conectados al switch.
hopts	Diccionario	SI	Parámetros generales de los host (cpu, cores etc).
sopts	Diccionario	SI	Parámetros generales de los switchs.
lopts	Diccionario	SI	Parámetros generales de los links (ancho de banda, retardo entre otros).

Tabla 71. Descripción Parámetros Constructor Clase SingleSwitchTopo

Ejemplo

```
topos = SingleSwitchTopo( k = 10 )
```

El siguiente ejemplo muestra cómo crear una red con 10 host conectados a un switch.

Observación

En la actual versión de mininet el atributo hopts no es funcional.

3.5.6.3 Clase SingleSwitchReversedTopo

3.5.6.3.1 Diagrama de Herencia

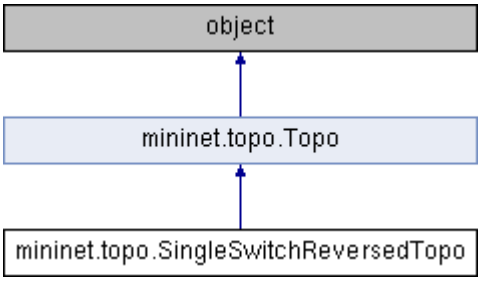


Figura 23. Diagrama de Herencia Clase SingleSwitchReversedTopo

3.5.6.3.2 Constructor

```
def mininet.topo.SingleSwitchReversedTopo.__init__ (  self,
                                                      k = 2,
                                                      hopts = None,
                                                      sopts = None,
                                                      lopts = None
)
```

Descripción: Constructor de la clase SingleSwitchReversedTopo permite crear una topología personalizada (K hosts conectados a un switch con puertos reservados)

Parámetros	Tipo	Opcional	Descripción
K	Int	SI	Atributo que indica la cantidad de host que estarán conectados al switch.
hopts	Diccionario	SI	Parámetros generales de los host (cpu, cores etc).
sopts	Diccionario	SI	Parámetros generales de los switchs.
lopts	Diccionario	SI	Parámetros generales de los links (ancho de banda, retardo entre otros).

Tabla 72 . Descripción Parámetros Constructor Clase SingleSwitchReversedTopo

Ejemplo

```
topos = SingleSwitchReversedTopo( k = 10 )
```

El siguiente ejemplo muestra cómo crear una red con 10 host conectados a un switch con puertos reservados en sus enlaces.

Observación

En la actual versión de mininet el atributo hopts no es funcional.

3.5.6.4 Clase LinearTopo

3.5.6.4.1 Diagrama de Herencia

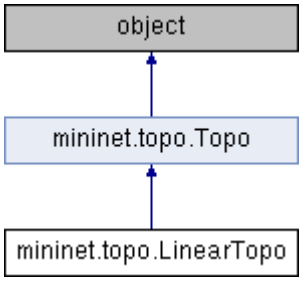


Figura 24. Diagrama de Herencia Clase LinearTopo

3.5.6.4.2 Constructor

```
def mininet.topo.LinearTopo.__init__ ( self,
                                       k = 2,
                                       n = 1
                                       hopts = None,
                                       sopts = None,
                                       lopts = None
                                     )
```

Descripción: Constructor de la clase LinearTopo permite crear una topología personalizada (k switchs con n hosts, además los switchs estarán conectados entre sí)

Parámetros	Tipo	Opcional	Descripción
N	Int	SI	Atributo que indica la cantidad de host que estarán conectados a cada switch.
K	Int	SI	Atributo que indica la cantidad de switchs presentes en la red.
hopts	Diccionario	SI	Parámetros generales de los host (cpu, cores etc).
sopts	Diccionario	SI	Parámetros generales de los switchs.
lopts	Diccionario	SI	Parámetros generales de los links (ancho de banda, retardo entre otros).

Tabla 73. Descripción Parámetros Constructor Clase LinearTopo

Ejemplo

```
topos = LinearTopo ( k = 3 , n = 3 )
```

El siguiente ejemplo muestra cómo crear una red con 9 host conectados a 3 switch (por cada switch 3 host).

Observación

En la actual versión de mininet el atributo hopts aun no es funcional, causando inconsistencias.

3.5.7 Archivo topolib.py

3.5.7.1 Clase TreeTopo

3.5.7.1.1 Diagrama de Herencia

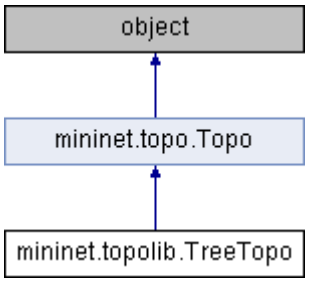


Figura 25. Diagrama de Herencia Clase TreeTopo

3.5.7.1.2 Constructor

```
def mininet.topolib.TreeTopo.__init__ ( self,
                                        depth = 2,
                                        fanout = 1
                                        hopts = None,
                                        sopts = None,
                                        lopts = None
)
```

Descripción: Constructor de la clase TreeTopo permite crear una topología tipo árbol personalizable (cantidad de switchs por nivel: fanout elevado depth – 1. Cantidad de host en el último nivel fanout elevado a depth).

Parámetros	Tipo	Opcional	Descripción
depth	Int	SI	Atributo que indica la cantidad de niveles presentes en el árbol. (formalmente son depth mas uno niveles)
fanout	Int	SI	Atributo que indica la cantidad de switchs por nivel y la cantidad de host que estarán conectados a los switchs del ultimo nivel del árbol.
hopts	Diccionario	SI	Parámetros generales de los host (cpu,cores etc).
sopts	Diccionario	SI	Parámetros generales de los switchs.
lopts	Diccionario	SI	Parámetros generales de los links (ancho de banda, retardo entre otros).

Tabla 74. Descripción Parámetros Constructor Clase TreeTopo

Ejemplo

```
topos = TreeTopo ( k = 3 , n = 3 )
```

El siguiente ejemplo muestra cómo crear una red con 27 host y13 switchs (por cada switch 3 host conectados al los switchs del ultimo nivel).

Observación

Los tiempos de ejecución pueden ser tardíos, en la creación de árboles de mas de 4 niveles .

3.5.7.2 Clase TorusTopo

3.5.7.2.1 Diagrama de Herencia

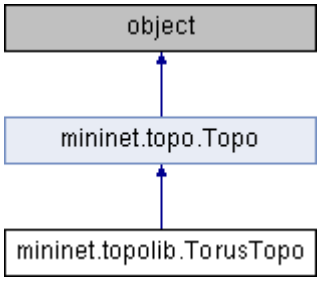


Figura 26. Diagrama de Herencia Clase TorusTopo

3.5.7.2.2 Constructor

```
def mininet.topolib.TorusTopo.__init__ ( self,
                                         x = 2,
                                         y = 1
                                         hopts = None,
                                         sopts = None,
                                         lopts = None
)
```

Descripción: Constructor de la clase TorusTopo permite crear una topología tipo Torus-2D personalizable.

Observación

[8]

Parámetros	Tipo	Opcional	Descripción
x	Int	SI	Atributo que indica la cantidad de filas que tendrá la red torus .
y	Int	SI	Atributo que indica la cantidad de columnas que tendra la red torus.
hopts	Diccionario	SI	Parámetros generales de los host (cpu, cores etc).
sopts	Diccionario	SI	Parámetros generales de los switchs.
lopts	Diccionario	SI	Parámetros generales de los links (ancho de banda, retardo entre otros).

Tabla 75. Descripción Parámetros Constructor Clase TorusTopo

Ejemplo

```
topos = TorusTopo ( x = 4 , y = 4 )
```

El siguiente ejemplo muestra cómo crear una red torus 2D con 16 host y16 switchs.

Observación

Los tiempos de ejecución pueden ser tardíos, en la creación de topologías tipo torus con x>4 e y>4.

3.5.7.2.3 Funciones

3.5.7.2.3.1 Topología Árbol

```
def mininet.topolib.TreeNet ( depth = 2,
                              fanout = 1,
                              hopts = None,
                              sopts = None,
                              lopts = None
                            )
```

Retorna: Mininet

Descripción: Función que permite crear una topología tipo árbol personalizable (cantidad de switchs por nivel: fanout elevado depth – 1. Cantidad de host en el último nivel fanout elevado a depth)

Parámetros	Tipo	Opcional	Descripción
depth	Int	SI	Atributo que indica la cantidad de niveles presentes en el árbol. (Formalmente son depth mas uno niveles).
fanout	Int	SI	Atributo que indica la cantidad de switchs por nivel y la cantidad de host que estarán conectados a los switchs del último nivel del árbol.
hopts	Diccionario	SI	Parámetros generales de los host (cpu, cores etc).
sopts	Diccionario	SI	Parámetros generales de los switchs.
lopts	Diccionario	SI	Parámetros generales de los links (ancho de banda, retardo entre otros).

Tabla 76. Descripción Parámetros Topología Árbol

Ejemplo

```
topo = TreeNet ( k = 3 , n = 3 )
```

El siguiente ejemplo muestra cómo crear una red con 27 host y13 switchs (por cada switch 3 host conectados al los switchs del ultimo nivel).

Observación

-Los tiempos de ejecución pueden ser tardíos, en la creación de árboles de más de 4 niveles.

-El método retorna un objeto tipo Mininet, esto quiere decir que no es necesario pasar la topología a un objeto tipo Mininet, basta con iniciar con el método start().

3.5.8 Archivo term.py

3.5.8.1 Funciones

3.5.8.1.1 Crear Terminales

```
def mininet.cli.makeTerms (  node,
                             title ='Node' ,
                             term = 'xterm' ,
)
```

Retorna: Lista

Descripción: Función utilizada para la creación de terminales sobre los nodos de la red.

Parámetros	Tipo	Opcional	Descripción
node	Lista de Node	NO	Nodos a los que se les creara una consola.
title	String	SI	Titulo de la consola.
term	String	SI	Tipo de terminal (Xterm o Gterm).

Tabla 77. Descripción Parámetros Crear Terminales

Ejemplo

```
# Red ya creada e iniciada hosts: h1, h2 y h3. Switch s0

makeTerms(( h1 , h2 , h3 ,s0 ), title = 'Nodos')
```

El siguiente ejemplo muestra cómo crear un conjunto de terminales para los nodos h1, h2, h3 y S0.

Observación

En el caso de trabajar con terminales Gterm cabe destacar que mininet solo creara una terminal de este tipo, debido a lo anterior es recomendable trabajar con terminales del tipo Xterm.

3.5.8.1.2 Crear Terminal

```
def mininet.cli.makeTerm (   node,
                             title ='Node' ,
                             term = 'xterm' ,
                             display = None
)
```

Retorna: Lista

Descripción: Función utilizada para la creación de una terminal sobre un nodo.

Parámetros	Tipo	Opcional	Descripción
node	Node	NO	Nodos a los que se les creara una consola.
title	String	SI	Título de la consola.
term	String	SI	Tipo de terminal (Xterm o Gterm).
display	String	SI	Atributo de uso interno.

Tabla 78. Descripción Parámetros Crear Terminal

Ejemplo

```
# Red ya creada e iniciada hosts: h1,h2 y h3. Switch s0

makeTerm( h1, title = ‘Terminal Host h1’)
```

El siguiente ejemplo muestra cómo crear una terminal sobre un host.

Observación

En el caso de trabajar con terminales Gterm cabe destacar que mininet solo creara una terminal de este tipo, debido a lo anterior es recomendable trabajar con terminales del tipo Xterm.

3.5.9 Archivo Clean.py

3.5.9.1 Funciones

3.5.9.1.1 Limpiar

def mininet.clean.cleanup ( )

Retorna: Void

Descripción: Función utilizada para limpiar todos los elementos de la red que fueron creados.

Ejemplo

```
#Red creada e iniciada
#última línea de código del script

cleanup()
```

EL siguiente ejemplo muestra como limpiar nuestro ordenador luego de ejecutar un script Mininet.

Observaciones

-Al cerrar un script en Mininet no siempre se limpiaara adecuadamente, debido a lo anterior es recomendable el uso de la funcion cleanup()

-Actualmente Mininet solo permite la ejecución de un script a la vez, si se llegase a ejecutar un segundo este inutiliza al primero.

3.5.10 Consideraciones generales en la creación de scripts

Al generar scripts en mininet están las siguientes consideraciones

- Al asignar un puerto a un Open vSwitch se debe tomar en cuenta que el puerto no sea el mismo de controlador de lo contrario es el script no correra.
- Todo controlador que tenga enlaces a uno o mas switch debe ser iniciado antes que los switches a los que esta enlasado.
- Todo switch que posea un enlace a un controlador deben ser iniciados junto a su controlador por parámetro.
- Todo switch y controlador debe ser iniciado en la red con el método start().
- Toda red debe ser detenida después de ser utilizada.
- Todo link, nodo y host puede tener una y solo una clase personalizada cls.
- Los legacy switch no requieren y no pueden conectarse a un controlador para obtener conectividad entre los host de una red.
- No se puede realizar conectividad directa (punto a punto) entre dos host, host y un controlador o entre dos controladores, toda conexión debe ser por medio de un switch.
- Toda simulación debe ser realizada después de iniciar todos los nodos y enlaces de la red y antes de lanzar la consola CLI , si es que esta estuviera presente en los scripts.
- No es recomendable la combinación de niveles de la API de mininet a menos que se tenga un buen conocimiento de esta.

4 DEFINICIÓN PROYECTO

4.1 Objetivos del proyecto

Se definen los objetivos generales y específicos que se desean lograr por medio del proyecto actual.

4.1.1 Objetivo General

Diseñar y construir un software que permita la generación de scripts para la creación y simulación de redes compatibles con "Mininet".

4.1.2 Objetivos Específicos

- Desarollar una interfaz gráfica generadora de script para la creación de redes definidas por el usuario.
- Mejorar la documentación de la API existente de Mininet.
- Efectuar una simulación de un escenario de redes de computadoras utilizando un script generado.

4.2 Ambiente de Ingeniería de Software

4.2.1 Justificación de Metodología de Desarrollo

Teniendo en cuenta la poca documentación e información mas profunda sobre "Mininet" y la falta de experiencia del alumno tesista, se ha escogido una metodología de entrega por etapas, debido a que permite ver el proyecto de forma general a un nivel mas detallado, esto permite la retroalimentación que brindara el estudio continuo del manejo de los temas y la aplicación misma.

Una dificultad asociada a esta metodología será la especificación de requerimientos que deberá ser clara y bien especificada, de igual forma las etapas deben ser claras y diferenciables y retroalimentándose de las etapas anteriores para que puedan ser implementadas de forma efectiva, para aquello la investigación continúa y la experiencia del profesor guía será fundamental.

4.2.2 Técnicas y notaciones

4.2.3 Estándares de documentación, producto o proceso

Para mantener una lectura y comprensión técnica del documento que respalda el desarrollo del software, se deberá usar un estándar de acuerdo a la necesidad del proyecto.

Es por esto que se utilizará una adaptación basada en IEEE Software Test Documentation Std 829-1998, la cual proveerá las herramientas para la correcta comprension e interpretación del proyecto. Cada seccion (Con excepción de la seccion 3) de este documento será revisado según este estándar y se deberá considerar en todo momento de explicación y redacción.

4.2.4 Herramientas de apoyo al desarrollo de software que serán utilizadas

- **Netbeans IDE 8.0:** Herramienta de desarrollo de código fuente.
- **Gedit:** Herramienta de edición y lectura de código fuentes.
- **Wireshark:** Herramienta de análisis de trafico de paquetes.
- **GNOME Terminal:** Terminal de comandos en Linux basada en GNOME.
- **XTERM**: Terminal ligera de comandos de Linux.
- **Iperf:** Herramienta utilizada para la medición de ancho de banda.
- **Ping:** Herramienta utilizada para comprobar la comunión entre un nodo origen y destino**.**
- **Power Designer**: Utilizado para el desarrollo de Casos Uso.
- **MS Proyect**: Utilizado para la creación de Cartas Gantt dentro del proyecto.
- **GIT:** Utilizado para repositorios dentro del Sw.
- **Sublime Text 2.0.2:** Utilizado como plataforma de texto en el desarrollo de programación del software.
- **Balsamiq Mockups 2.2.10:** Utilizado para el desarrollo de maquetado de interfaces.

4.3 Definiciones, Siglas y Abreviaciones

En el presente proyecto serán utilizadas las siguientes siglas y abreviaciones para el buen entendimiento de este documento.

- **API:** Interfaz de programación de aplicaciones.
- **TCP:** Protocolo de control de transmisión.
- **SDN:** Software Defined Network.
- **ARP:** Protocolo de resolución de direcciones.
- **Veth:** Dispositivo Ethernet Virtual.
- **HTTP:** Protocolo de Transferencia de Hipertexto.
- **POP3:** Protocolo de Oficina Postal.
- **SMTP:** Protocolo para la transferencia simple de correo electrónico.

5 ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

5.1 Alcances

El sistema actual se diferencia de otros en los siguientes ítems:

- Generación de simulaciones de forma simple y rápida sobre una red personalizada.
- Generador de script para los comandos Ping e Iperf.
- Generador de scripts para Vlans y Directorios Privados.
- Creación nativa de directorios (persistentes y temporales) y vlans sin la utilización de clases.
- Validación de campos de entrada para mitigación de errores de ingreso.
- Generador de comentarios sobre los script generados.

Alcances:

- El software si bien genera script de creación y simulación de redes este no ejecuta tales scripts, estos deben ser ejecutados en una terminal en Linux, en donde previamente deberá estar instalado “Mininet” en el ordenador o en su defecto ser ejecutados en la máquina virtual de “Mininet”.

- Mininet posee la característica de crear redes compatibles con SDN, no obstante "Mininet" no programa los controladores o configura los switches OpenFlow, el presente proyecto solo abarca una parte del estudio ("Mininet"), por ende las siguientes temáticas escapan del alcance del presente proyecto: Switch OpenFlow, Controladores OpenFlow, IVS Switch, tablas de flujos sobre un controlador, Controlador Remoto OpenFlow y OVS controller. No obstante estas temáticas estan dentro de la generación de scripts y no afecta la creación y simulación de redes generadas en "Mininet".

- La generación de scripts es generada para redes locales de computadoras.

5.2 Objetivo del software

En esta sección se presentaras los objetivos generales y específicos del proyecto de software.

5.2.1 Objetivo General

El objetivo principal de este software es apoyar la creación y simulación de redes virtuales emuladas por software, lo anterior se logra mediante la generación de scripts basados en la API escrita en Python de Mininet.

5.2.2 Objetivos Específicos

- El software apoyara la creación de redes personalizadas definidas por el usuario a través de una Interfaz Gráfica.

.

- El usuario podrá crear una colección de host, switches, enlaces y controladores con valores internos por defecto o bien personalizables por el usuario.

- EL usuario podrá generar scripts de redes compatibles con Mininet a partir de una red generada por el mismo mediante la interfaz gráfica.

- Generación de scripts de simulaciones acotadas sobre los nodos de la red.

- Generación de archivos de textos que almacenaran el resultado de las simulaciones efectuadas.

5.3 Descripción Global del Producto

5.3.1 Interfaz De Hardware

Si bien el software no interactúa directamente con el Hadware, los scripts generados si lo hacen, por eso hay que tener en cuenta que la configuración de los nodos virtuales no ocupen los siguientes puertos:

- Puertos del 0 al 1024, estos puertos no pueden ser utilizados debido a que son ocupados por el sistema operativo (POP3/SMTP entre otros).
- Puerto 6633, este puerto es utilizado por el controlador OpenFlow, utilizarlo podría causar conflictos en la simulación.

- Puerto 5001, puerto utilizado normalmente por la herramienta Iperf, Si bien Iperf permite la posibilidad de cambiar el puerto de escucha es recomendable no utilizarlo.

5.3.2 Interfaz Software

El generador de scripts interactúa con los siguientes softwares:

Nombre	Abreviación	Versión	Fuente
Gedit	Gedit	3.14.2 o inferior	https://wiki.gnome.org/Apps/Gedit/
X Term	Xterm	Patch #312 o inferior	http://invisible-island.net/xterm/
Gnome Term	Gterm	3.14.2 o inferior	http://sourceforge.net/projects/gterm/

Tabla 79. Interfaces de Software.

5.4 Requerimientos Específicos

5.4.1 Requerimientos Funcionales del sistema

ID	Nombre	Descripción
RF01	Generar Host	El usuario genera un host que tendrá valores por defectos predefinidos.
RF02	Modificar Host	El usuario modifica un host y puede cambiar los valores predefinidos de este.
RF03	Eliminar Host	El usuario puede eliminar los host generados. Obs: Al eliminar un Host se eliminaran todos los enlaces conectados a otros nodos.
RF04	Generar Switch	El usuario genera un switch que tendrá valores por defecto.
RF05	Modificar Switch	El usuario es capaz de modificar un switch y sus valores por defecto.
RF06	Eliminar Switch	El usuario puede eliminar un switch ingresado en el sistema
RF07	Generar Controlador	El usuario genera un controlador que tendrá valores por defectos predefinidos.
RF08	Modificar Controlador	El usuario es capaz de modificar un controlador y sus valores por defectos.
RF09	Eliminar Controlador	El usuario puede eliminar un controlador generado. Obs: Al eliminar un controlador todos sus enlaces serán eliminados.
RF10	Generar Enlace	El usuario es capaz de generar enlaces que une a los distintos nodos de la red. Obs: Un switch del tipo Legacy no puede conectarse a un controlador, un Host no puede conectarse directamente a un controlador y los controladores no pueden conectarse entre si.
RF11	Modificar Enlace	El usuario es capaz de personalizar los valores por defecto de los enlaces.
RF12	Eliminar Enlace	El usuario es capaz de eliminar un enlace generado.
RF13	Generar Red	El usuario puede generar un script en Python en base a la red creada.
RF14	Generar Simulación	El usuario es capaz de generar simulaciones acotadas sobre los hosts.

Tabla 80. Requisitos Funcionales

5.4.2 Interfaces externas de entrada

ID	Nombre del ítem	Detalle de Datos contenidos en ítem
IE1	Datos del Host	Nombre,IP,MAC, Mascara, CPU
IE2	Datos del Switch	Nombre, Puerto, Tipo (DEFAULT, OVSKERNELSWITCH, IVSSWITCH, LEGACYSWITCH).
IE3	Datos del Controlador	Nombre, Puerto, IP, Tipo (DEFAULT, REMOTE COTROLLER, OVSCONTROLLER).
IE4	Datos del Enlace	Ancho de banda, Retraso, perdida(%), tamaño Queue, Jitter, Aceleración.
IE5	Datos Directorio	Nombre, Ruta, Tipo (Temporal, Persistente).
IE6	Datos Vlan	Identificador.
IE7	Datos Iperf Servidor	Tamaño de la Ventana, Puerto, Formato(Kbits, Mbits, KBytes, MBytes)
IE8	Datos Iperf Cliente	Puerto, Tamaño de la Ventana, Formato (Kbytes, Mbytes), Cantidad, Tiempo.

Tabla 81. Interfaces Externas de Entrada

5.4.3 Interfaces externas de Salida

ID	Nombre del ítem	Detalle de Datos contenidos en ítem	Medio Salida
IS1	Script Host	Nombre,IP,MAC, Mascara, CPU , Script Host	Archivo Python Archivo TXT Pantalla
IS2	Datos del Switch	Nombre, Puerto, Tipo (DEFAULT, OVSKERNELSWITCH, IVSSWITCH, LEGACYSWITCH), Script Switch	Archivo Python Archivo TXT Pantalla
IS3	Datos del Controlador	Nombre, Puerto, IP , Tipo(DEFAULT,REMOTE COTROLLER, OVSCONTROLLER), Script Controlador	Archivo Python Archivo TXT Pantalla
IS4	Datos del Enlace	Ancho de banda, Retraso, perdida(%), tamaño Queue, Jitter, Aceleración, Script Enlace	Archivo Python Archivo TXT Pantalla
IS5	Datos Directorio	Nombre, Ruta, Tipo (Temporal, Persistente), Script Directorio	Archivo Python Archivo TXT Pantalla
IS6	Datos Vlan	Identificador, Script Vlan	Archivo Python Archivo TXT Pantalla
IS7	Datos Iperf Servidor	Tamaño de la Ventana, Puerto, Formato(Kbits, Mbits, KBytes, MBytes), Script Servidor	Archivo Python Archivo TXT Pantalla
IS8	Datos Iperf Cliente	Puerto, Tamaño de la Ventana, Formato (Kbytes, Mbytes), Cantidad, Tiempo, Script Cliente	Archivo Python Archivo TXT Pantalla.

ID	Nombre del ítem	Detalle de Datos contenidos en ítem	Medio Salida
IS9	Error IP	"IP INVALIDA"	Pantalla
IS10	Error MAC	"MAC INVALIDA"	Pantalla
IS11	Error Vlan	"Ingrese solo números"	Pantalla

Tabla 82. Interfaces externas de Salida

6 ANÁLISIS

6.1 Diagrama de casos de uso

6.1.1 Actores

Actor	Rol	Nivel de Conocimientos	Nivel privilegios y Funciones
Usuario	Usuario	El nivel de conocimientos que debe tener el usuario del sistema es de un nivel básico de uso de computadoras y poseer un nivel de conocimiento medio de generación de redes de ordenadores.	El usuario podrá acceder a todas las funcionalidades descritas en el punto 5.4.1 del presente informe

Tabla 83. Descripción Actores Casos de Uso

6.1.2 Diagramas de Casos de Uso

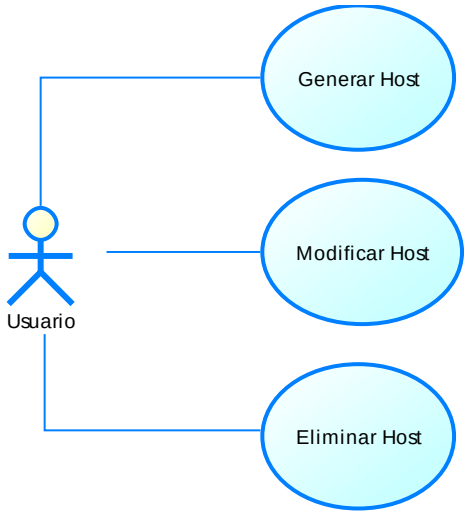


Figura 27. Interacción con Host

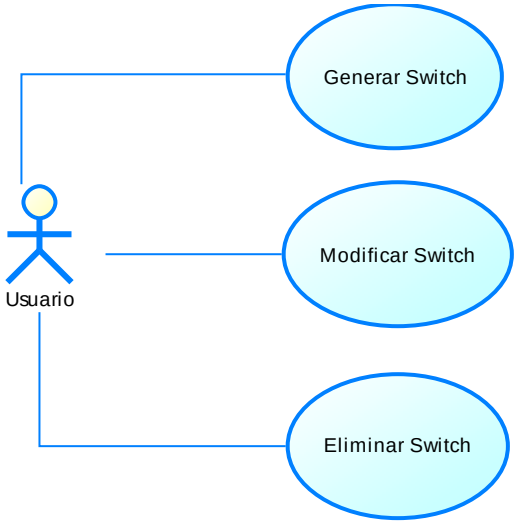


Figura 28. Interacción con Switchs

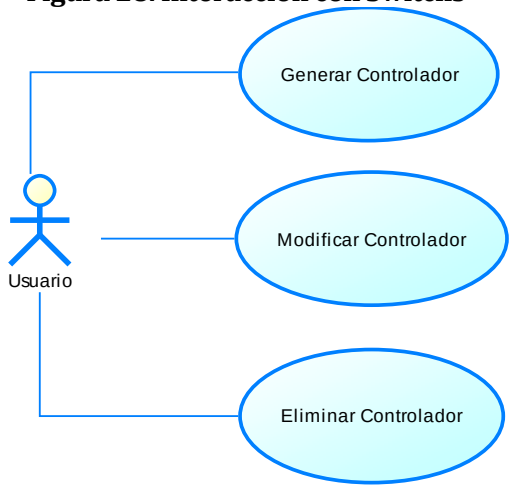


Figura 29. Interacción Controlador

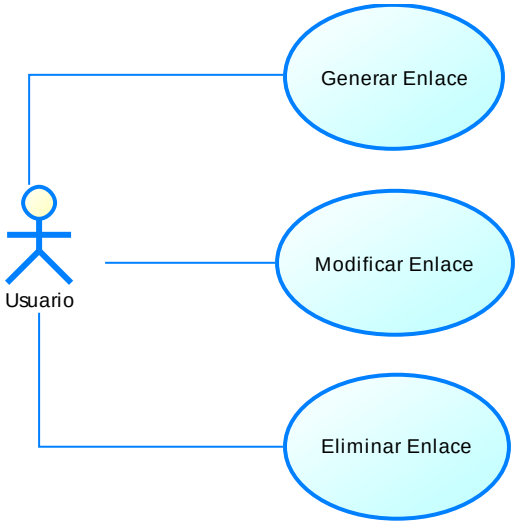


Figura 30. Interacción Enlaces

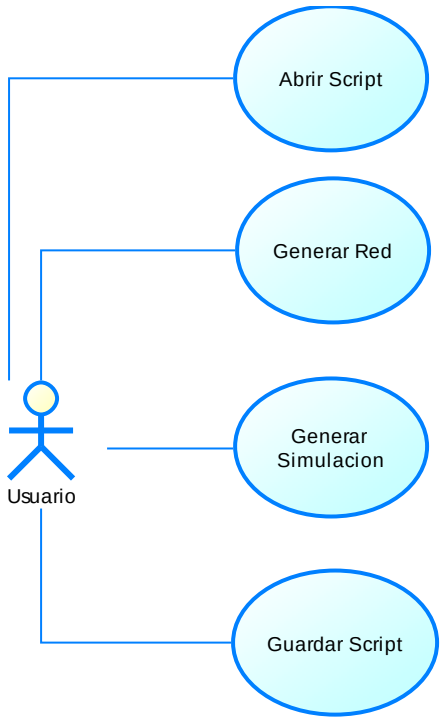


Figura 31. Interacción General

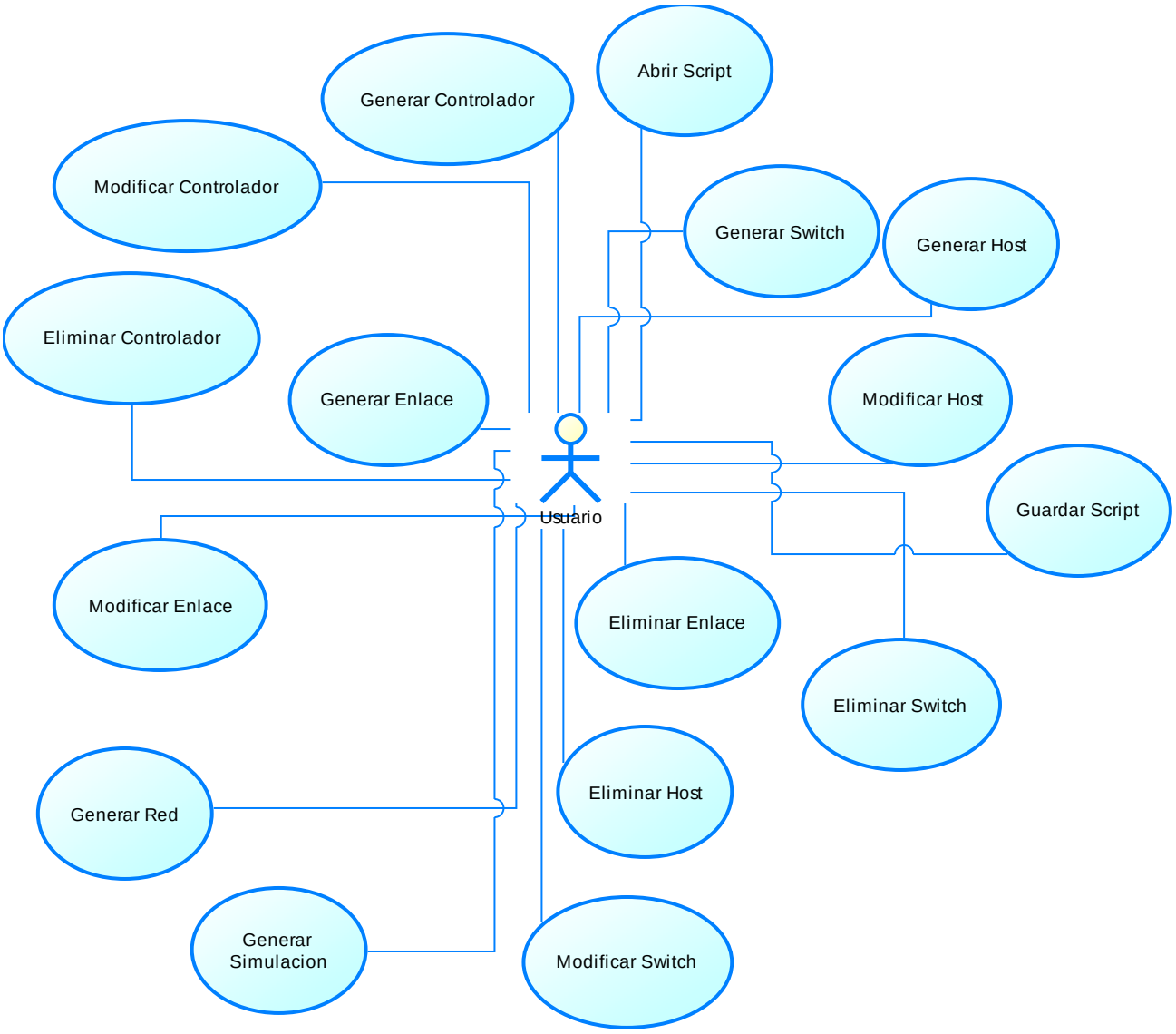


Figura 32. Diagrama de Casos de uso General

6.1.3 Especificación de los Casos de Uso

El siguiente ítem especificara los casos de uso descritos en el punto 6.1.2

6.1.3.1 Caso de Uso: <Generar Host>

- **Descripción:** El actor puede generar un host en el sistema por medio de la interfaz gráfica.
- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón host debe estar seleccionado y la generación del host debe ser dentro del área de dibujo.
- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem host y genera el host en el panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón host, si es así se procede a generar el host en la posición indicada por el mouse del usuario, el host se generara con un nombre, ip y mascara por defecto

Tabla 84. Flujo de Eventos Básicos <Generar Host>.

- **Flujo de Eventos Alternativos:**

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem host e intenta generar un host en el panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón host, como este no esta seleccionado no se genera el host solicitado.

Tabla 85. Flujos de Eventos Alternativos 1 <Generar Host>

Al actor	El sistema
3 El usuario selecciona en la barra de herramientas el ítem host e intenta generar un host fuera del panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón host, pero el panel de dibujo al no recibir acciones del mouse no genera el host solicitado por el usuario.

Tabla 86. Flujos de Eventos Alternativos 2 <Generar Host>

- **Post-Condiciones:** Se almacena el host dentro del sistema con sus datos por defecto.

6.1.3.2 Caso de Uso: <Modificar Host>

- **Descripción:** El actor puede modificar un host ingresado en el sistema por medio de la interfaz gráfica.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón modificar debe estar seleccionado y debe existir al menos un host en el sistema.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el host a modificar.	**4** El sistema verifica si esta seleccionado el botón modificar y si el usuario está posicionado sobre un host.
5 El usuario presiona el botón derecho del ratón.	**6** El sistema despliega un menú con la opción de editar el host.
7 El usuario abre el panel de edición del host.	**8** El sistema despliega el panel de edición para el host correspondiente.
9 El usuario edita los campos del host	**10** El sistema valida si los campos están correctamente ingresados

Tabla 87. Flujo de Eventos Básicos <Modificar Host>.

- **Flujo de Eventos Alternativo**

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el host a modificar.	**4** El sistema verifica si esta seleccionado el botón modificar, como este no esta seleccionado no hay acciones para el panel modificar host.

Tabla 88. Flujos de Eventos Alternativos <Modificar Host>

- **Post-Condiciones:** Se almacena el host dentro del sistema con los datos modificados.

6.1.3.3 Caso de Uso: <Eliminar Host>

- **Descripción:** El actor puede eliminar un host ingresado en el sistema.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón eliminar debe estar seleccionado y debe existir al menos un host en el sistema.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem eliminar y se posiciona con el mouse sobre el host a eliminar.	**4** El sistema verifica si esta seleccionado el botón eliminar y si el usuario está posicionado sobre un host.
5 El usuario elimina el host	**6** El sistema elimina el host del panel de dibujo y sus datos dentro del sistema

Tabla 89. Flujo de Eventos Básicos <Eliminar Host>

- **Flujo de Eventos Alternativos**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem eliminar y se posiciona con el mouse sobre el host a modificar.	**4** El sistema verifica si esta seleccionado el botón eliminar como este no esta seleccionado no hay acciones para eliminar el host

Tabla 90. Flujo de Eventos Alternativos <Eliminar Host>.

- **Post-Condiciones:** Se elimina los datos dentro del sistema del host seleccionado y además se eliminan todos sus enlaces.

6.1.3.4 Caso de Uso: <Generar Switch>

- **Descripción:** El actor puede generar un switch en el sistema por medio de la interfaz gráfica.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón switch debe estar seleccionado y la generación del switch debe ser dentro del área de dibujo.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el switch y genera el switch en el panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón switch, si es así se procede a generar el switch en la posición indicada por el mouse del usuario, el switch se generara con un nombre, puerto y tipo por defecto

Tabla 91. Flujos de Eventos Básicos <Generar Switch>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el switch e intenta generar un switch en el panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón switch, como este no esta seleccionado no se genera el switch solicitado.

Tabla 92. Flujos de Eventos Alternativos 1 <Generar Switch>

Al actor	El sistema
3 El usuario selecciona en la barra de herramientas el ítem switch e intenta generar un switch fuera del panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón switch, pero el panel de dibujo a no recibir acciones del mouse este no genera el switch solicitado por el usuario.

Tabla 93. Flujos de Eventos Alternativos 2 <Generar Switch>

- **Post-Condiciones:** Se almacena el switch dentro del sistema con sus datos por defecto.

6.1.3.5 Caso de Uso: <Modificar Switch>

- **Descripción:** El actor puede modificar un switch ingresado en el sistema por medio de la interfaz gráfica.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón modificar debe estar seleccionado y debe existir al menos un switch en el sistema.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el switch a modificar.	**4** El sistema verifica si esta seleccionado el botón modificar y si el usuario está posicionado sobre un switch.
5 El usuario presiona el botón derecho del ratón	**6** El sistema despliega un menú con la opción de editar el switch.
7 El usuario abre el panel de edición del switch.	**8** El sistema despliega el panel de edición para el switch correspondiente.
9 El usuario edita los campos del switch.	**10** El sistema valida si los campos están correctamente ingresados

Tabla 94. Flujo de Eventos Básicos <Modificar Switch>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el switch a modificar.	**4** El sistema verifica si esta seleccionado el botón modificar, como este no esta seleccionado no hay acciones para el panel modificar switch.

Tabla 95. Flujo de Eventos Alternativos < Modificar Switch>

- **Post-Condiciones:** Se almacena el switch dentro de sistema con los datos modificados.

6.1.3.6 Caso de Uso: <Eliminar Switch>

- **Descripción:** El actor puede eliminar un switch ingresado en el sistema.
- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón eliminar debe estar seleccionado y debe existir al menos un switch en el sistema.
- **Flujo de Eventos Básicos**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem eliminar y se posiciona con el mouse sobre el switch a eliminar.	**4** El sistema verifica si esta seleccionado el botón eliminar y si el usuario está posicionado sobre un switch.
5 El usuario elimina el switch.	**6** El sistema elimina el switch del panel de dibujo y sus datos dentro del sistema

Tabla 96. Flujo de Eventos Básicos <Eliminar switch>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem eliminar y se posiciona con el mouse sobre el switch a eliminar.	**4** El sistema verifica si esta seleccionado el botón eliminar como este no esta seleccionado no hay acciones para eliminar el switch.

Tabla 97. Flujo de Eventos Alternativos <Eliminar switch>.

- **Post-Condiciones:** Se elimina los datos dentro del sistema del switch seleccionado y además se eliminan todos sus enlaces.

6.1.3.7 Caso de Uso: <Generar Controlador>

- **Descripción:** El actor puede generar un controlador en el sistema por medio de la interfaz gráfica.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón Controlador debe estar seleccionado, la generación del controlador debe ser dentro del área de dibujo.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem controlador y genera el controlador en el panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón controlador, si es así se procede a generar el controlador en la posición indicada por el mouse del usuario, el controlador se generara con un nombre, ip, tipo y puerto por defecto.

Tabla 98. Flujo de Eventos Básico <Generar Controlador>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem controlador e intenta generar un controlador en el panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón controlador como este no esta seleccionado no se genera el controlador solicitado.

Tabla 99. Flujos de Eventos Alternativos 1 <Generar Controlador >

Al actor	El sistema
3 El usuario selecciona en la barra de herramientas el ítem controlador e intenta generar un controlador fuera del panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón controlador, pero el panel de dibujo a no recibir acciones del mouse no genera el controlador solicitado por el usuario.

Tabla 100. Flujos de Eventos Alternativos 2 <Generar Controlador >

- **Post-Condiciones:** Se almacena el controlador dentro del sistema con sus datos por defecto.

6.1.3.8 Caso de Uso: <Modificar Controlador>

- **Descripción:** El actor puede modificar un controlador ingresado en el sistema por medio de la interfaz gráfica.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón modificar debe estar seleccionado y debe existir al menos un controlador en el sistema.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el controlador a modificar.	**4** El sistema verifica si esta seleccionado el botón modificar y si el usuario está posicionado sobre un controlador.
5 El usuario presiona el botón derecho del ratón	**6** El sistema despliega un menú con la opción de editar el controlador.
7 El usuario abre el panel de edición del controlador.	**8** El sistema despliega el panel de edición para el controlador correspondiente.
9 El usuario edita los campos del controlador.	**10** El sistema valida si los campos están correctamente ingresados

Tabla 101. Flujo de Eventos Básicos <Modificar Controlador >.

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el controlador a modificar.	**4** El sistema verifica si esta seleccionado el botón modificar como este no esta seleccionado no hay acciones para el panel modificar controlador.

Tabla 102. Flujos de Eventos Alternativos <Modificar Host>

- **Post-Condiciones:** Se almacena el controlador dentro del sistema con los datos modificados.

6.1.3.9 Caso de Uso: <Eliminar Controlador>

- **Descripción:** El actor puede eliminar un controlador ingresado en el sistema.
- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón eliminar debe estar seleccionado y debe existir al menos un controlador en el sistema.
- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem eliminar y se posiciona con el mouse sobre el controlador a eliminar.	**4** El sistema verifica si esta seleccionado el botón eliminar y si el usuario está posicionado sobre un controlador.
5 El usuario elimina el controlador	**6** El sistema elimina el controlador del panel de dibujo y sus datos dentro del sistema

Tabla 103.Flujo de Eventos Básicos <Eliminar controlador>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem eliminar y se posiciona con el mouse sobre el controlador a eliminar.	**4** El sistema verifica si esta seleccionado el botón eliminar como este no esta seleccionado no hay acciones para eliminar el controlador.

Tabla 104. Flujo de Eventos Alternativos <Eliminar controlador>.

- **Post-Condiciones:** Se elimina los datos dentro del sistema del controlador seleccionado y además se eliminan todos sus enlaces.

6.1.3.10 Caso de Uso: <Generar Enlace>

- **Descripción:** El actor puede generar un enlace en el sistema por medio de la interfaz gráfica.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón enlace debe estar seleccionado, la generación del enlace debe ser dentro del área de dibujo.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem enlace y genera el enlace sobre los nodos seleccionados en el panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón enlace, si es así se procede a generar el enlace en los nodos indicados por el usuario.

Tabla 105. Flujo de Eventos Básicos <Generar Enlace>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem enlace e intenta generar un enlace en el panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón enlace como este no esta seleccionado no se genera el enlace solicitado.

Tabla 106. Flujos de Eventos Alternativos 1 <Generar Enlace >

Al actor	El sistema
3 El usuario selecciona en la barra de herramientas el ítem enlace e intenta generar un enlace fuera del panel de dibujo.	**4** El sistema verifica si esta seleccionado el botón enlace, pero el panel de dibujo a no recibir acciones del mouse no genera el enlace solicitado por el usuario.

Tabla 107. Flujos de Eventos Alternativos 2 <Generar Enlace >

Al actor	El sistema
3 El usuario selecciona en la barra de herramientas el ítem enlace e intenta generar un enlace entre dos nodos que ya poseen un enlace.	**4** El sistema verifica si esta seleccionado el botón enlace, al ver que los nodos ya poseen un enlace este no se generara.

Tabla 108. Flujos de Eventos Alternativos 3 <Generar Enlace >

- **Post-Condiciones:** Se almacena el enlace dentro del sistema con sus datos por defecto.

6.1.3.11 Caso de Uso: <Modificar Enlace>

- **Descripción:** El actor puede modificar un enlace ingresado en el sistema por medio de la interfaz gráfica.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón modificar debe estar seleccionado, debe existir al menos un enlace en el sistema.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el switch que contiene los enlaces a modificar.	**4** El sistema verifica si esta seleccionado el botón modificar y si el usuario está posicionado sobre un switch.
5 El usuario presiona el botón derecho del ratón.	**6** El sistema despliega un menú con la opción de editar enlace.
7 El usuario abre el panel de edición del enlace seleccionado.	**8** El sistema despliega el panel de edición para el enlace correspondiente.
9 El usuario edita los campos del enlace	**10** El sistema valida si los campos están correctamente ingresados.

Tabla 109. Flujo de Eventos Básicos <Modificar enlace>.

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el enlace a modificar.	**4** El sistema verifica si esta seleccionado el botón modificar como este no esta seleccionado no hay acciones para el panel modificar los enlaces.

Tabla 110. Flujos de Eventos Alternativos 1 <Modificar Enlace>

Al actor	El sistema
9 El usuario ingresa mal los campos de un enlace.	**4** El sistema verifica si los campos están bien validados, como estos poseen errores, despliega un mensaje de error y no genera el enlace solicitado.

Tabla 111. Flujos de Eventos Alternativos 2 <Modificar Enlace>

- **Post-Condiciones:** Se almacena el enlace dentro del sistema con los datos modificados.

6.1.3.12 Caso de Uso: <Eliminar Enlace>

- **Descripción:** El actor puede eliminar un enlace ingresado en el sistema.
- **Pre-Condiciones:** El área de dibujo debe estar abierta, el botón eliminar debe estar seleccionado y debe existir al menos un enlace en el sistema.
- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y abre el panel de dibujo.	**2** Se carga el panel de dibujo en el sistema y se despliega la barra de herramientas.
3 El usuario selecciona en la barra de herramientas el ítem eliminar y se posiciona con el mouse sobre el switch y selecciona el enlace a eliminar.	**4** El sistema verifica si esta seleccionado el botón eliminar y si el usuario está posicionado sobre un enlace.
5 El usuario elimina el enlace.	**6** El sistema elimina el enlace del panel de dibujo y sus datos dentro del sistema

Tabla 112. Flujo de Eventos Básicos <Eliminar Enlace>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem modificar y se posiciona con el mouse sobre el enlace a eliminar.	**4** El sistema verifica si esta seleccionado el botón modificar como este no esta seleccionado no hay acciones para eliminar el enlace.

Tabla 113. Flujo de Eventos Alternativos <Eliminar Enlace>.

Post-Condiciones: Se elimina los datos del enlance dentro del sistema.

6.1.3.13 Caso de Uso: <Generar Red>

- **Descripción:** El usuario puede generar una script de red mediante los nodos generados a través de la interfaz gráfica.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, debe existir al menos un nodo en el sistema.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Una vez generados los nodos, el usuario por medio del menú superior presiona el botón generar script	**2** El sistema carga todos los nodos, atributos y genera un script en python compatible con "Mininet.

Tabla 114. Flujo de Eventos Básicos <Generar Script>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
1 El usuario intenta generar un script sin abrir el panel de dibujo	**2** Verifica si el panel de dibujo esta abierto, como no lo esta el sistema envía una mensaje de error al usuario.

Tabla 115. Flujo de Eventos Alternativos 1 <Generar Script>

Al actor	El sistema
1 El usuario intenta generar un script sin haber ingresado nodos en el panel de dibujo	**2** Verifica si el panel de dibujo esta abierto y verifica si existe al menos un nodo en el panel, como no hay nodos el sistema envía un mensaje de error al usuario.

Tabla 116. Flujo de Eventos Alternativos 2<Generar Script>

- **Post-Condiciones:** Se genera el script en python compatible con "Mininet"

6.1.3.14 Caso de Uso: <Guardar Script>

- **Descripción:** El usuario puede guardar una red creada a través de la interfaz grafica.
- **Pre-Condiciones:** El área de dibujo debe estar abierta y debe existir al menos un nodo en el sistema.
- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Una vez gerada la red en el panel de dibujo el usuario se dirige al menú superior y presiona guardar script	**2** El sistema verifica que haya al menos un nodo en el panel de dibujo, genera el script y un archivo txt en donde se almacenara la red.

Tabla 117. Flujo de Eventos Básicos <Guardar Script>

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
1 El usuario intenta guardar la red sin ningún nodo generado.	**4** El sistema verifica que haya al menos un nodo en el panel de dibujo, como no hay el sistema envía un mensaje de error al usuario.

Tabla 118. Flujos de Eventos Alternativos <Guardar Script>

- **Post-Condiciones:** Se almacena la red generada en un archivo txt.

6.1.3.15 Caso de Uso: <Abrir Script>

- **Descripción:** El usuario puede abrir un script generado con anterioridad.
- **Pre-Condiciones:** Debe existir al menus un archivo.
- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica y y selecciona el item abrir.	**2** Despliega un panel para buscar el archivo
3 El usuario selecciona el archivo de red.	**4** El sistema despliega la red del archivo.

Tabla 119. Flujo de Eventos Básicos <Abrir Script>.

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no puede seleccionar el archivo ya que no existe	**4** El sistema no realiza ninguna accion.

Tabla 120. Flujos de Eventos Alternativos 1 <Abrir Script>

- **Post-Condiciones:** Se carga en el sistema la red almacenada en el achivo.

6.1.3.16 Caso de Uso: <Generar Simulación>

- **Descripción:** El actor puede generar simulaciones acotadas sobre los nodos del panel de dibujo.

- **Pre-Condiciones:** El área de dibujo debe estar abierta, el area de simulaciones debe estar abierta y debe existir al menos un nodo en el sistema.

- **Flujo de Eventos Básicos:**

Al actor	El sistema
1 Entra a la interfaz gráfica, abre el panel de dibujo, abre el panel de simulaciones y genera una red sobre el panel de dibujo.	**2** Se carga el panel de dibujo y el panel de simulaciones y se genera la red en el sistema
3 El usuario selecciona en la barra de herramientas una de las simulaciones y selecciona uno de los nodos.	**4** El sistema verifica si esta seleccionado el botón modificar y si el usuario está posicionado sobre un host.
5 El usuario genera la simulacion.	**6** El sistema genera la simulacion en el panel de simulaciones.

Tabla 121. Flujo de Eventos Básicos <Generar Simulacion>.

- **Flujo de Eventos Alternativo**:

Al actor	El sistema
3 El usuario no selecciona en la barra de herramientas el ítem modificar o algun item de simulacion y se posiciona con el mouse sobre el host a generar la simulacion.	**4** El sistema verifica si esta seleccionado el botón modificar como este no esta seleccionado o no hay ningun item de simulacion seleccionado no hay acciones para una simulacion.

Tabla 122. Flujos de Eventos Alternativos 1 <Generar Simulacion>

Post-Condiciones: Se generar las simulaciones en el panel de simulacion del sistema.

6.2 Modelamiento de datos

6.2.1 Diagrama de Clases

6.2.1.1 Entidades

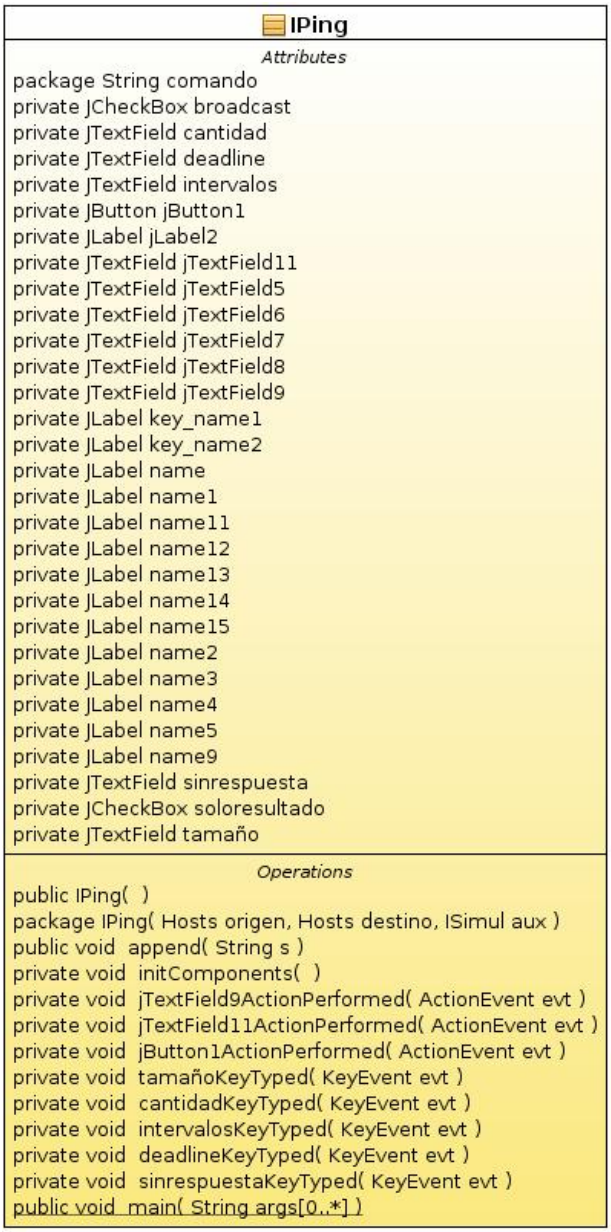


Figura 33. Clase IPing

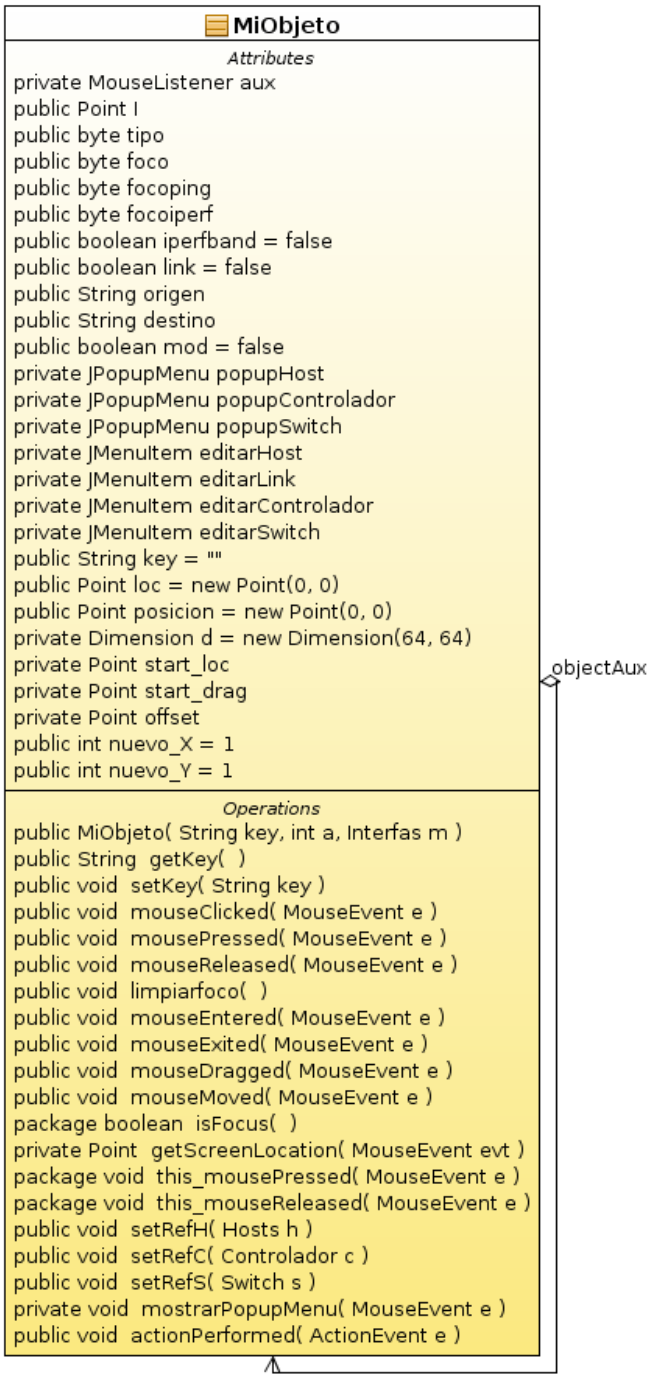


Figura 34. Clase MiObjeto

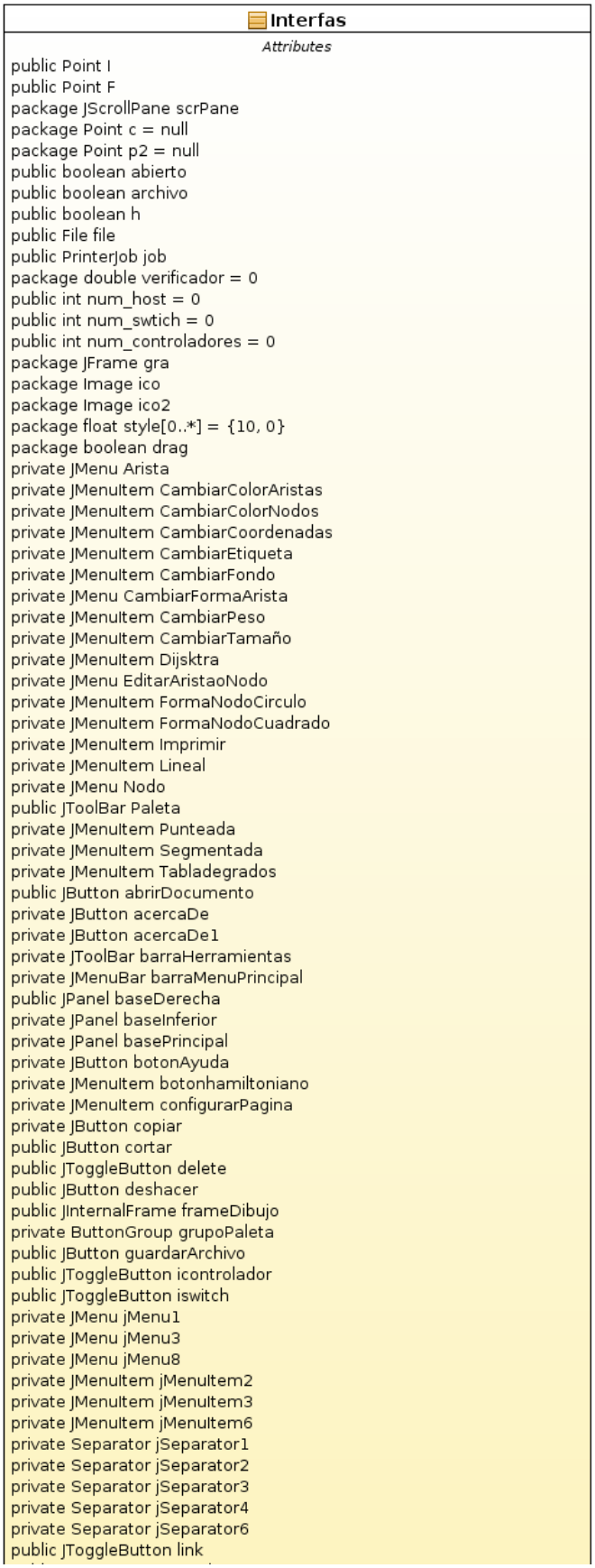


Figura 35. Clase Interfas Parte I

Figura 36. Clase Interfas Parte II

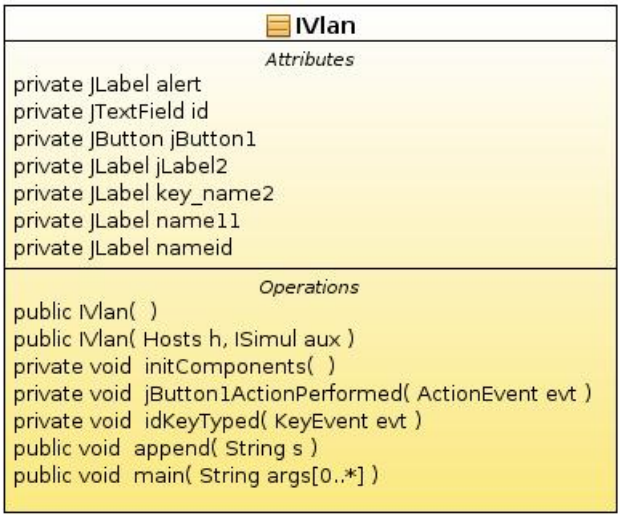


Figura 37. Clase IVlan

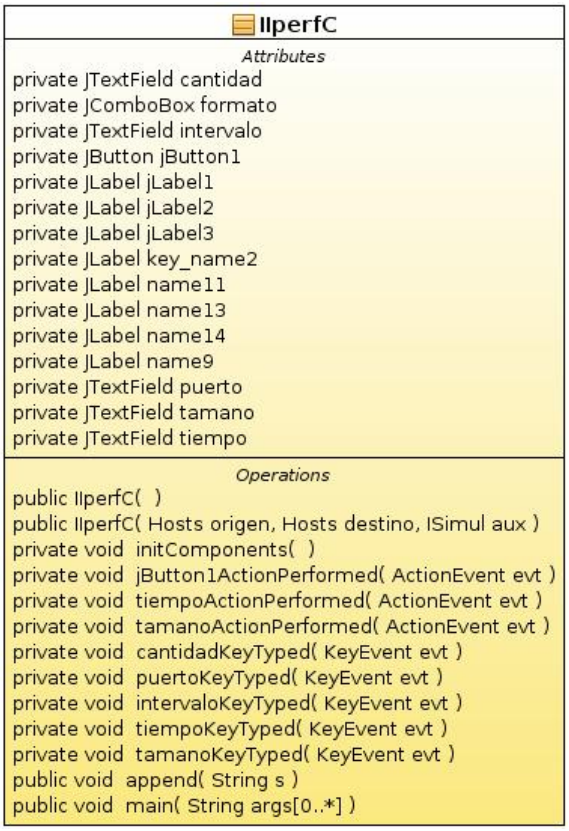


Figura 38. Clase IIperfC

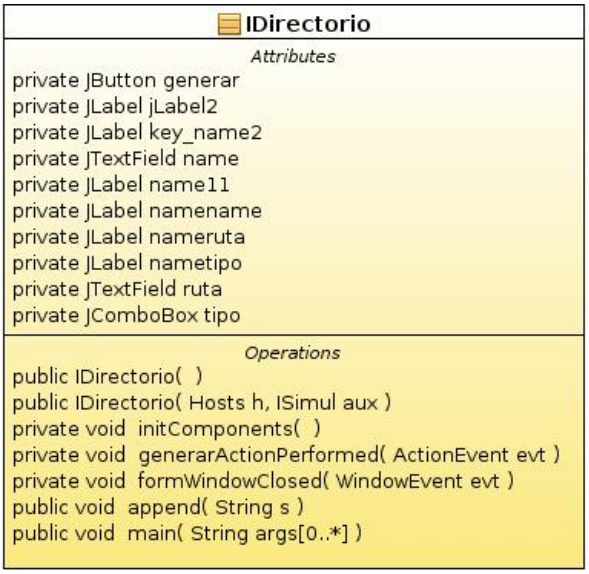


Figura 39. Clase IDirectorio

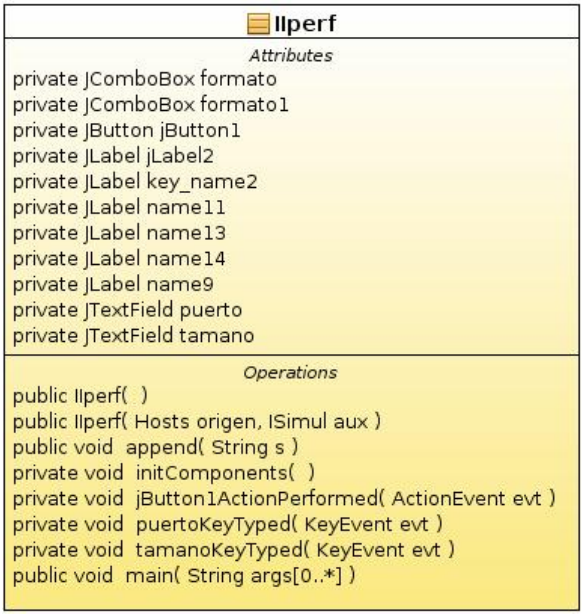


Figura 40. Clase IIperf

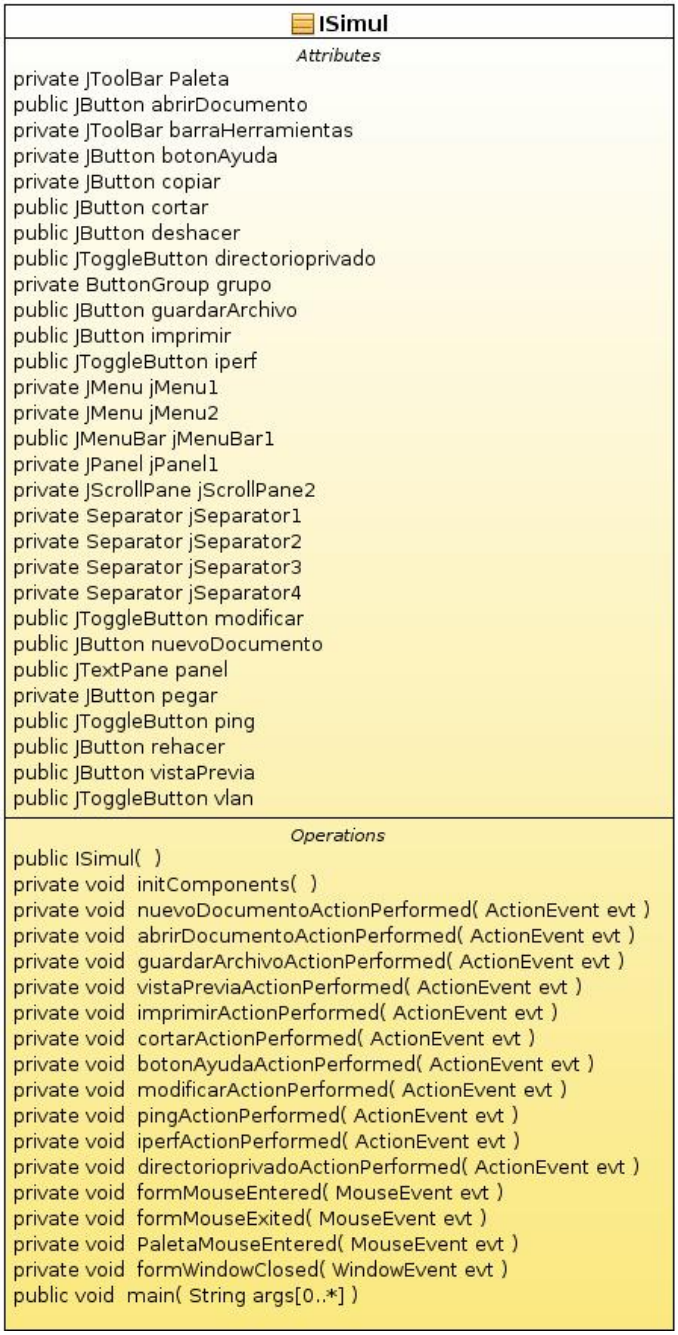


Figura 41. Clase ISimul

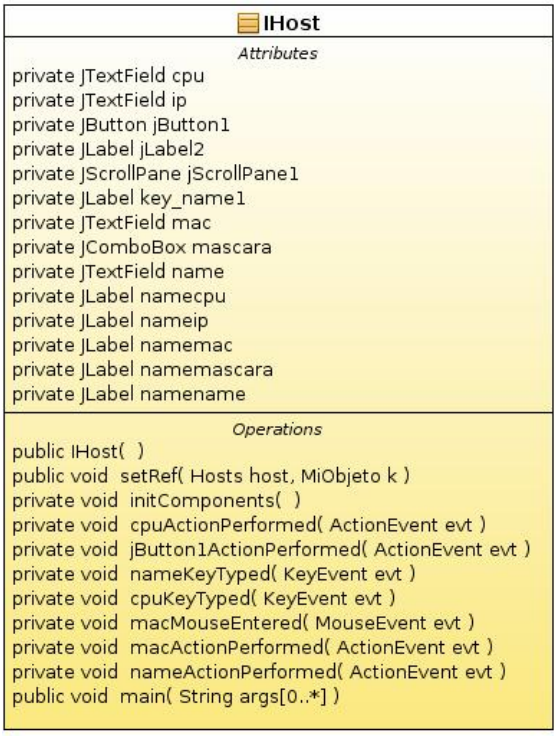


Figura 42. Clase IHost

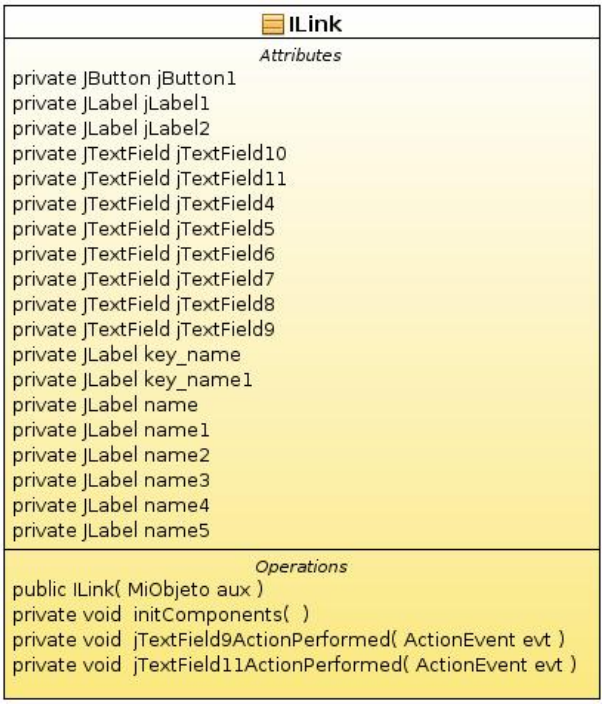


Figura 43. Clase ILink

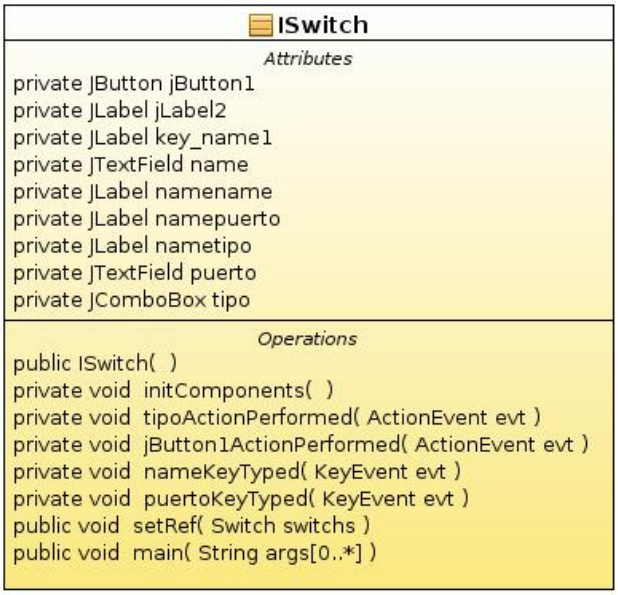


Figura 44. Clase ISwitch

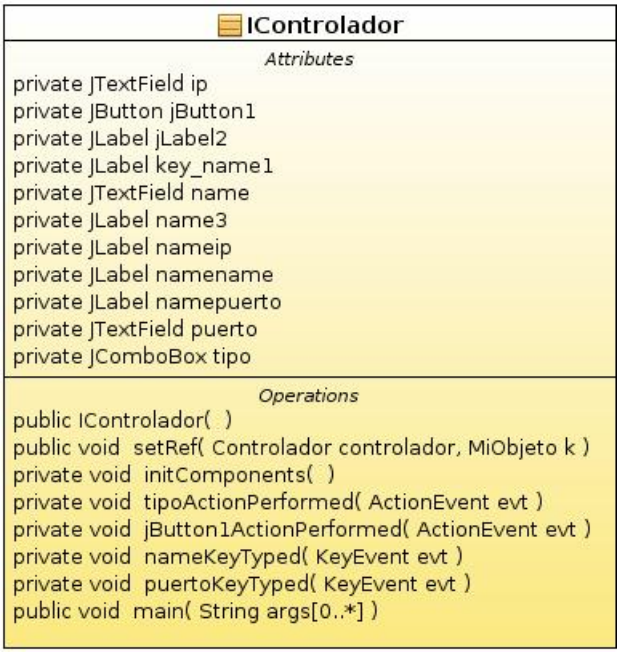


Figura 45. Clase IControlador

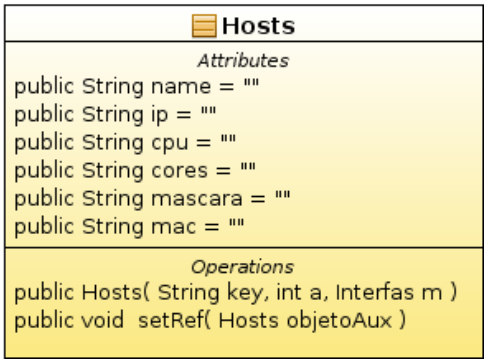


Figura 46. Clase Hosts

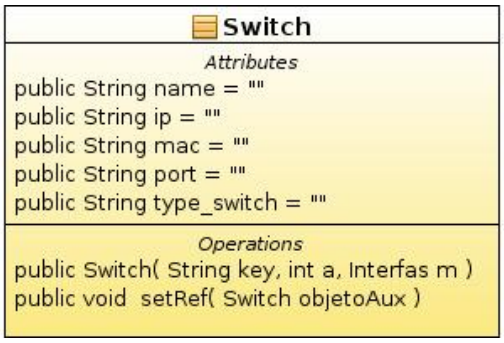


Figura 47. Clase Switch

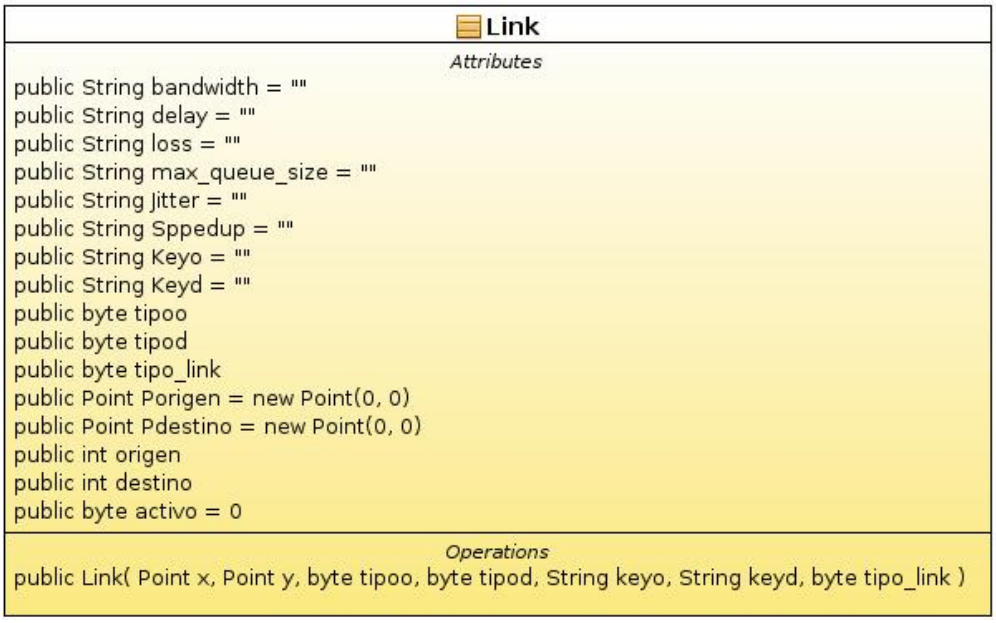


Figura 48. Clase Link

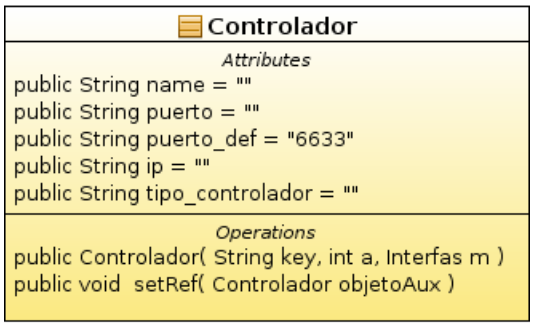


Figura 49. Clase Controlador

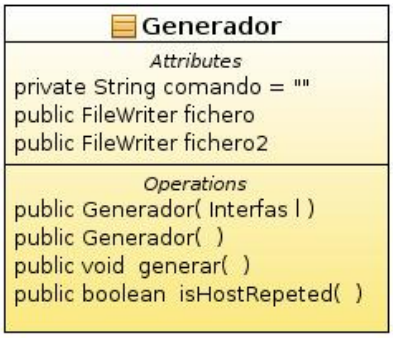


Figura 50. Clase Generador

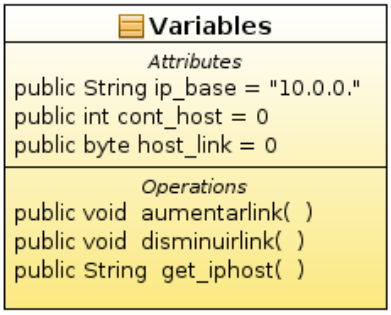


Figura 51. Clase Variables

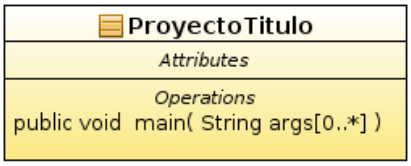


Figura 52. Clase ProyectoTitulo

6.2.1.2 Diagrama de Clases

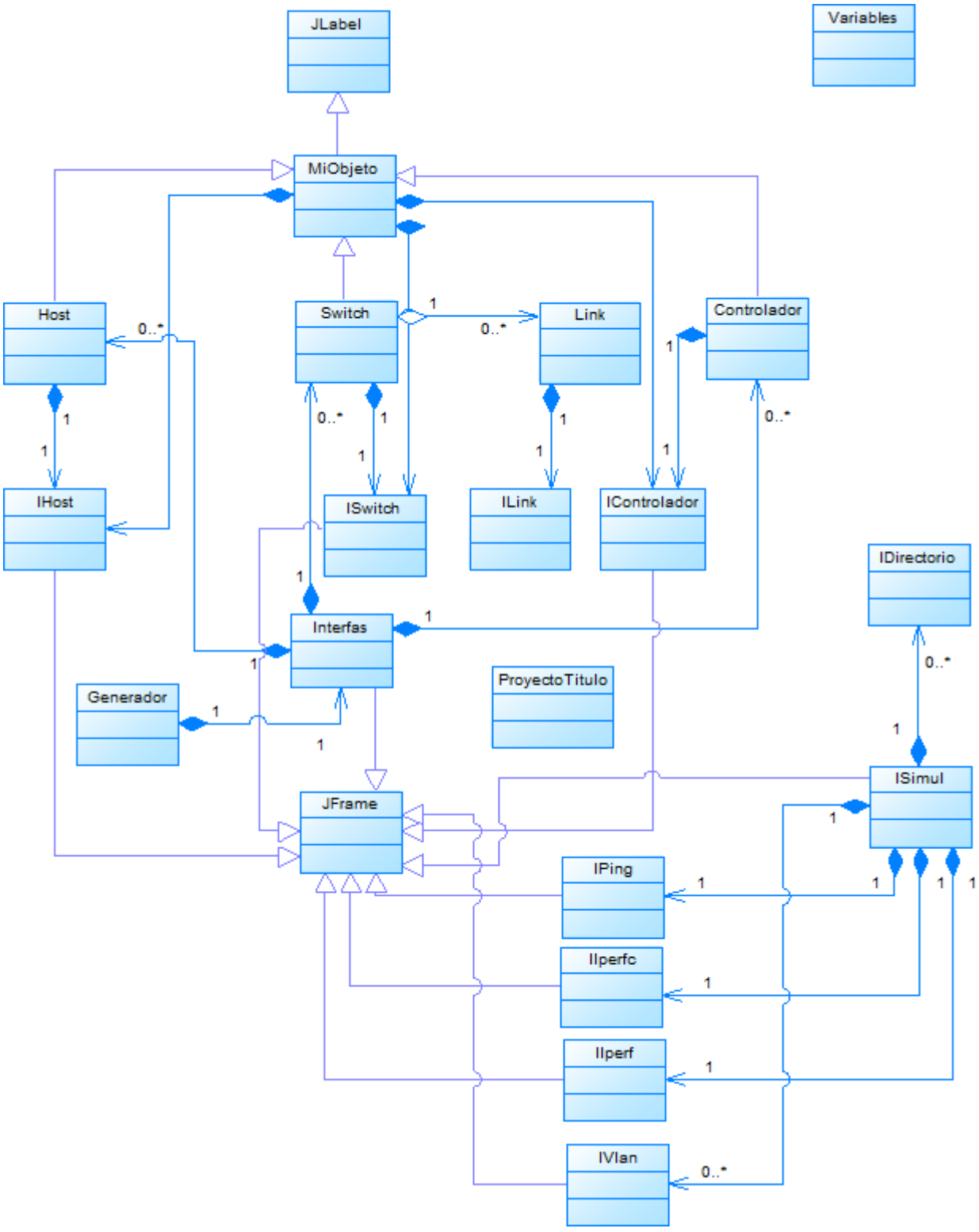


Figura 53. Diagrama de Clases

7 DISEÑO

7.1 Diseño interfaz y navegación

El siguiente ítem explicara las diferentes pantallas que se poseerá el sistema.

7.1.1 Esquema especificación de Interfaces

Figura 54. Interfaz Principal

ID	Descripción
1	Barra superior 1. Incluye los submenús: Archivo (Nueva Red, Abrir Archivo, Guardar, Guardar Como y salir) y Editar (IP Base).
2	Barra superior 2. Incluye los siguientes botones: Nuevo, Abrir, Guardar, Acerca de y Simulación.
3	Barra inferior. Cumple la función de separación del panel
4	Frame Principal: Cumple la función de contener a todos los elementos de la interfaz
5	Barra lateral. Incluye los botones: Modificar, Host, Enlace, Switch, Controlador y Eliminar
6	Barra superior Frame. Incluye el icono de la aplicación y el título del sistema

Tabla 123. Descripción de Interfaz Principa

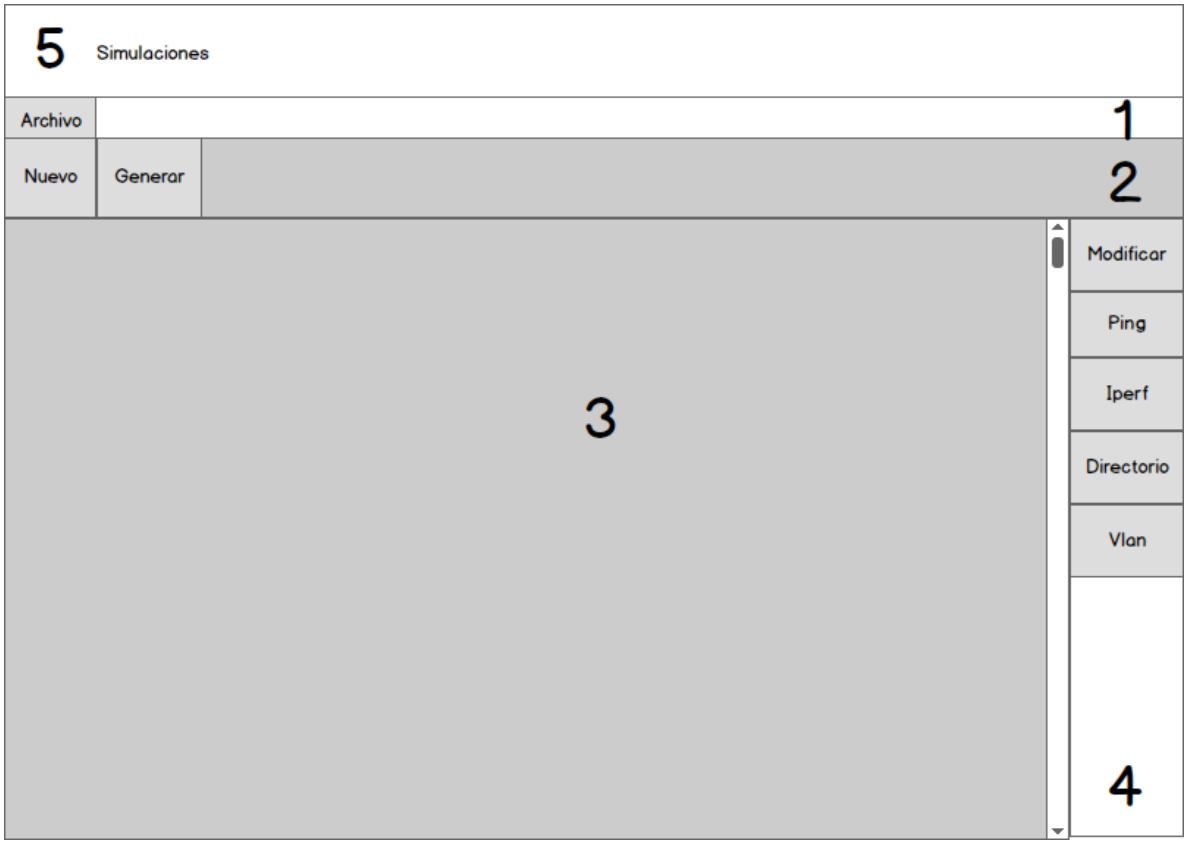


Figura 55. Interfaz Simulación.

ID	Descripción
1	Barra superior 1. Incluye los submenús: Archivo (Nueva Simulación, Guardar y salir).
2	Barra superior 2. Incluye los siguientes botones: Nuevo y Generar
3	Sección de texto scrollable : Cumple con mostrar todas las simulaciones generadas en los host
4	Barra lateral. Incluye los botones para realizar las simulaciones como: Modificar, Ping, Iperf, Directorio y Vlan
5	Barra superior Frame. Incluye el icono de la aplicación y el del Frame

Tabla 124. Descripción Interfaz Simulación

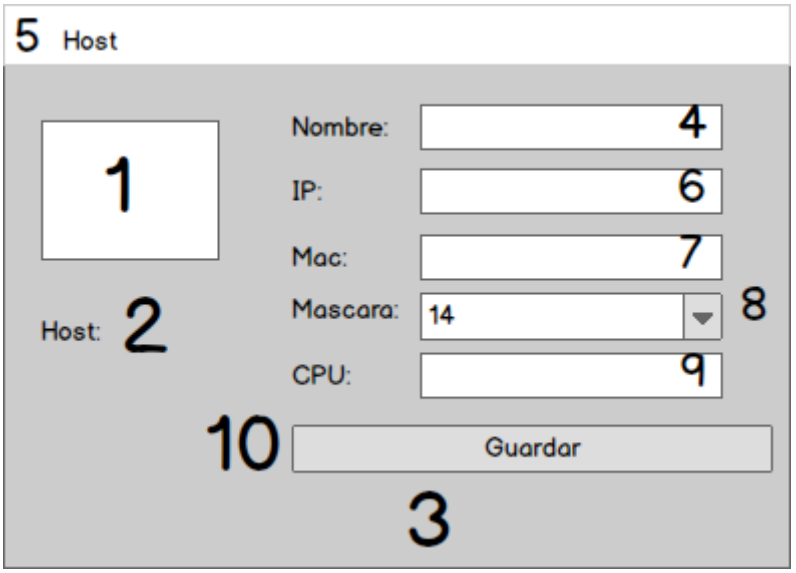


Figura 56. Interfaz Modificas Host

ID	Descripción
1	Imagen: Imagen descriptiva de un host.
2	Texto: Indica el nombre del host que se modifica
3	Frame principal que contiene todos los elementos del host
4	Área de texto en donde se almacena el nombre del host
5	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
6	Área de texto que almacena la ip del host
7	Área de texto que almacena la dirección MAC del host
8	Combo Box que almacena las posibles mascaras del host
9	Área de texto que almacena la cantidad de CPU que puede tener un host
10	Botón que guarda los datos de un host

Tabla 125. Descripción Interfaz Modificar Host.

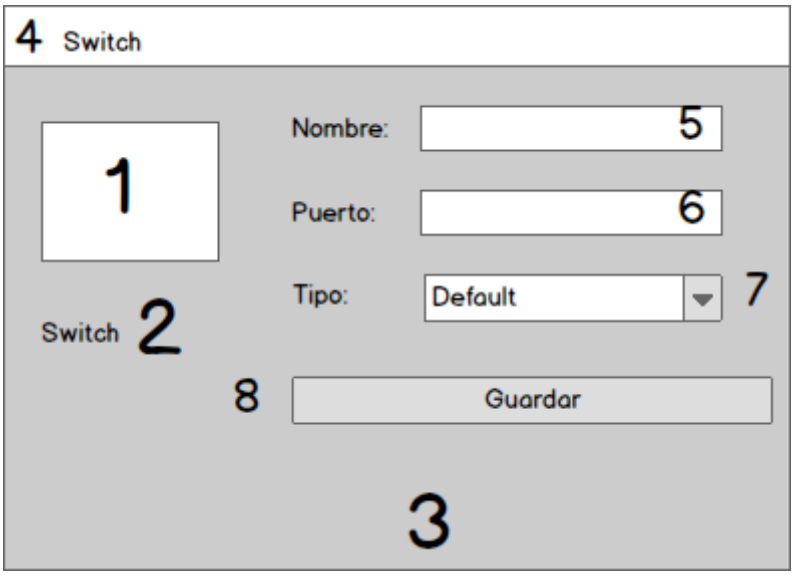


Figura 57. Interfaz Modificar Switch

ID	Descripción
1	Imagen: Imagen descriptiva de un switch.
2	Texto: Indica el nombre del switch que se modifica
3	Frame principal que contiene todos los elementos del switch
4	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
5	Área de texto en donde se almacena el nombre del host
6	Área de texto que almacena el puerto de escucha OpenFlow del switch
7	Combo Box que indica el tipo de switch
8	Botón que guarda los datos de un switch

Tabla 126. Descripción Interfaz Modificar Switch

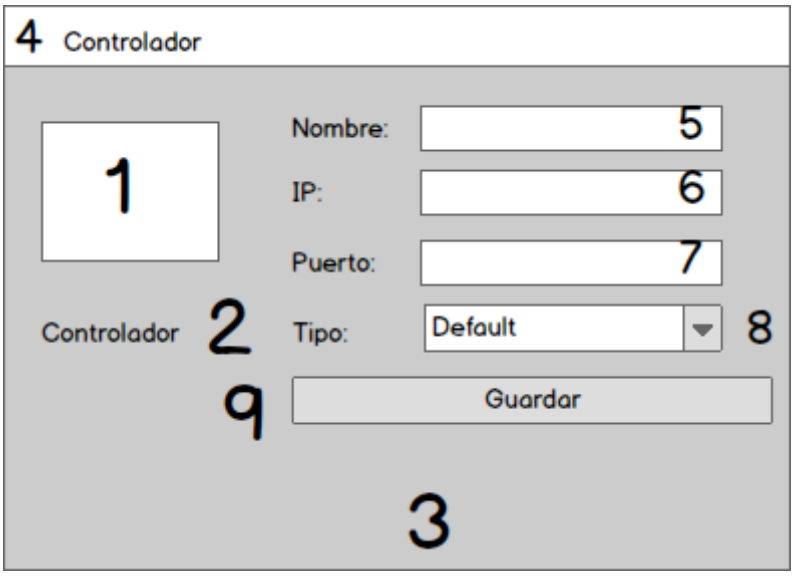


Figura 58. Interfaz Modificar Controlador

ID	Descripción
1	Imagen: Imagen descriptiva de un controlador.
2	Texto: Indica el nombre del switch que se modifica
3	Frame principal que contiene todos los elementos del controlador
4	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
5	Área de texto en donde se almacena el nombre del controlador
6	Área de texto que almacena la ip del controlador
7	Área de texto que almacena el puerto de escucha OpenFlow del controlador
8	Combo Box que almacena el tipo de controlador.
9	Botón que guarda los datos de un switch

Tabla 127. Descripción Interfaz Modificar Controlador

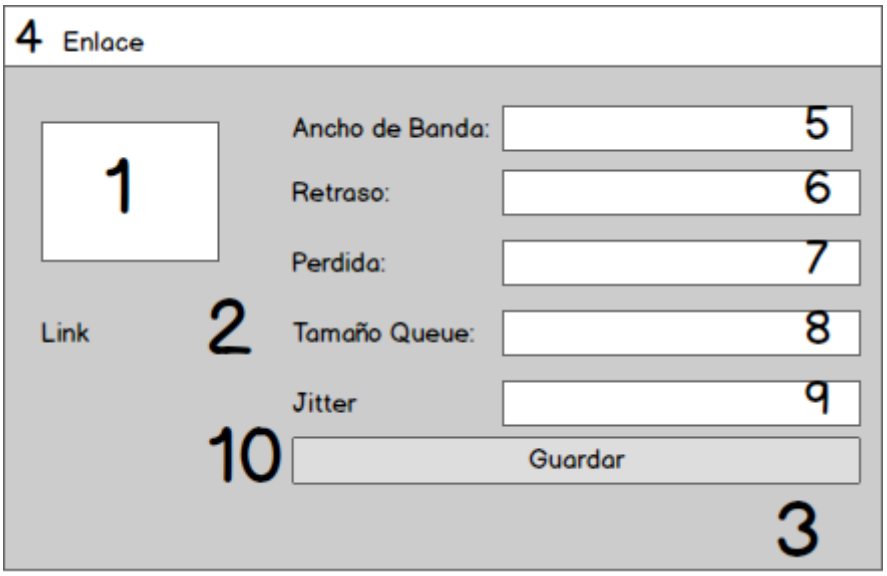


Figura 59. Interfaz Modificar Enlace

ID	Descripción
1	Imagen: Imagen descriptiva de edición.
2	Texto: Indica el nombre del link que se modifica
3	Frame principal que contiene todos los elementos del enlace.
4	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
5	Área de texto en donde se almacena el ancho de banda del enlace
6	Área de texto que almacena el retraso que tiene el enlace
7	Área de texto que almacena el porcentaje de pérdida del enlace
8	Área de texto que indica el tamaño de la fila del enlace
9	Área de texto que almacena la cantidad de Jitter que tendrá un enlace
10	Botón que almacena los datos de un enlace.

Tabla 128. Descripción Interfaz Modificar Enlace

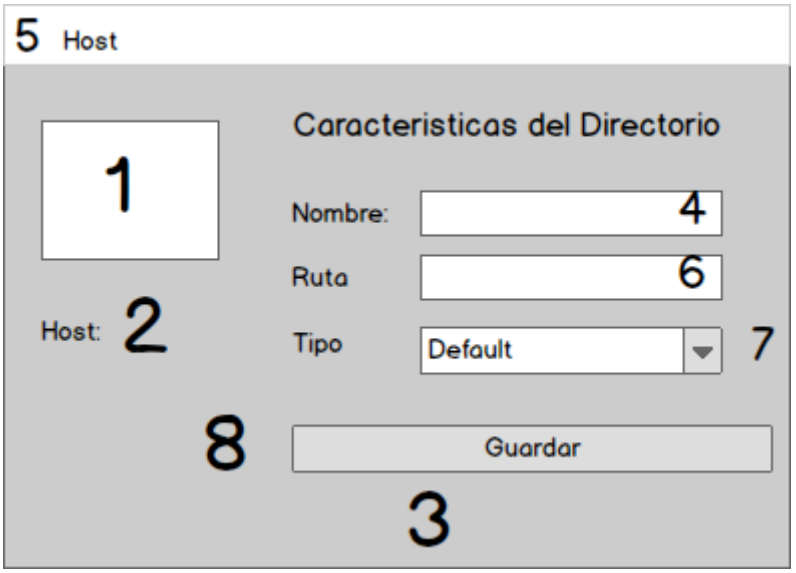


Figura 60. Interfaz Directorio

ID	Descripción
1	Imagen: Imagen descriptiva de edición.
2	Texto: Indica el nombre del host que se modifica
3	Frame principal que contiene todos los elementos de la simulación.
4	Área de texto que almacena el nombre del directorio
5	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
6	Área de texto que almacena la ruta del directorio
7	Combo Box que almacena el tipo de directorio que será.
8	Botón que guarda los cambios de los directorios

Tabla 129.Descripción Interfaz Simular Directorio

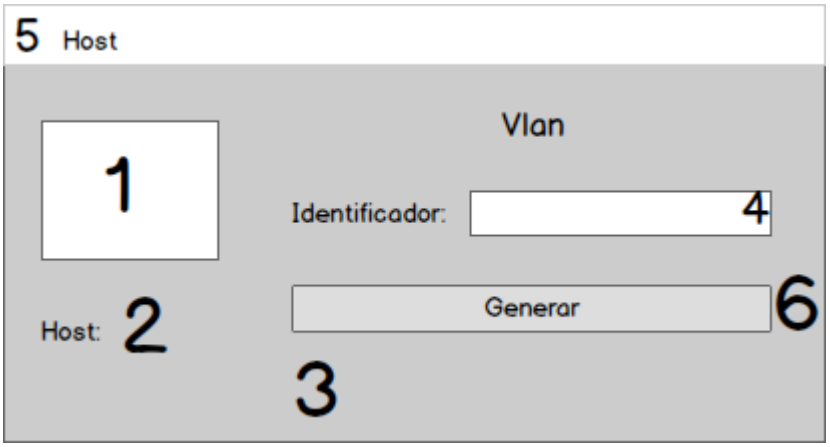


Figura 61. Interfaz Simular Vlan

ID	Descripción
1	Imagen: Imagen descriptiva de edición.
2	Texto: Indica el nombre del host que se modifica
3	Frame principal que contiene todos los elementos de la simulación.
4	Área de texto el identificador de la Vlan
5	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
6	Botón que genera el código de la Vlan en el host designado.

Tabla 130. Descripción Interfaz Simular Vlan

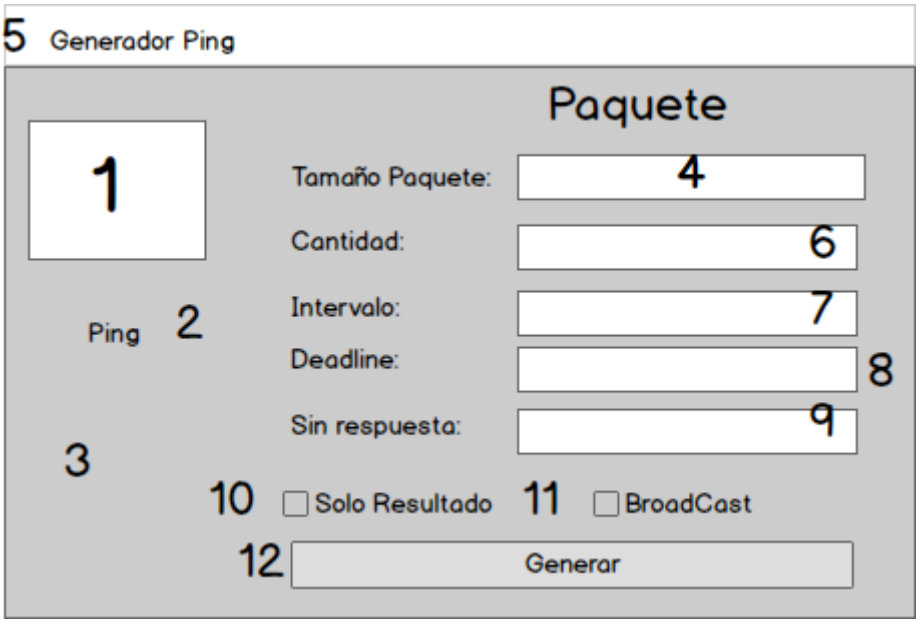


Figura 62. Interfaz Generador Ping

ID	Descripción
1	Imagen: Imagen descriptiva de edición.
2	Texto: Indica el nombre del host que se modifica
3	Frame principal que contiene todos los elementos del host
4	Área de texto en donde se almacena el tamaño del paquete ping
5	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
6	Área de texto que almacena la cantidad de paquetes
7	Área de texto que almacena el intervalo con que se mostraran los resultados del ping
8	Área de texto que almacena el tiempo de vida del comando
9	Área de texto que almacena la cantidad de paquetes que no tendrán respuesta
10	Check Box que almacena si se mostraran o no los resultados intermedios del ping
11	Check Box que almacena si se realiza un ping a la dirección broadcast
12	Botón que genera el código ping entre dos host

Tabla 131. Descripción Interfaz Simular Ping

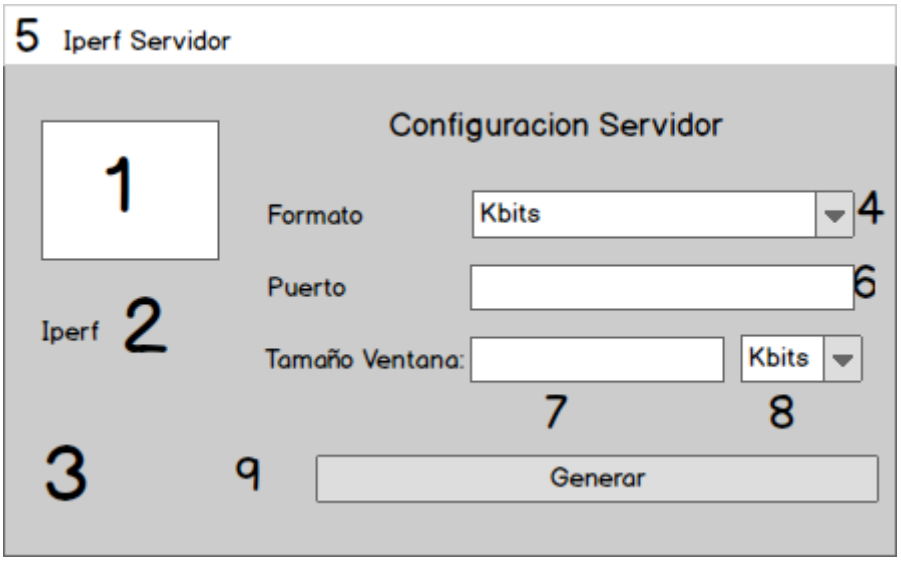


Figura 63. Interfaz Simulación Iperf Servidor

ID	Descripción
1	Imagen: Imagen descriptiva de edición.
2	Texto: Indica el nombre del host que se modifica
3	Frame principal que contiene todos los elementos de configuración del servidor iperf
4	Combo Box en donde se almacena el formato que tendrá el test
5	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
6	Área de texto que almacena el puerto de escucho del comando iperf
7	Área de texto que almacena el tamaño de la ventana del comando iperf
8	Combo Box que almacena el formato que tendrá el tamaño de la ventana
9	Botón que genera el código iperf servidor

Tabla 132. Interfaz Simular Iperf Servidor

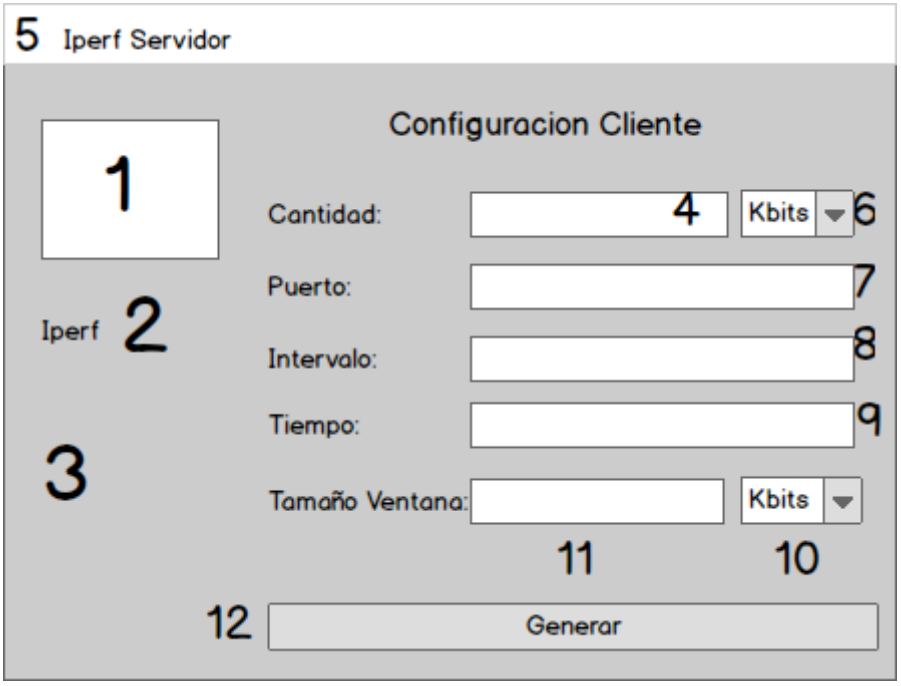


Figura 64. Interfaz Simulación Iperf Cliente

ID	Descripción
1	Imagen: Imagen descriptiva de edición.
2	Texto: Indica el nombre del host que se modifica
3	Frame principal que contiene todos los elementos de configuración del servidor iperf
4	Area de texto que almacena la cantidad de información transferida en el test
5	Barra superior Frame. Incluye el icono de la aplicación y el título del frame.
6	Combo Box que almacena el formato de la cantidad
7	Área de texto que almacena el puerto de escucho del comando Iperf
8	Área de texto que almacena el intervalo en cual se mostraran los resultados
9	Area de texto que almacena el tiempo que durara la prueba
10	Area de texto que almacena el tamaño que tendrá la ventana
11	Combo Box que almacena el formato del tamaño de la ventana
12	Botón que genera el código iperf cliente

Tabla 133. Descripción Interfaz Simulación Iperf Cliente

7.1.2 Diagrama de Menús

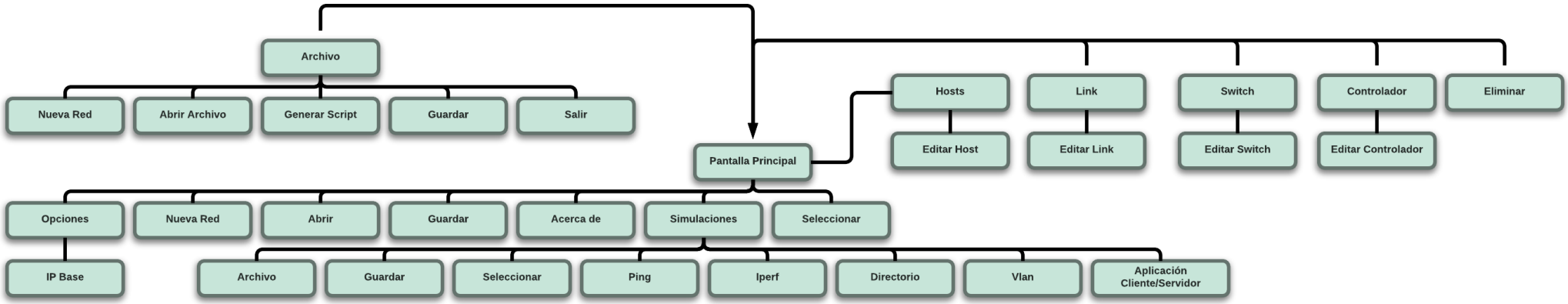


Figura 65. Diagrama de Menús

8 PRUEBAS

Adaptación basada en *IEEE Software Test Documentation* Std 829-1998

8.1 Elementos de prueba

Los módulos del sistema (clases) que serán probados corresponden a todos los campos de entrada y salida que tendra el sistema.

Clase	Descripción
IControlador	Clase utilizada para ingresar campos personalizados de un controlador.
IDirectorio	Clase utilizada para la creación de directorios persistentes o temporales.
IHost	Clase utilizada para ingresar campos personalizados a un host.
IIperf	Clase utilizada para el ingreso de campos de una prueba iperf.
IIperfC	Clase utilizada para el ingreso de campos de una prueba iperf.
ILink	Clase utilizada para ingresar campos personalizados a un link entre dos nodos.
IPing	Clase utilizada para el ingreso de campos de una prueba ping.
ISimul	Clase utilizada para generar simulaciones sobre los nodos.
ISwitch	Clase utilizada para ingresar campos personalizados de un switch.
Generador	Clase utilizada para la generación de código Python compatible con "Mininet".
IVlan	Clase utilizada para el ingreso de Vlans sobre los nodos.

Tabla 134. Módulos a Probar

8.2 Especificación de las pruebas

Características a probar	Nivel de prueba	Objetivo de la Prueba	Enfoque para la definición de casos de prueba	Técnicas para la definición de casos de prueba	Actividades de prueba	Criterios de cumplimiento
Funcionalidad	**Unidad**	**Asegurar el correcto funciona-miento de las entradas y salidas del sistema**	**Caja negra**	**Valores Límites, particiones y valores de entorno**	**Ingresar datos a los diferentes campos, eliminar los datos de los diferentes campos y actualizar**	**Los cambios realizados en los campos se deben ver reflejados en los nodos modificados o por su defecto en las simulaciones realizadas**

8.3 Detalle de las pruebas

8.3.1 <IHost>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de un host.

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	IP	MAC	Mascara	Cpu				
01	Caracteres a nombre del host	Vacío	Vacío	Vacío	Vacío	Vacío	El nombre del host debe contener solo letras y/o números	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo nombre este vacío el software ingresara un nombre por defecto al host.
02	Caracteres a nombre del host	H1	Vacío	Vacío	Vacío	Vacío	El nombre del host debe contener solo letras y/o números	El sistema indica que el ingreso es valido	EXITO	
03	Caracteres a nombre del host	H1!	Vacío	Vacío	Vacío	Vacío	El nombre del host debe contener solo letras y/o números	El sistema no permite tipear caracteres que no sean letras o números	FRACASO	
04	Caracteres a ip del host	Vacío	Vacío	Vacío	Vacío	Vacío	La ip debe ser valida	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo ip este vacío el software ingresara una ip por defecto al host.
05	Caracteres a ip del host	Vacío	192.168.0.1	Vacío	Vacío	Vacío	La ip debe ser valida	El sistema indica que el ingreso es valido	EXITO	
06	Caracteres a ip del host	Vacío	192.168.400.2	Vacío	Vacío	Vacío	La ip debe ser valida	El sistema muestra un mensaje que la ip es invalida	FRACASO	

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	IP	MAC	Mascara	Cpu				
07	Caracteres a MAC del host	Vacío	Vacío	Vacío	Vacío	Vacío	La MAC debe ser valida	El sistema indica que el ingreso es valido	ÉXITO	En caso de que el campo MAC este vacío el software ingresara una MAC por defecto al host.
08	Caracteres a MAC del host	Vacío	Vacío	00:00:00:00:00:01	Vacío	Vacío	La MAC debe ser valida	El sistema indica que el ingreso es valido	ÉXITO	
09	Caracteres a MAC del host	Vacío	Vacío	00:00:333:00:00:0Z	Vacío	Vacío	La MAC debe ser valida	El sistema indica por medio de un mensaje que la MAC es invalida	FRACASO	
10	Caracteres a Mascara del host	Vacío	Vacío	Vacío	Vacío	Vacío	La Máscara debe ser valida	El sistema indica que el ingreso es valido	ÉXITO	En caso de que el campo Máscara este vacío el software ingresara una Máscara por defecto al host.
11	Caracteres a Mascara del host	Vacío	Vacío	Vacío	/13	Vacío	La Máscara debe ser valida	El sistema indica que el ingreso es valido	ÉXITO	
12	Caracteres a Mascara del host	Vacío	Vacío	Vacío	/40	Vacío	La Máscara debe ser valida	El sistema no permite el ingreso de mascara fuera del rango.	FRACASO	
13	Caracteres a cpu del Host	Vacío	Vacío	Vacío	Vacío	Vacío	El número de cpu debe ser valido	El sistema indica que el ingreso es valido	ÉXITO	En caso de que el campo Cpu este vacío el software no asignara recursos personalizados de cpu al host

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	IP	MAC	Mascara	Cpu				
14	Caracteres a cpu del Host	Vacío	Vacío	Vacío	Vacío	1	El número de cpu debe ser valido	El sistema indica que el ingreso es valido	ÉXITO	
15	Caracteres a cpu del Host	Vacío	Vacío	Vacío	Vacío	K	El número de cpu debe ser valido	El sistema no permite el ingreso de letras, solo números son validos	FRACASO	

Tabla 135. Prueba Unidad IHost

8.3.2 <ISwitch>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de un switch.

ID Caso De Prueba	Características a Probar	Datos de Entrada			Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	Puerto	Tipo				
16	Caracteres a nombre del switch	Vacío	Vacío	Vacío	El nombre del switch debe contener solo letras y/o números	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo nombre este vacío el software ingresara un nombre por defecto al switch.
17	Caracteres a nombre del switch	S0	Vacío	Vacío	El nombre del switch debe contener solo letras y/o números	El sistema indica que el ingreso es valido	EXITO	
18	Caracteres a nombre del switch	S!	Vacío	Vacío	El nombre del switch debe contener solo letras y/o números	El sistema no permite tipear caracteres que no sean letras o números	FRACASO	
19	Caracteres a puerto del switch	Vacío	Vacío	Vacío	El puerto del switch debe contener solo letras y/o números	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo puerto este vacío el software ingresara un puerto de comunicación por defecto al switch.

ID Caso De Prueba	Características a Probar	Datos de Entrada			Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	Puerto	Tipo				
20	Caracteres a puerto del switch	Vacío	7000	Vacío	El puerto del switch debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
21	Caracteres a puerto del switch	Vacío	D212	Vacío	El puerto del switch debe contener solo números	El sistema sólo permite tipear números	FRACASO	
22	Caracteres a tipo de switch	Vacío	Vacío	Vacío	El tipo de switch debe pertenecer al rango	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo tipo este por defecto el software ingresara un switch por defecto.
23	Caracteres a tipo de switch	Vacío	Vacío	Vacío	El tipo de switch debe pertenecer al rango	El sistema indica que el ingreso es valido	EXITO	
24	Caracteres a tipo de switch	Vacío	Vacío	Tipo 3	El tipo de switch debe pertenecer al rango	El sistema no permite el ingreso de tipos de switch fuera del rango	FRACASO	

Tabla 136. Prueba Unidad ISwitch

8.3.3 <IControlador>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de un controlador.

ID Caso De Prueba	Características a Probar	Datos de Entrada				Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	IP	Puerto	Tipo				
25	Caracteres a nombre del controlador	Vacío	Vacío	Vacío	Vacío	El nombre del controlador debe contener solo letras y/o números	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo nombre este vacío el software ingresara un nombre por defecto al controlador.
26	Caracteres a nombre del controlador	C1	Vacío	Vacío	Vacío	El nombre del controlador debe contener solo letras y/o números	El sistema indica que el ingreso es valido	EXITO	
27	Caracteres a nombre del controlador	C1!	Vacío	Vacío	Vacío	El nombre del controlador debe contener solo letras y/o números	El sistema no permite tipear caracteres que no sean letras o números	FRACASO	
28	Caracteres a ip del controlador	Vacío	Vacío	Vacío	Vacío	La ip debe ser valida	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo ip este vacío el software ingresara una ip por defecto al controlador.
29	Caracteres a ip del controlador	Vacío	192.168.0.5	Vacío	Vacío	La ip debe ser valida	El sistema indica que el ingreso es valido	EXITO	
30	Caracteres a ip del controlador	Vacío	192.400.3.1	Vacío	Vacío	La ip debe ser valida	El sistema muestra un mensaje que la ip es invalida	FRACASO	
31	Caracteres a puerto del controlador	Vacío	Vacío	Vacío	Vacío	El puerto del controlador debe contener solo números	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo puerto este vacío el software ingresara puerto por defecto al controlador.

ID Caso De Prueba	Características a Probar	Datos de Entrada				Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	IP	Puerto	Tipo				
32	Caracteres a puerto del switch	Vacío	Vacío	7001	Vacío	El puerto del controlador debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
33	Caracteres a puerto del switch	Vacío	Vacío	D7000	Vacío	El puerto del controlador debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	
34	Caracteres a tipo de controlador	Vacío	Vacío	Vacío	Vacío	El tipo de controlador debe pertenecer al rango	El sistema indica que el ingreso es valido	EXITO	En caso de que el campo tipo este vacío el software ingresara un controlador por defecto.
35	Caracteres a tipo de controlador	Vacío	Vacío	Vacío	OVSController	El tipo de controlador debe pertenecer al rango	El sistema indica que el ingreso es valido	EXITO	
36	Caracteres a tipo de controlador	Vacío	Vacío	Vacío	Tipo 5	El tipo de controlador debe pertenecer al rango	El sistema no permite el ingreso de tipos de controladores fuera del rango	FRACASO	

Tabla 137. Prueba Unidad IControlador

8.3.4 <ILink>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de un link entre dos nodos.

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Ancho de Banda	Retraso	Perdida	Tamaño Queue	Jitter				
37	Caracteres a ancho de banda del link	Vacío	Vacío	Vacío	Vacío	Vacío	El ancho de banda del link debe contener solo números	El sistema indica que el ingreso es valido	EXITO	El sistema ingresa un diccionario vacío en el caso de que alguno de los campos de link esté vacío.

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Ancho de Banda	Retraso	Perdida	Tamaño Queue	Jitter				
38	Caracteres a ancho de banda del link	10000	Vacío	Vacío	Vacío	Vacío	El ancho de banda del link debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
39	Caracteres a ancho de banda del link	1000D	Vacío	Vacío	Vacío	Vacío	El ancho de banda del link debe contener solo números	El sistema sólo permite tipear números	FRACASO	
40	Caracteres a retraso del link	Vacío	Vacío	Vacío	Vacío	Vacío	El retraso del link debe contener solo números	El sistema indica que el ingreso es valido	EXITO	El sistema ingresa un diccionario vacío en el caso de que alguno de los campos de link esté vacío.
41	Caracteres a retraso del link	Vacío	1000	Vacío	Vacío	Vacío	El retraso del link debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
42	Caracteres a retraso del link	Vacío	1000f	Vacío	Vacío	Vacío	El retraso del link debe contener solo números	El sistema sólo permite tipear números	FRACASO	
43	Caracteres a pérdida del link	Vacío	Vacío	Vacío	Vacío	Vacío	La pérdida del link debe contener solo números	El sistema indica que el ingreso es valido	EXITO	El sistema ingresa un diccionario vacío en el caso de que alguno de los campos de link esté vacío.
44	Caracteres a pérdida del link	Vacío	Vacío	30	Vacío	Vacío	La pérdida del link debe contener solo números	El sistema indica que el ingreso es valido	EXITO	

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Ancho de Banda	Retraso	Perdida	Tamaño Queue	Jitter				
45	Caracteres a pérdida del link	Vacío	Vacío	40D	Vacío	Vacío	La pérdida del link debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	
46	Caracteres a tamaño queue del link	Vacío	Vacío	Vacío	Vacío	Vacío	El tamaño debe contener solo números	El sistema indica que el ingreso es valido	EXITO	El sistema ingresa un diccionario vacío en el caso de que alguno de los campos de link esté vacío.
47	Caracteres a tamaño queue del link	Vacío	Vacío	Vacío	300	Vacío	El tamaño debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
48	Caracteres a tamaño queue del link	Vacío	Vacío	Vacío	300D	Vacío	El tamaño debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	
49	Caracteres a jitter del link	Vacío	Vacío	Vacío	Vacío	Vacío	El jitter debe contener solo números	El sistema indica que el ingreso es valido	EXITO	El sistema ingresa un diccionario vacío en el caso de que alguno de los campos de link esté vacío.
50	Caracteres a jitter del link	Vacío	Vacío	Vacío	Vacío	4	El jitter debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
51	Caracteres a jitter del link	Vacío	Vacío	Vacío	Vacío	4D	El jitter debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	

Tabla 138. Prueba Unidad ILink

8.3.5 <IPing>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de un ping entre dos nodos.

ID Caso De Prueba	Características a Probar	Datos de Entrada							Salida Esperada	Salida Obtenida	Éxito/ Fracaso
		Tamaño	Cantidad	Intervalo	Deadline	Sin repuesta	Resultado	BroadCast			
52	Caracteres a tamaño	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	El tamaño debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
53	Caracteres a tamaño	300	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	El tamaño debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
54	Caracteres a tamaño	300D	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	El tamaño debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO
55	Caracteres a cantidad	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	La cantidad debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
56	Caracteres a cantidad	Vacío	4	Vacío	Vacío	Vacío	Vacío	Vacío	La cantidad debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
57	Caracteres a cantidad	Vacío	4D	Vacío	Vacío	Vacío	Vacío	Vacío	La cantidad debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO

ID Caso De Prueba	Características a Probar	Datos de Entrada							Salida Esperada	Salida Obtenida	Éxito/ Fracaso
		Tamaño	Cantidad	Intervalo	Deadline	Sin repuesta	Resultado	BroadCast			
58	Caracteres a intervalo	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	El intervalo debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
59	Caracteres a intervalo	Vacío	Vacío	2	Vacío	Vacío	Vacío	Vacío	El intervalo debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
60	Caracteres a intervalo	Vacío	Vacío	2D	Vacío	Vacío	Vacío	Vacío	El intervalo debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO
61	Caracteres a deadline	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	El deadline debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
62	Caracteres a deadline	Vacío	Vacío	Vacío	3	Vacío	Vacío	Vacío	El deadline debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
63	Caracteres a deadline	Vacío	Vacío	Vacío	3D	Vacío	Vacío	Vacío	El deadline debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO
64	Caracteres a sin respuesta	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	El parámetro sin respuesta debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO

ID Caso De Prueba	Características a Probar	Datos de Entrada							Salida Esperada	Salida Obtenida	Éxito/ Fracaso
		Tamaño	Cantidad	Intervalo	Deadline	Sin repuesta	Resultado	BroadCast			
65	Caracteres a sin respuesta	Vacío	Vacío	Vacío	Vacío	2	Vacío	Vacío	El parámetro sin respuesta debe contener solo números	El sistema indica que el ingreso es valido	ÉXITO
66	Caracteres a sin respuesta	Vacío	Vacío	Vacío	Vacío	2D	Vacío	Vacío	El parámetro sin respuesta debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO
67	Caracteres a resultado	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	El parámetro resultado debe ser booleano	El sistema indica que el ingreso es valido	ÉXITO
68	Caracteres a resultado	Vacío	Vacío	Vacío	Vacío	Vacío	TRUE	Vacío	El parámetro resultado debe ser booleano	El sistema indica que el ingreso es valido	ÉXITO
69	Caracteres a resultado	Vacío	Vacío	Vacío	Vacío	Vacío	1D	Vacío	El parámetro resultado debe ser booleano	El sistema no permite ingresos que no sean booleanos	FRACASO
70	Caracteres a broadcast	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	El parámetro broadcast debe ser booleano	El sistema indica que el ingreso es valido	ÉXITO
71	Caracteres a broadcast	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	TRUE	El parámetro broadcast debe ser booleano	El sistema indica que el ingreso es valido	ÉXITO
72	Caracteres a broadcast	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	1D	El parámetro broadcast debe ser booleano	El sistema no permite ingresos que no sean booleanos	FRACASO

Tabla 139. Prueba Unidad IPing

8.3.6 <IIperfC>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de una simulación Iperf cliente en un nodo.

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Cantidad	Puerto	Intervalo	Tiempo	Tamaño Ventana				
73	Caracteres a cantidad	Vacío	Vacío	Vacío	Vacío	Vacío	La cantidad debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
74	Caracteres a cantidad	300	Vacío	Vacío	Vacío	Vacío	La cantidad debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
75	Caracteres a cantidad	300D	Vacío	Vacío	Vacío	Vacío	La cantidad debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	
76	Caracteres a puerto	Vacío	Vacío	Vacío	Vacío	Vacío	El puerto debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
77	Caracteres a puerto	Vacío	7500	Vacío	Vacío	Vacío	El puerto debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
78	Caracteres a puerto	Vacío	7500D	Vacío	Vacío	Vacío	El puerto debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	
79	Caracteres a intervalo	Vacío	Vacío	Vacío	Vacío	Vacío	El intervalo debe contener solo números	El sistema indica que el ingreso es valido	EXITO	

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Cantidad	Puerto	Intervalo	Tiempo	Tamaño Ventana				
80	Caracteres a intervalo	Vacío	Vacío	3	Vacío	Vacío	El intervalo debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
81	Caracteres a intervalo	Vacío	Vacío	3D	Vacío	Vacío	El intervalo debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	
82	Caracteres a tiempo	Vacío	Vacío	Vacío	Vacío	Vacío	El parámetro tiempo debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
83	Caracteres a tiempo	Vacío	Vacío	Vacío	4000	Vacío	El parámetro tiempo debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
84	Caracteres a tiempo	Vacío	Vacío	Vacío	4000D	Vacío	El parámetro tiempo debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	
85	Caracteres a tamaño ventana	Vacío	Vacío	Vacío	Vacío	Vacío	El parámetro tamaño ventana debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
86	Caracteres a tamaño ventana	Vacío	Vacío	Vacío	Vacío	1000	El parámetro tamaño ventana debe contener solo números	El sistema indica que el ingreso es valido	EXITO	

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Cantidad	Puerto	Intervalo	Tiempo	Tamaño Ventana				
87	Caracteres a tamaño ventana	Vacío	Vacío	Vacío	Vacío	1000D	El parámetro tamaño ventana debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	

Tabla 140. Prueba Unidad IIperfC

8.3.7 <IIperf>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de una simulación Iperf servidor en un nodo.

ID Caso De Prueba	Características a Probar	Datos de Entrada		Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Puerto	Tamaño Ventana				
88	Caracteres a puerto	Vacío	Vacío	El puerto debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
89	Caracteres a puerto	7500	Vacío	El puerto debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
90	Caracteres a puerto	7500D	Vacío	El puerto debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	
91	Caracteres a tamaño ventana	Vacío	Vacío	El tamaño ventana debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
92	Caracteres a tamaño ventana	Vacío	1000	El tamaño ventana debe contener solo números	El sistema indica que el ingreso es valido	EXITO	

ID Caso De Prueba	Características a Probar	Datos de Entrada		Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Puerto	Tamaño Ventana				
93	Caracteres a tamaño ventana	Vacío	1000D	El tamaño ventana debe contener solo números	El sistema no permite tipear caracteres que no sean números	FRACASO	

Tabla 141. Prueba Unidad IIperf

8.3.8 <IDirectorio>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de una simulación tipo directorio dentro un nodo.

ID Caso De Prueba	Características a Probar	Datos de Entrada		Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	Ruta				
94	Caracteres a nombre	Vacío	Vacío	El nombre debe contener solo letras y/o números	El sistema indica por medio de un mensaje que el nombre no puede estar vacío	FRACASO	
95	Caracteres a nombre	Directorio	Vacío	El nombre debe contener solo letras y/o números	El sistema indica que el ingreso es valido	EXITO	
96	Caracteres a nombre	Directorio!	Vacío	El nombre debe contener solo letras y/o números	El sistema no permite tipear caracteres que no sean letras o números	FRACASO	
97	Caracteres a ruta	Vacío	Vacío	El nombre debe contener solo letras, números o /.	El sistema indica por medio de un mensaje que la ruta no puede estar vacía	FRACASO	

ID Caso De Prueba	Características a Probar	Datos de Entrada		Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre	Ruta				
98	Caracteres a ruta	Vacío	Home /carpeta	El nombre debe contener solo letras, números o /.	El sistema indica que el ingreso es valido	EXITO	
99	Caracteres a ruta	Vacío	Home /carpeta#	El nombre debe contener solo letras, números o /.	El sistema no permite tipear caracteres que no sean letras , números o /	FRACASO	

Tabla 142. Prueba Unidad IDirectorio

8.3.9 <IVlan>

En este ítem se especifican:

- Configuración estándar tanto para hardware como para software y SO.
- Pre condiciones de las pruebas: El usuario debe haber abierto la ventana del frame de una simulación tipo vlan dentro un nodo.

ID Caso De Prueba	Características a Probar	Datos de Entrada	Salida Esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Identificador				
100	Caracteres a identificador	Vacío	Identificador debe contener solo números	El sistema indica por medio de un mensaje que el identificador no puede estar vacío	FRACASO	
101	Caracteres a identificador	10	Identificador debe contener solo números	El sistema indica que el ingreso es valido	EXITO	
102	Caracteres a identificador	10D	Identificador debe contener solo números	El sistema indica por medio de un mensaje que el identificador solo puede contener números	FRACASO	

Tabla 143. Prueba Unidad IVlan

8.4 Conclusiones de Prueba

Se concluye en las pruebas realizadas que se debe tener prolijidad en el ingreso, y verificación de datos esto es debido a que hay campos sensibles (ip,mac,nombre de nodos) que pueden afectar de forma rotunda a la generacion de scripts, es por esto que la validacion fue un punto de inflexion rotundo en las pruebas realizadas.

9 CASOS DE PRUEBA

El siguiente item muestra 3 casos de pruebas que dan conformidad a los objetivos del proyecto de titulo

9.1 Generación de una topología

Paso 1: Se debe abrir el software MininetGUI y abrir una nueva red.

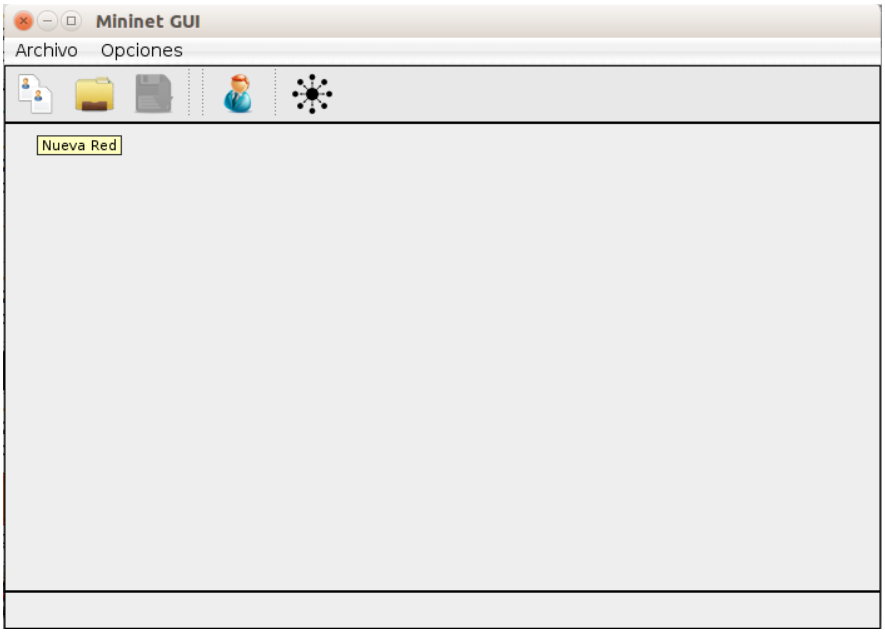


Figura 66. GUI - Nueva Red

Paso 2: Se abrirá un panel de trabajo en el que es posible agregar hosts, switches, controladores y enlaces a través de la barra de herramientas ubicada en el sector lateral derecho de la ventana.

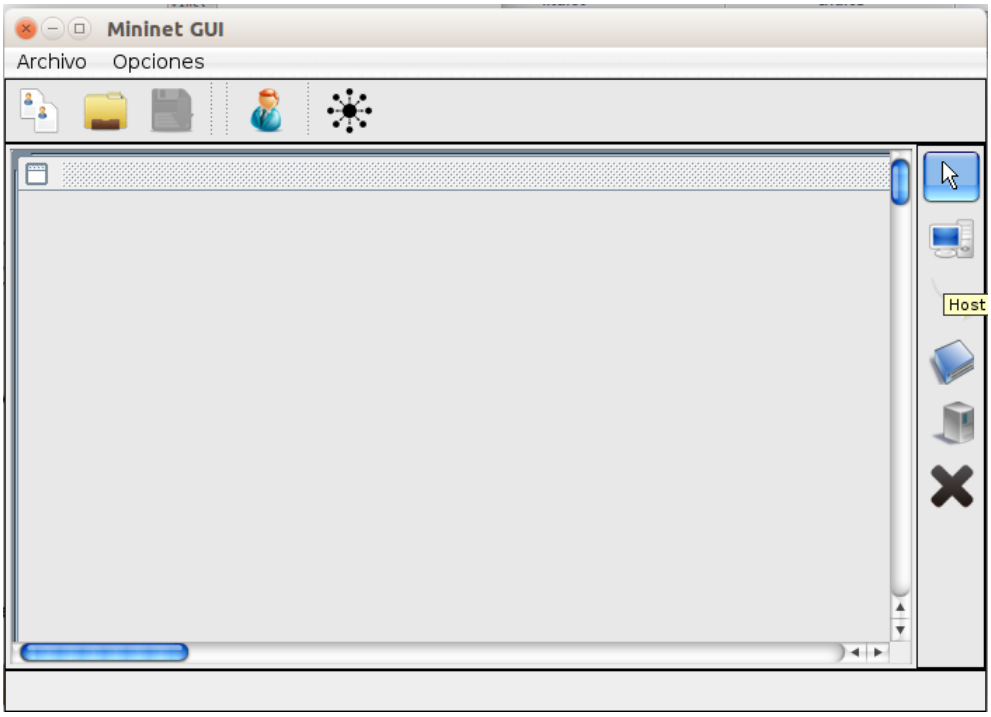


Figura 67. GUI - Panel de Trabajo

Paso 3: Una vez ubicado en la barra de herramientas se procederá a crear una topología con un controlador conectado a un switch y este switch conectado a 4 hosts.

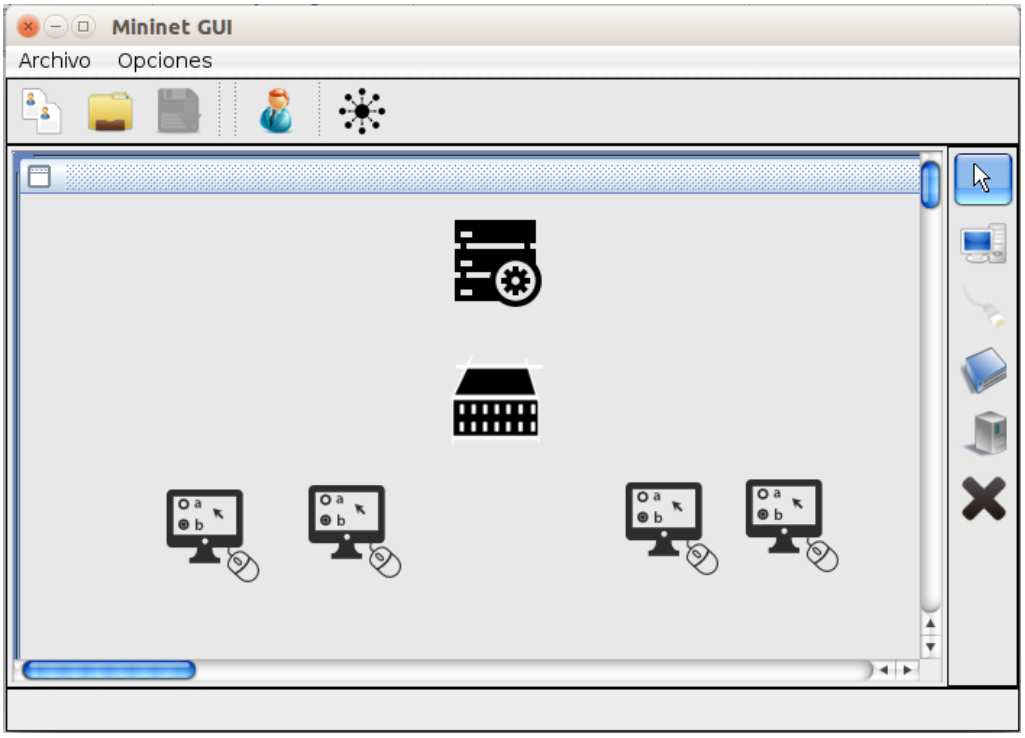


Figura 68. GUI - Topología I

Observación 1: La creación de un enlace se realiza mediante la selección de la herramienta Link, seleccionando los nodos correspondientes.

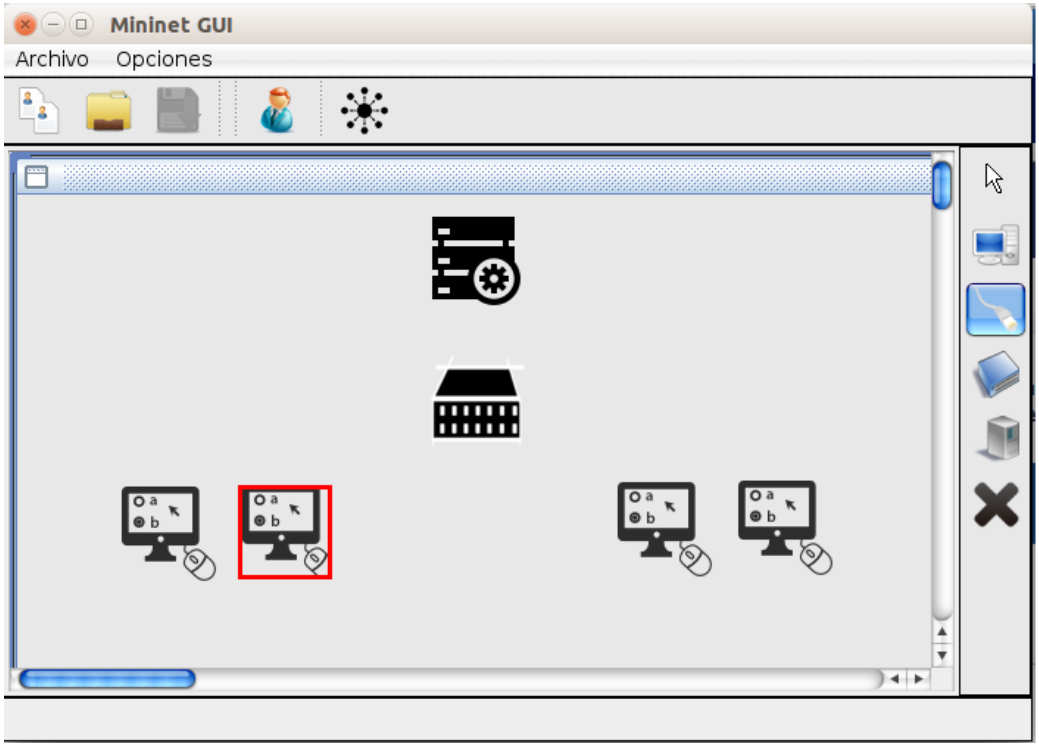


Figura 69. Creación de Enlace I

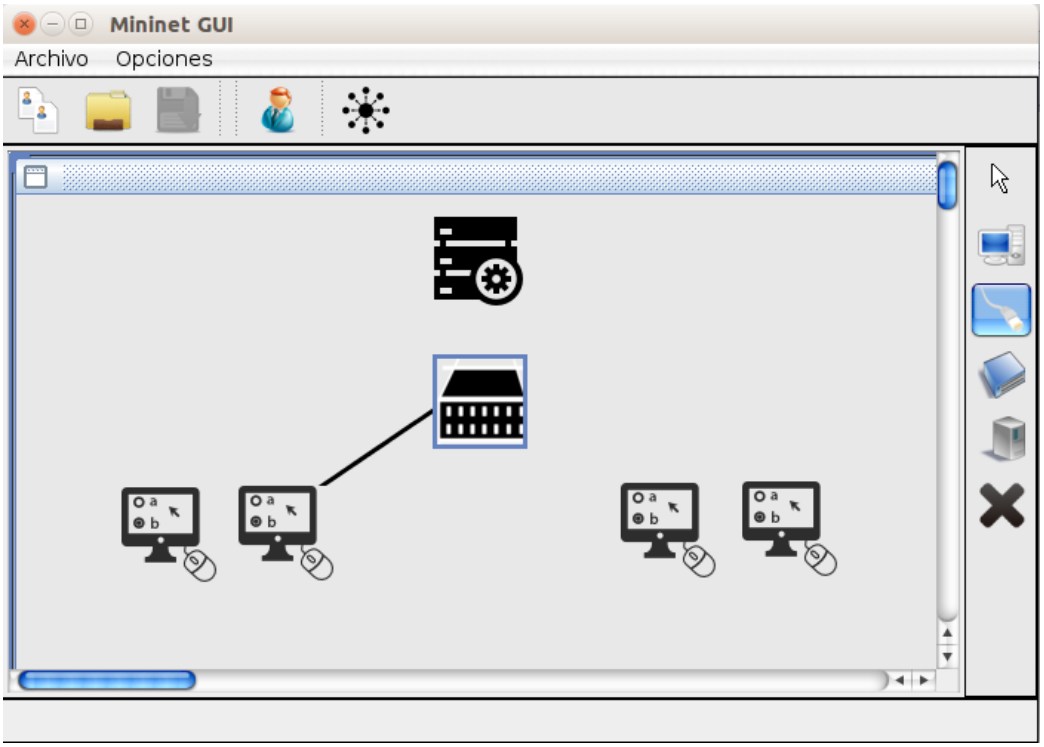


Figura 70. Creación de Enlace II

Paso 4: Una vez creados todos los enlaces es posible editar los parámetros de cada uno de los nodos (OPCIONAL) haciendo click derecho sobre ellos.

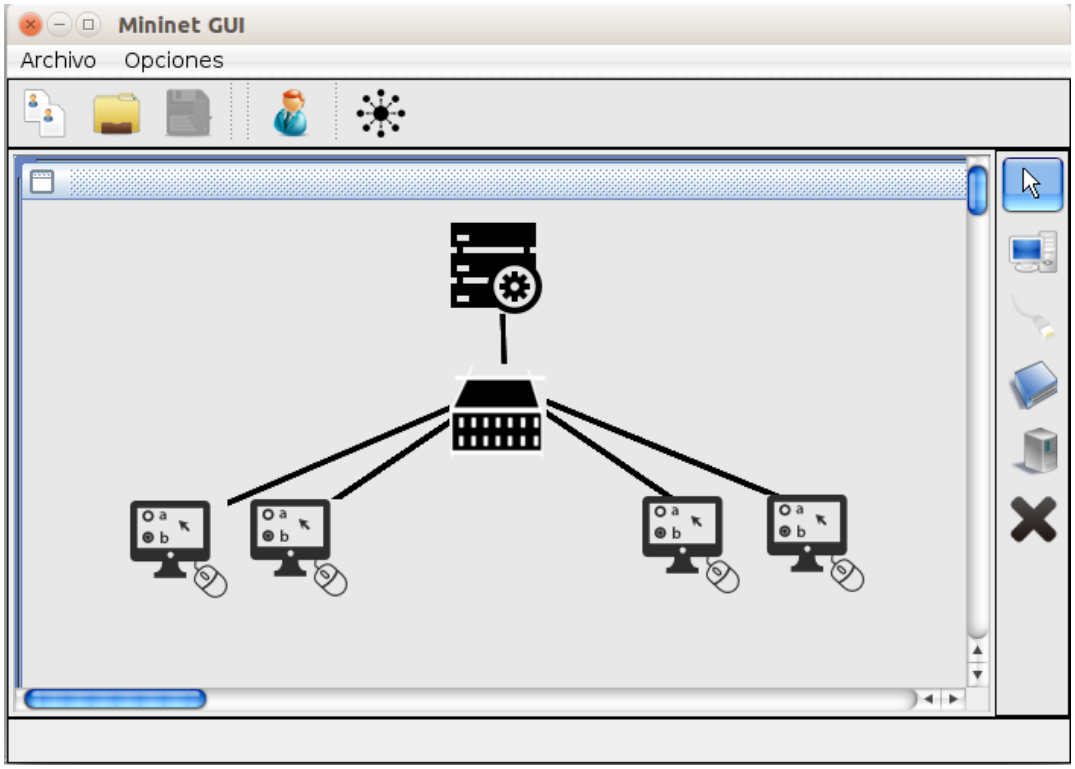


Figura 71. Topología de Red

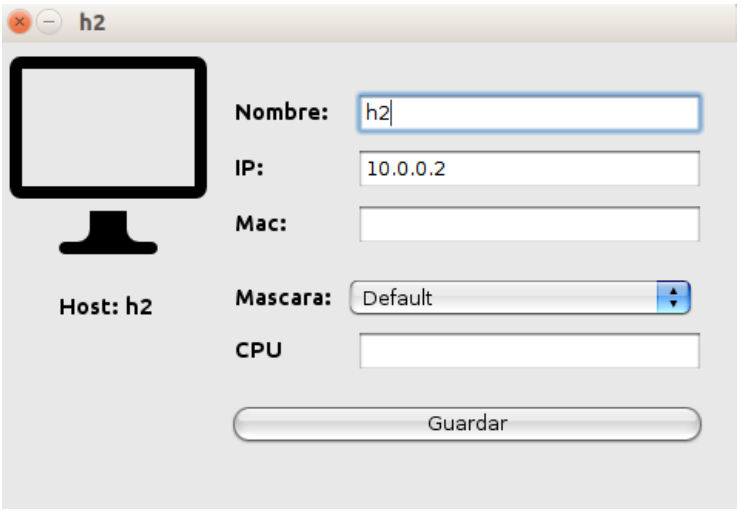


Figura 72. Edición de Host

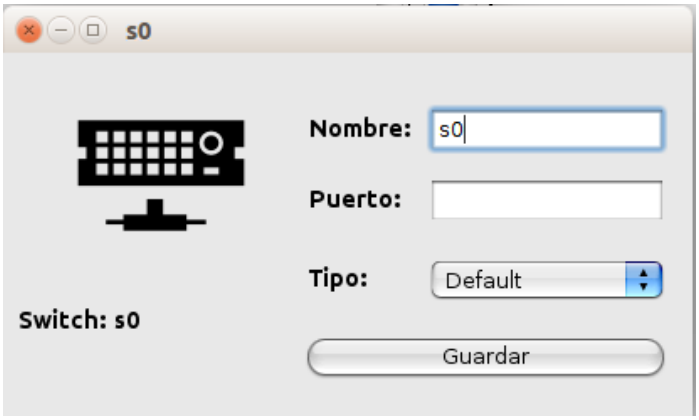


Figura 73. Edición Switch

Figura 74. Edición Controlador

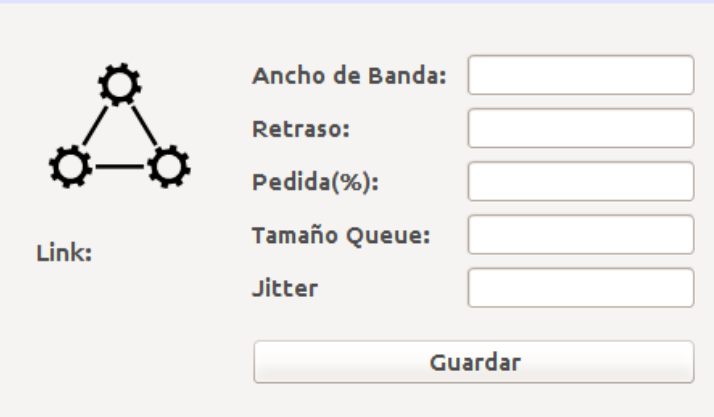


Figura 75. Edición Link

Paso 5: Una vez creada la topología se puede generar el código "Mininet" en la barra de herramientas superior, se debe dirigir a la sección "Archivo" y posteriormente "Generar Script" y guardarlo con el nombre que se desee.

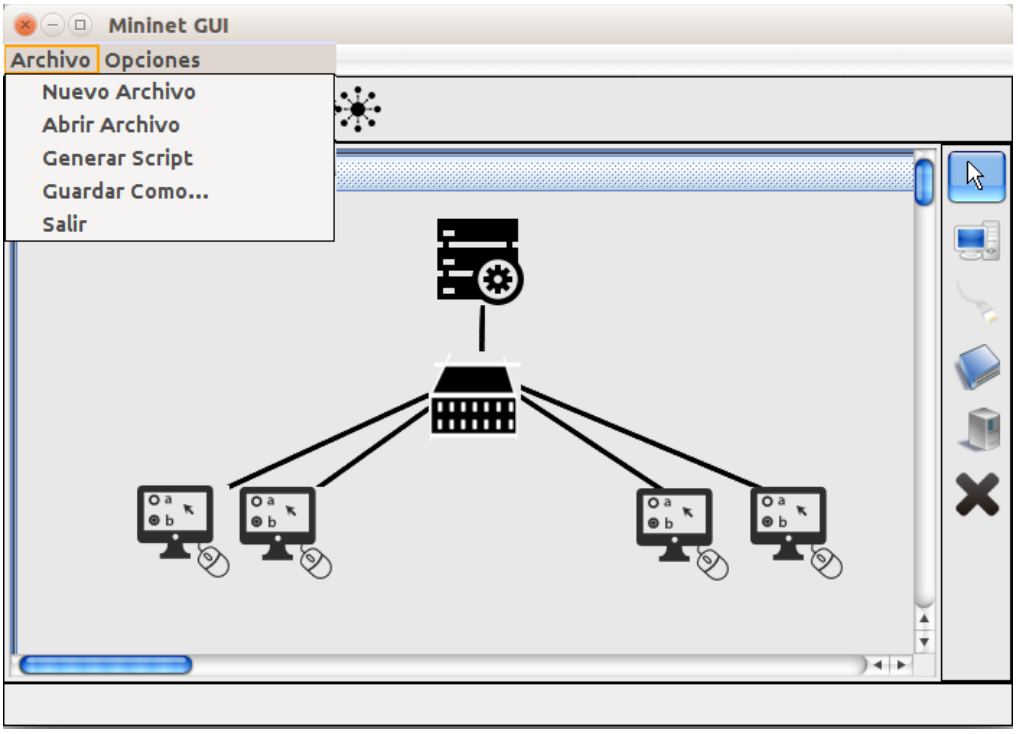


Figura 76. Generación de Topología

Paso 6: Este script se guardara en la carpeta de ejecución del proyecto.

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call



def myNetwork():

    net = Mininet(topo=None,build=False,ipBase='10.0.0.0/8')
    info( '-- Creando Controlador\n' )
    c0
=net.addController(name='c0',controller=Controller,protocol='tcp',port=
6633)

    info( '-- Creando Swtichs\n' )
    s0 =net.addSwitch('s0',cls=OVSKernelSwitch)

    info( '-- Creando Hosts\n' )
    h1 =net.addHost('h1', cls=Host, ip='10.0.0.1', defaulRoute=None)
    h2 =net.addHost('h2', cls=Host, ip='10.0.0.2', defaulRoute=None)
    h3 =net.addHost('h3', cls=Host, ip='10.0.0.3', defaulRoute=None)
    h4 =net.addHost('h4', cls=Host, ip='10.0.0.4', defaulRoute=None)

    info( '-- Creando Enlaces\n' )
    net.addLink(h2, s0)
    net.addLink(h1, s0)
    net.addLink(h3, s0)
    net.addLink(h4, s0)

    info( '-- Iniciando Red\n' )
    net.build()
    info( '-- Iniciando Controlador\n' )
    for controller in net.controllers:
        controller.start()

    info( '-- Iniciando Switchs\n' )
    net.get('s0').start([c0])
    info( '-- Iniciando Consola de Comandos\n')
    CLI(net)
```

```
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

Paso 7: Para ejecutar el código recién generado se debe ingresar a una consola en Linux y ejecutar el código recién generado con el comando Python y el nombre del archivo como lo muestra la figura 75.

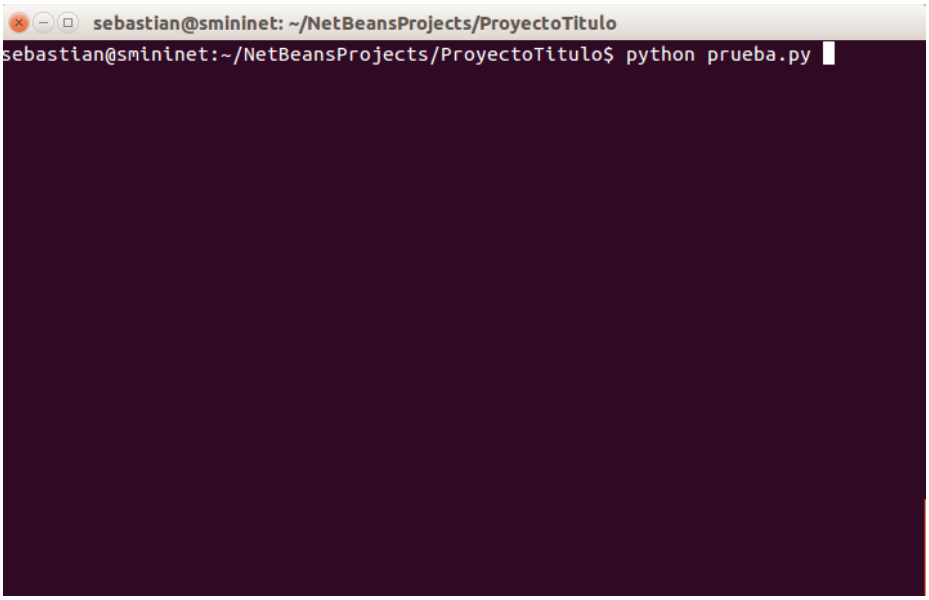


Figura 77. Ejecución de Script Generado

9.2 Generación de simulaciones

Paso 1: Una vez creada la simulación se procede abrir el panel de simulaciones.

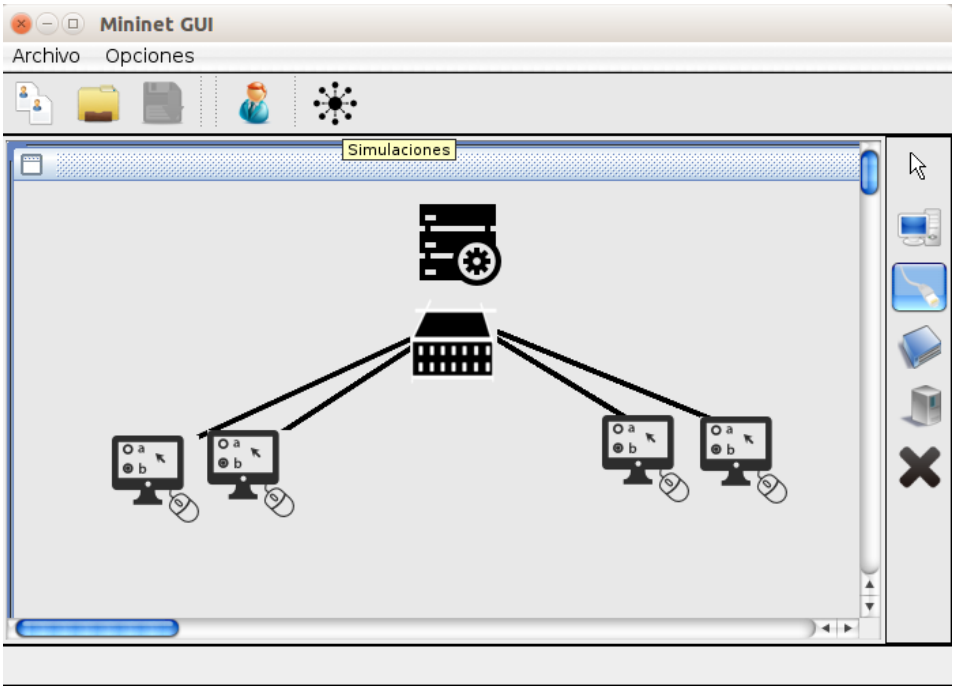


Figura 78. Panel de Simulaciones I

Paso 2: La barra lateral ofrece 5 tipos de simulaciones, entre ellas: prueba ping, prueba iperf, prueba directorio, prueba vlan. Entre ellas abarcaremos 2, ping y vlans.

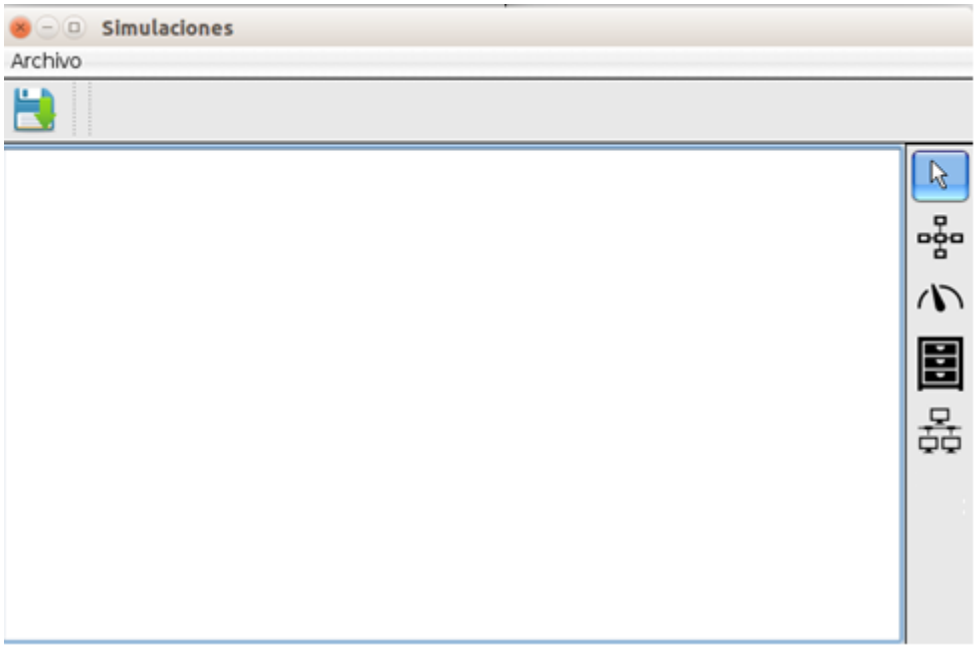


Figura 79. Panel de Simulaciones II

Paso 3: Para realizar una prueba ping basta con seleccionar la herramienta ping y seleccionar los host del panel de la red, elegir los parámetros a elección y generar el código de la simulación.

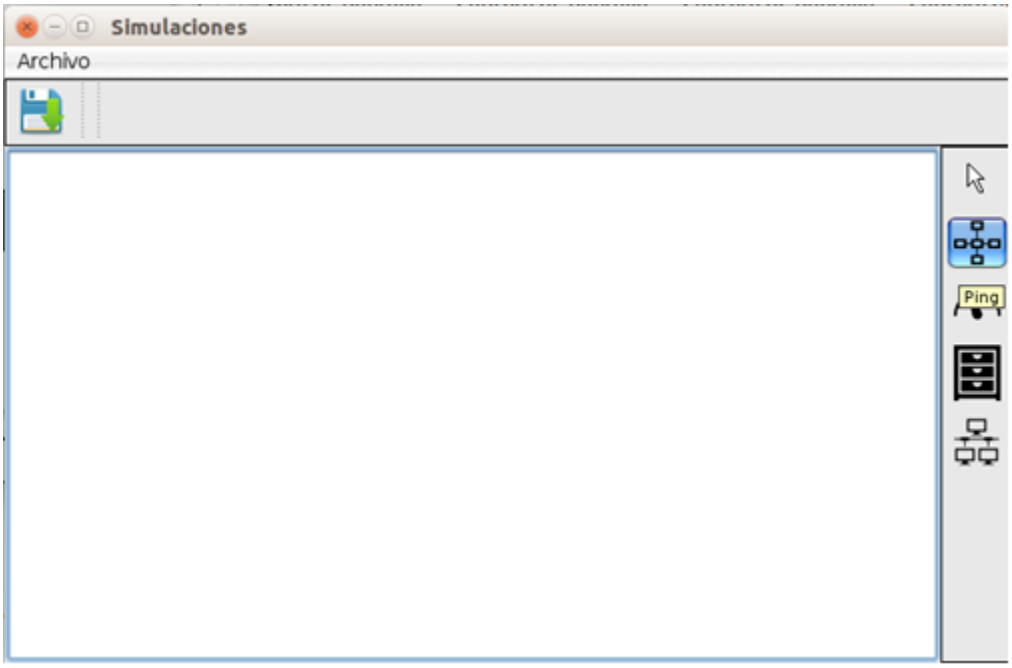


Figura 80. Prueba Ping I

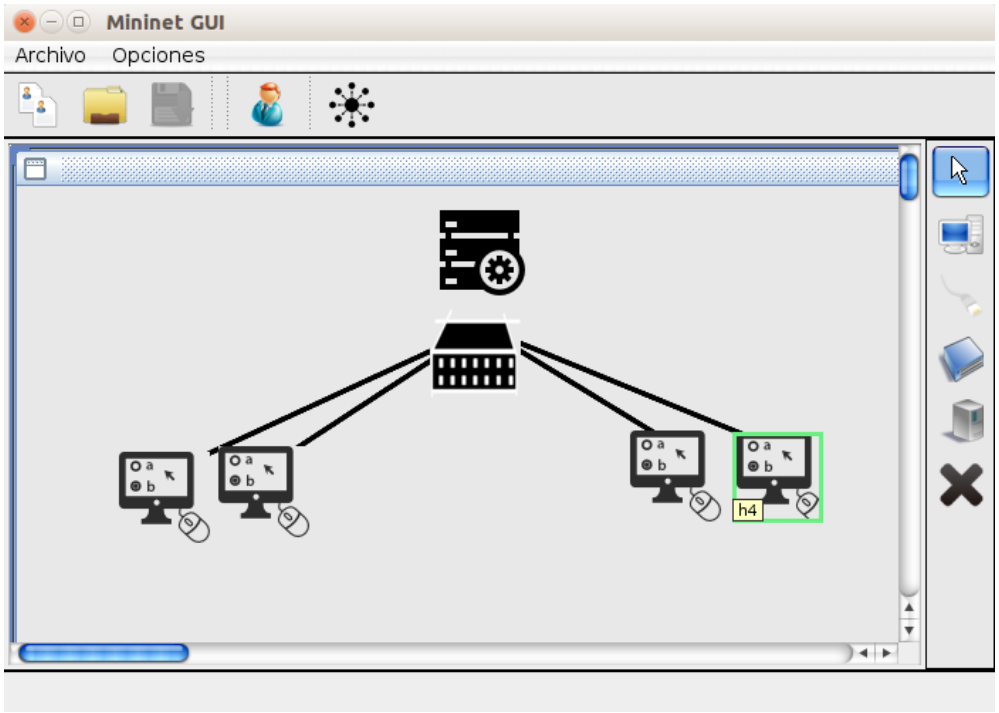


Figura 81. Prueba Ping II

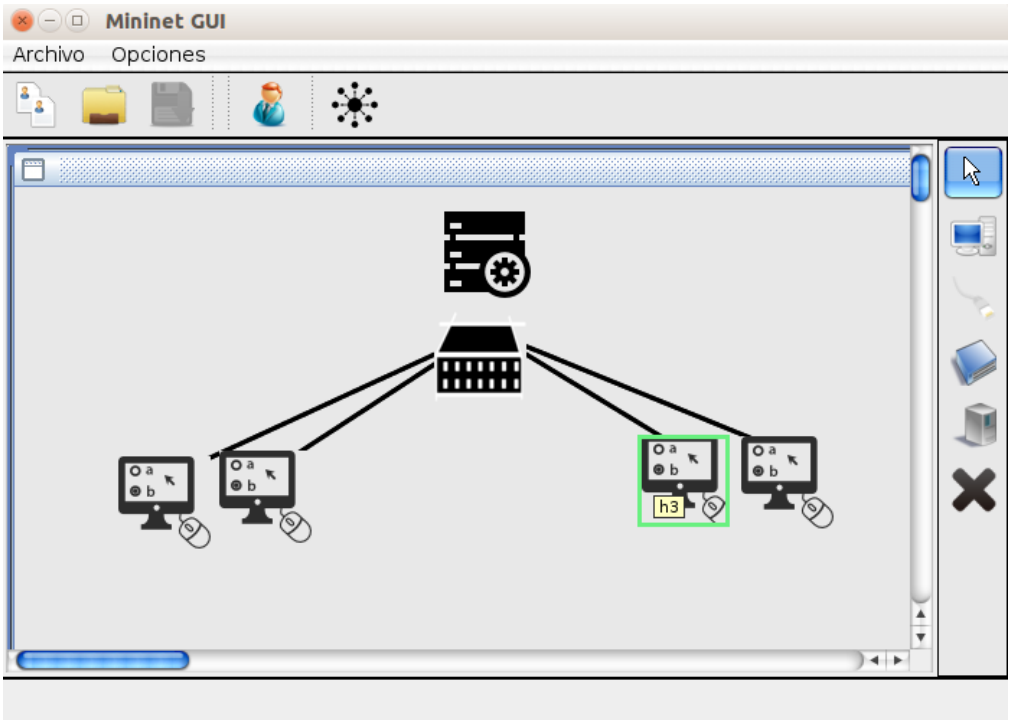


Figura 82. Prueba Ping III

Figura 83. Prueba Ping IV

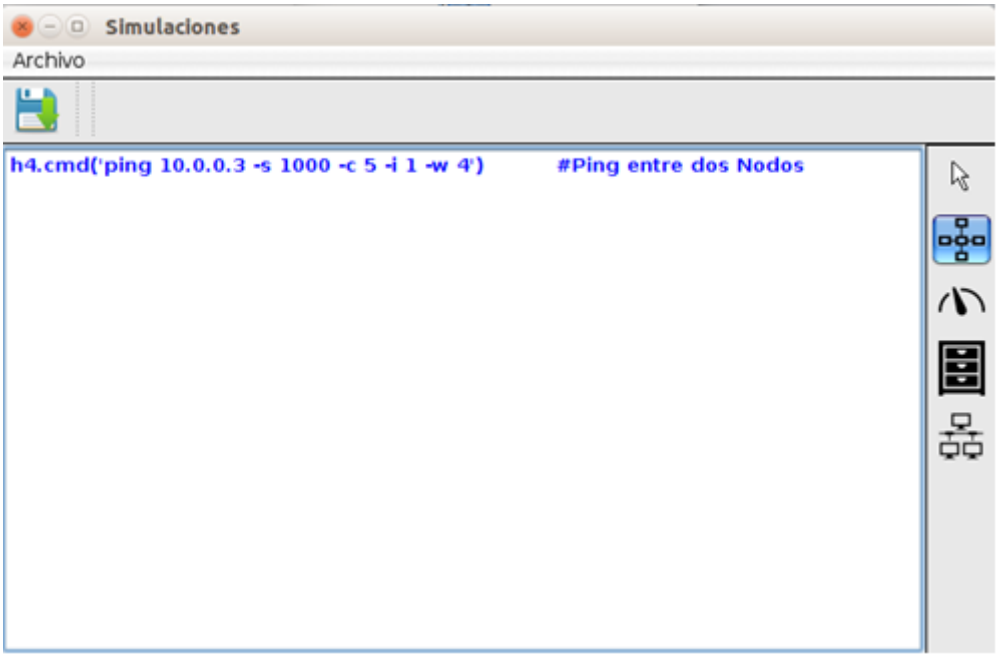


Figura 84. Prueba Ping V

Paso 4: Para el caso de la creación de vlan solo basta elegir la herramienta Vlan , elegir el host en el panel de red y agregar el identificador de la Vlan.

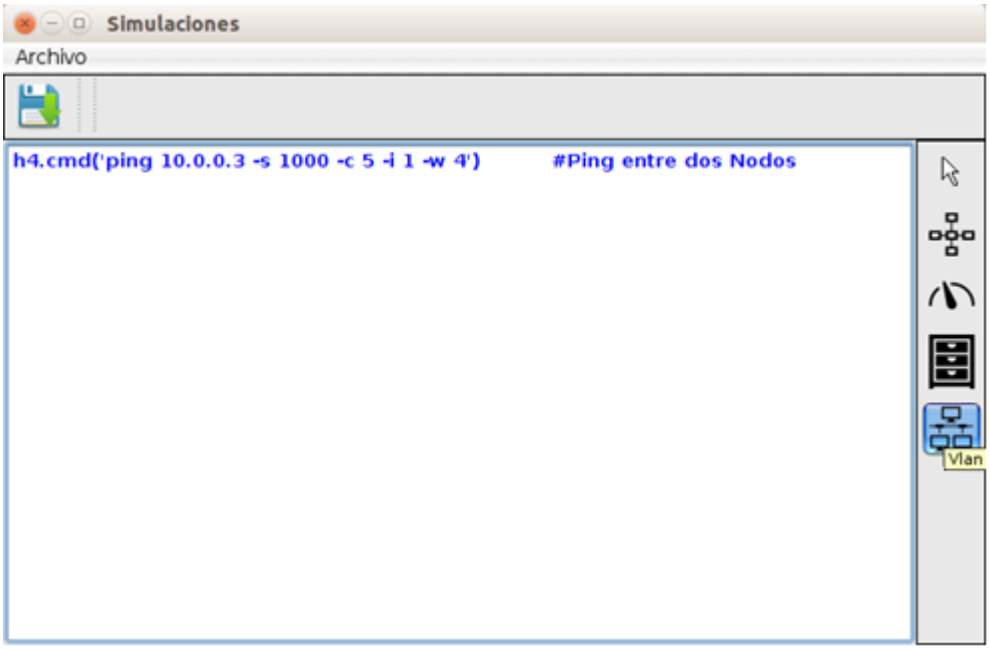


Figura 85. Prueba Vlan I

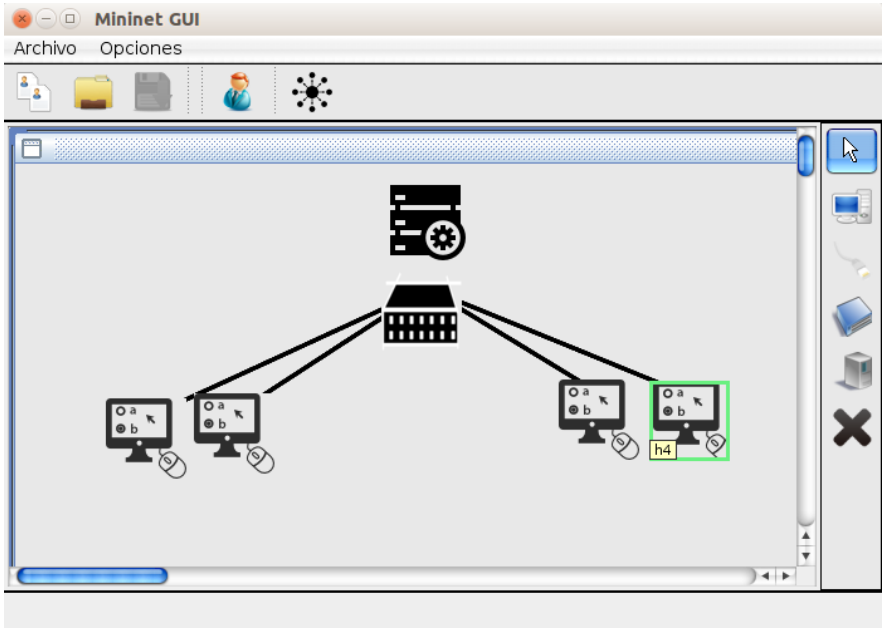


Figura 86. Prueba Vlan II

Figura 87. Prueba Vlan III

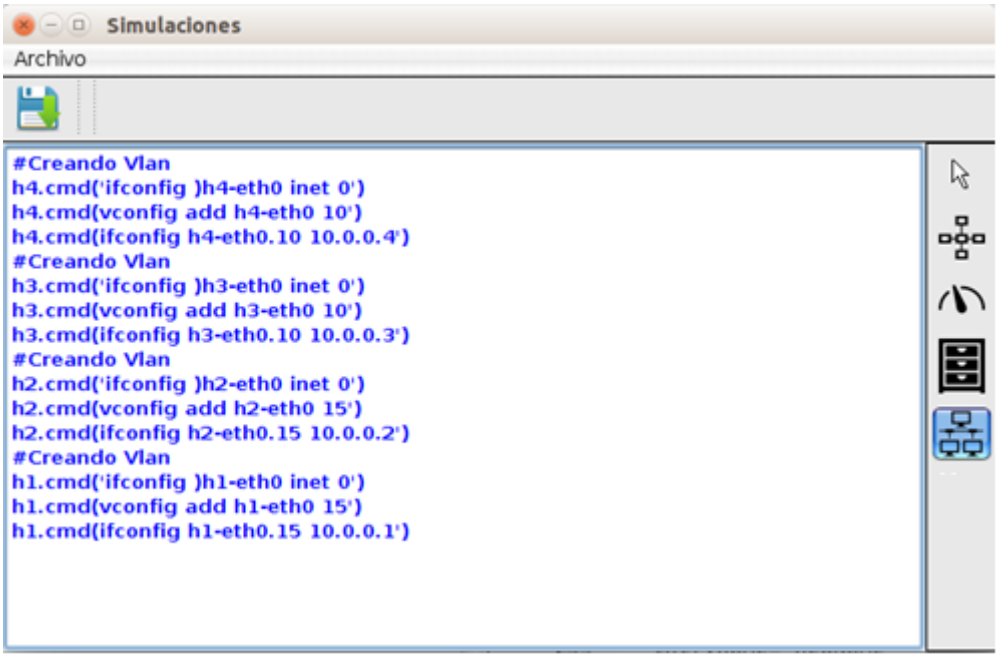


Figura 88. Prueba Vlan IV

Paso 5: Por último se procede a guardar la simulación, la cual se guardara en el archivo de la red actual para su posterior ejecución.

9.3 Prueba de Aplicación

Para el siguiente caso se utilizaran los siguientes códigos que permiten la utilizacion de un servidor chat y N clientes conectados a el.

Esto muestra lo poderoso que puede ser "Mininet" creando un servidor de chat virtual y 3 host virtuales conectados a el.

chat_server.py [11]

```
# chat_server.py

import sys
import socket
import select

HOST = '10.0.0.1'
SOCKET_LIST = []
RECV_BUFFER = 4096
PORT = 9009

def chat_server():
    if(len(sys.argv) <3):
      print 'Por favor ingrese la ip y el puerto del servidor'
        sys.exit()
    HOST = sys.argv[1]
    PORT = int(sys.argv[2])
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((HOST, PORT))
    server_socket.listen(10)

    # add server socket object to the list of readable connections
    SOCKET_LIST.append(server_socket)

    print "Chat server started on port " + str(PORT)

    while 1:

        # get the list sockets which are ready to be read through
select
        # 4th arg, time_out  = 0 : poll and never block
        ready_to_read,ready_to_write,in_error                    =
select.select(SOCKET_LIST,[],[],0)

        for sock in ready_to_read:
            # a new connection request recieved
            if sock == server_socket:
                sockfd, addr = server_socket.accept()
                SOCKET_LIST.append(sockfd)
```

```
                print "Client (%s, %s) connected" % addr


               broadcast(server_socket, sockfd, "[%s:%s] entered our
chatting room\n" % addr)

            # a message from a client, not a new connection
            else:
                # process data recieved from client,
                try:
                    # receiving data from the socket.
                    data = sock.recv(RECV_BUFFER)
                    if data:
                        # there is something in the socket
                        broadcast(server_socket, sock, "\r" + '[' +
str(sock.getpeername()) + '] ' + data)
                    else:
                        # remove the socket that's broken
                        if sock in SOCKET_LIST:
                            SOCKET_LIST.remove(sock)

                        # at this stage, no data means probably the
connection has been broken
                        broadcast(server_socket, sock, "Client (%s, %s)
is offline\n" % addr)

                # exception
                except:
                    broadcast(server_socket, sock, "Client (%s, %s) is
offline\n" % addr)
                    continue

    server_socket.close()

# broadcast chat messages to all connected clients
def broadcast (server_socket, sock, message):
    for socket in SOCKET_LIST:
        # send the message only to peer
        if socket != server_socket and socket != sock :
            try :
                socket.send(message)
            except :
                # broken socket connection
                socket.close()
                # broken socket, remove it
                if socket in SOCKET_LIST:
                    SOCKET_LIST.remove(socket)

if __name__ == "__main__":
    sys.exit(chat_server())
```

chat_client.py [11]

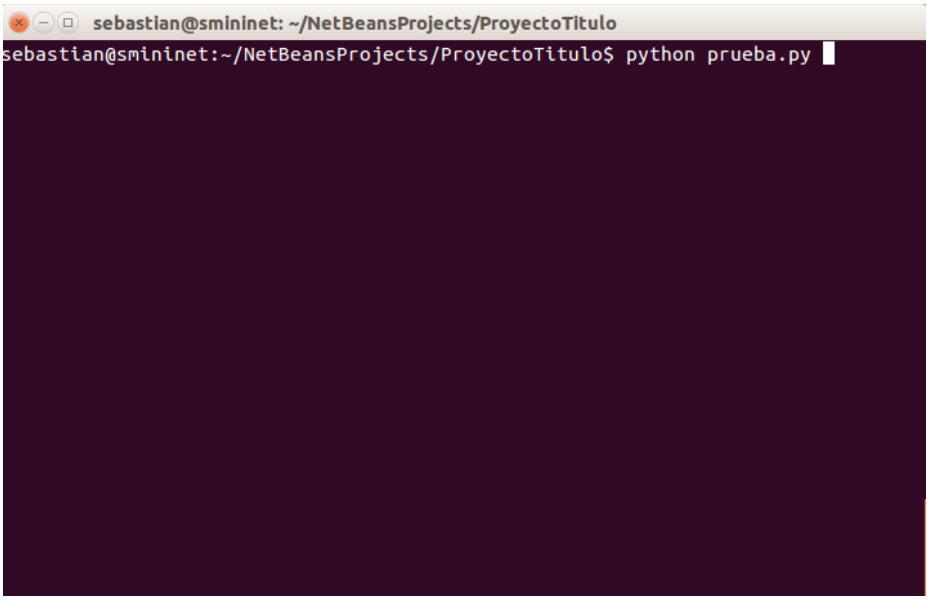
```
# chat_client.py

import sys
import socket
import select

def chat_client():
    if(len(sys.argv) < 3) :
        print 'Usage : python chat_client.py hostname port'
        sys.exit()

    host = sys.argv[1]
    port = int(sys.argv[2])

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(2)

    # connect to remote host
    try :
        s.connect((host, port))
    except :
        print 'Unable to connect'
        sys.exit()

    print 'Connected to remote host. You can start sending messages'
    sys.stdout.write('[Me] '); sys.stdout.flush()

    while 1:
        socket_list = [sys.stdin, s]

        # Get the list sockets which are readable
        ready_to_read,ready_to_write,in_error = select.select(socket_list , [], [])

        for sock in ready_to_read:
            if sock == s:
                # incoming message from remote server, s
                data = sock.recv(4096)
                if not data :
                    print '\nDisconnected from chat server'
                    sys.exit()
                else :
                    #print data
                    sys.stdout.write(data)
                    sys.stdout.write('[Me] '); sys.stdout.flush()

            else :
                # user entered a message
                msg = sys.stdin.readline()
```

```
        s.send(msg)
        sys.stdout.write('[Me] '); sys.stdout.flush()

if __name__ == "__main__":

  sys.exit(chat_client())
```

Paso 1: Ejecutar una red ya creada (prueba.py).

```
#!/usr/bin/ inine

from  ininet.net import Mininet
from  ininet.node import Controller, RemoteController, OVSController
from  ininet.node import CPULimitedHost, Host, Node
from  ininet.node import OVSKernelSwitch, UserSwitch
from  ininet.node import IVSSwitch
from  ininet.cli import CLI
from  ininet.log import setLogLevel, info
from  ininet.link import TCLink, Intf
from subprocess import call


def archivo():

    archivo = open('log.txt','a')
    archivo2 = open('log','r')
    datos = archivo2.readlines()
    archivo2.close()
    archivo.writelines(datos)
    archivo.close()

def myNetwork():

    net = Mininet(topo=None,build=False,ipBase='10.0.0.0/8')

    info( '—Creando Controlador\n' )
    c0
=net.addController(name='c0',controller=Controller,protocol='tcp',port=
6633)

    info( '—Creando Swtichs\n' )
    s0 =net.addSwitch('s0',cls=OVSKernelSwitch)

    info( '—Creando Hosts\n' )
    h1 =net.addHost('h1', cls=Host, ip='10.0.0.1', defaulRoute=None)
    h2 =net.addHost('h2', cls=Host, ip='10.0.0.2', defaulRoute=None)
    h3 =net.addHost('h3', cls=Host, ip='10.0.0.3', defaulRoute=None)
    h4 =net.addHost('h4', cls=Host, ip='10.0.0.4', defaulRoute=None)

    info( '—Creando Enlaces\n' )
```

```
    net.addLink(h2, s0)
    net.addLink(h1, s0)
    net.addLink(h3, s0)
    net.addLink(h4, s0)

    info( '-Iniciando Red\n' )
    net.build()
    info( '-Iniciando Controlador\n' )
    for controller in net.controllers:
        controller.start()

    info( '-Iniciando Switchs\n' )
    net.get('s0').start([c0])
    info( '-Iniciando Consola de Comandos\n')
    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```




```
python prueba.py
```

Figura 89. Ejecucion prueba.py

Paso 2: Dentro de Mininet abrir 4 terminales para la ejecucion de las aplicaciones, para aquello utilizaremos el comando xterm.

```
xterms h1 h2 h3 h4
```

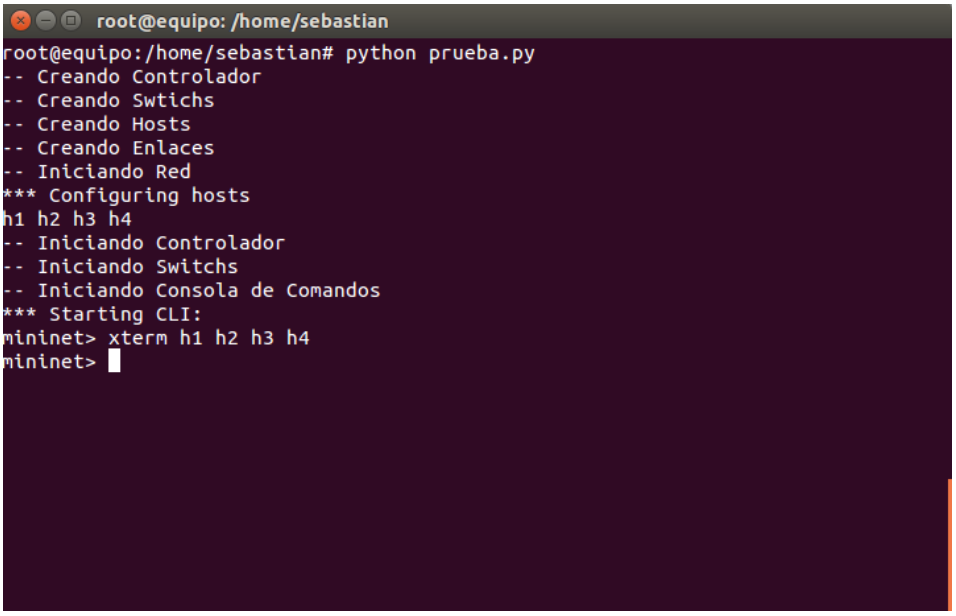


Figura 90. Ejecucion Xterms

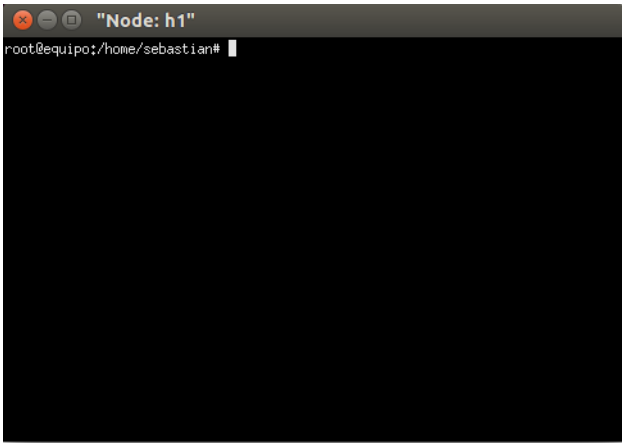


Figura 91. Xterm Nodo h1

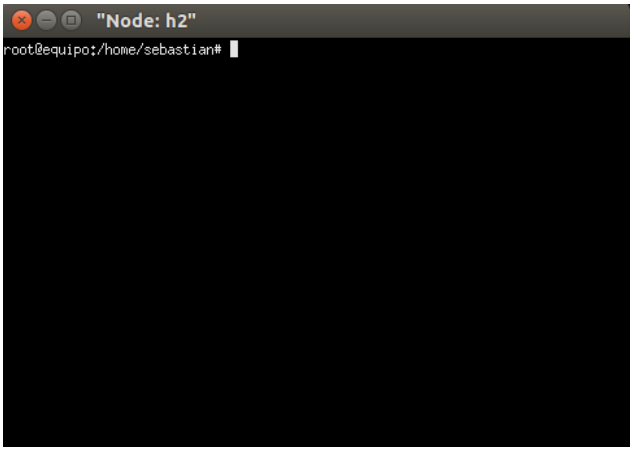


Figura 92. Xterm Nodo h2

Figura 93. Xterm Nodo h3

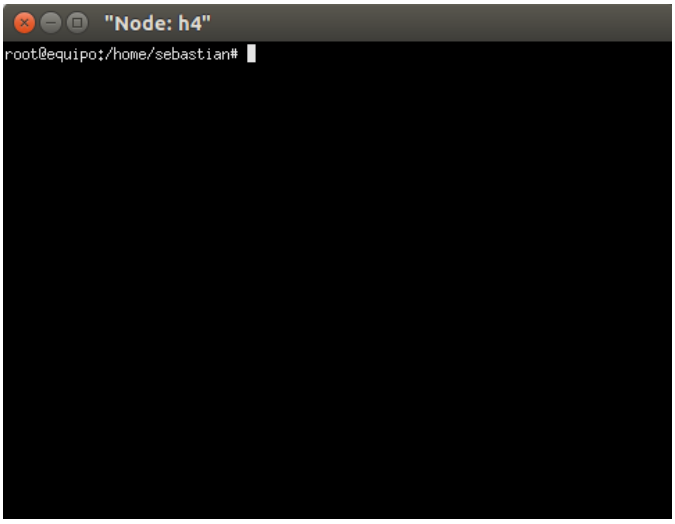


Figura 94.Xterm Nodo h

Paso 3: En la terminal del nodo h4 (10.0.0.4) ejecutar la aplicación chat_server.py, para ello debemos pasar por parametro la ip del host y el puerto en donde se correra nuestro servidor de chat, para este ejemplo usaremos el puerto 9000.

```
python chat_server.py 10.0.0.4 9000
```

Figura 95. Ejecucion chat_server.py

Paso 4: Luego procederemos a ejecutar el codigo clientes chat_server.py en los host h1 h2 h3, para ello ejecutaremos en cada una de las consolas el siguiente codigo indicando la ip 10.0.0.4 del servidor y el puerto 9000.

```
python chat_client.py 10.0.0.4 9000
```

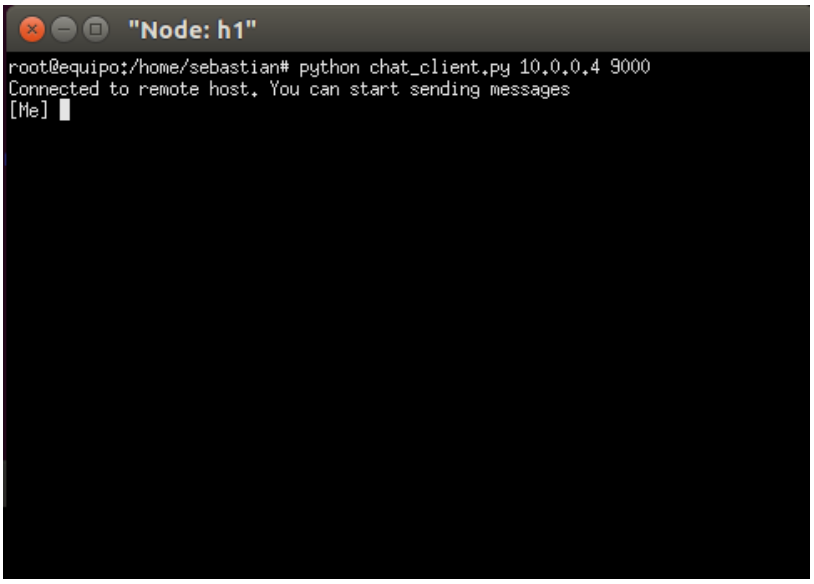


Figura 96. Ejecucion chat_client.py

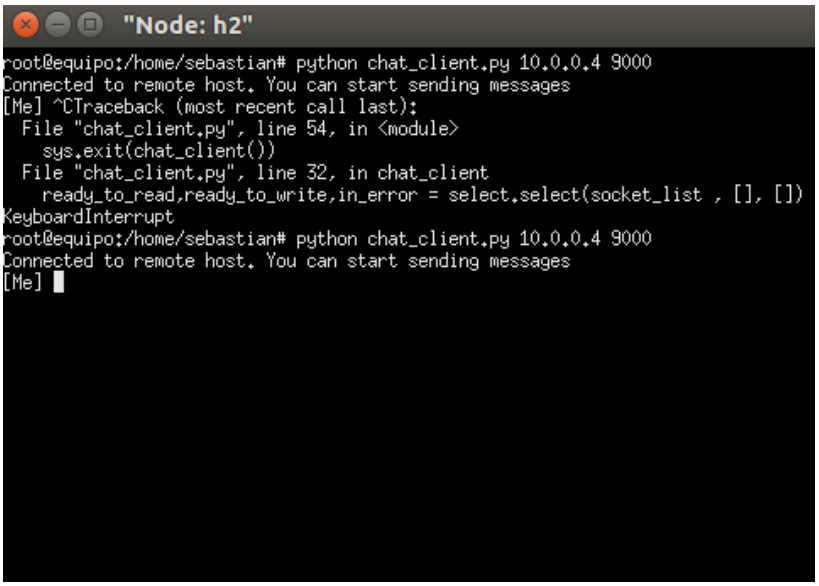


Figura 97. Ejecucion chat_client.py

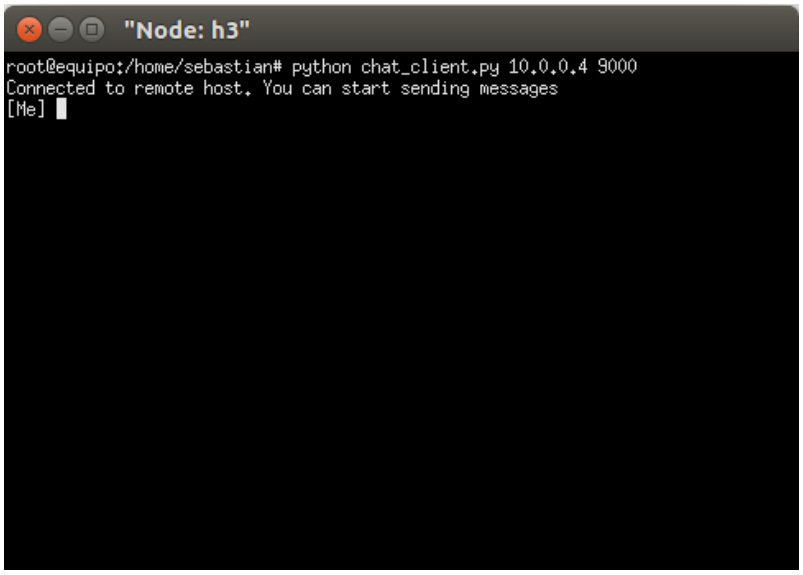


Figura 98. Ejecucion chat_client.py

Paso 5: Ya ejecutados los clientes y servidores en los respectivos hosts, ya es posible enviar mensajes entre los hosts virtuales conectados al servidor de chat. Como lo muestran las figuras 99 y 100.

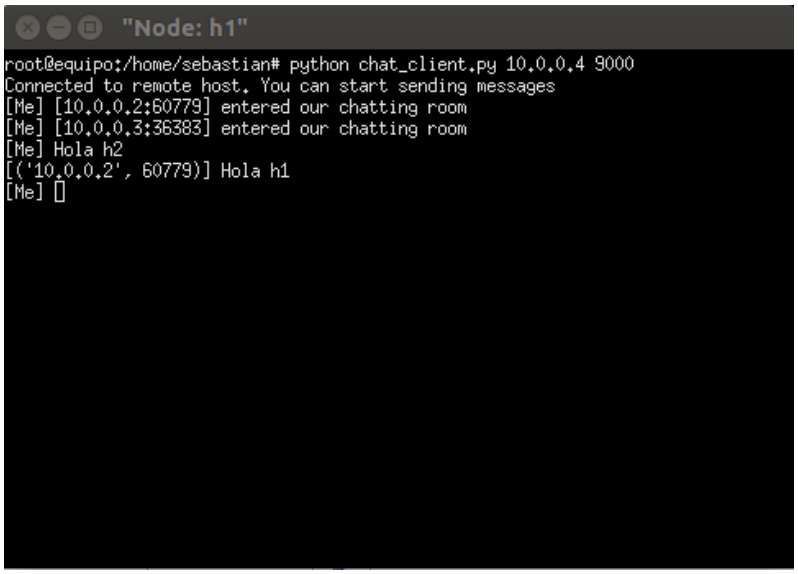


Figura 99. Chat Entre hosts I

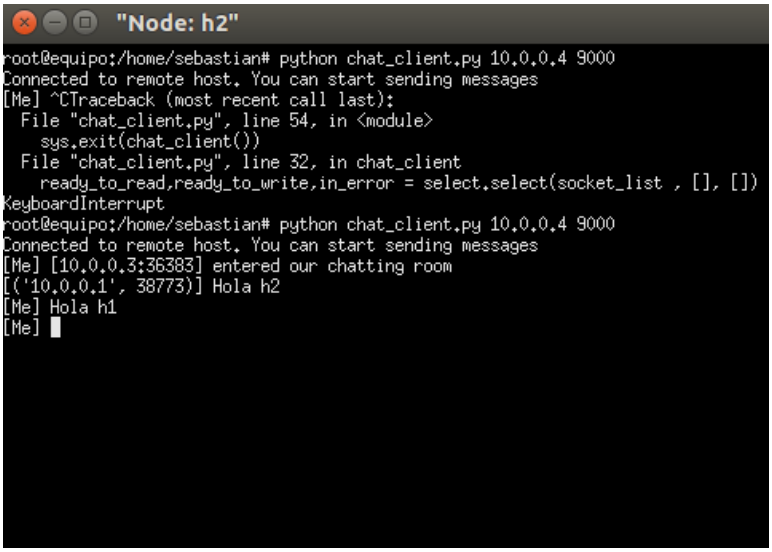


Figura 100. Chat Entre hosts II

10 CONCLUSIONES

Con base a los objetivos planteados durante el desarrollo del proyecto, es posible concluir que:

- Se desarrolló una interfaz gráfica en Java capaz de generar scripts compatibles con "Mininet". Esto se logró mediante un profundo análisis y estudio previo de la herramienta "Mininet" antes del desarrollo del proyecto. Esta interfaz es capaz de generar toda una red (Hosts, switchs, enlaces y controladores) con campos editables en cada uno de sus nodos y generar simulaciones sobre sus hosts.

- Se logra mejorar la documentación de la API existente de "Mininet". Esto se logró describiendo de mejor forma métodos, constructores, funciones, parámetros, tipo de datos de entrada (inexistentes con anterioridad), retorno de métodos (inexistentes con anterioridad) y funciones.

- Se efectúan simulaciones de diferentes escenarios de redes de ordenadores utilizando los script generados, estas simulaciones van desde emulación de vlans, pruebas de conectividad avanzadas a través del comando ping y la prueba de un servidor de chat en una red virtual local.

- Gracias al estudio de la herramienta se logra desarrollar un generador de script para redes rápidas de gran tamaño por medio del comando mn.

A nivel académico se logró implementar los conocimientos otorgados por la carrera, estos permitieron un rápido análisis y compresión a pesar del desconocimiento del software "Mininet" que poseía una escaza documentación.

El presente trabajo puede servir de base en el futuro para desarrollar otro proyecto enfatizado en la investigación de Openflow y toda la arquitectura de SDN, de esta manera, se puede profundizar en el estudio e innovación de las redes de computadoras haciendo uso de la herramienta "Mininet".

11 BIBLIOGRAFÍA

Formato de referencias y bibliografía según los estándares de biblioteca.

1. Universidad del Bio Bio. Recuperado 12, 2014, de www.ubb.cl
2. Cgroup Recuperado 11,2014 de https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt
3. Cget Recuperado 11,2014 de http://linux.die.net/man/1/cgget
4. CFS Scheduler Recuperado 11, 2014 de https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt
5. RT Scheduler Recuperado 11,2014 de https://www.kernel.org/doc/Documentation/scheduler/sched-rt-group.txt
6. Mount. Recuperado 11,2014 , de http://es.wikipedia.org/wiki/Mount
7. Ifconfig. Recuperado 11/2014 de https://www.hscripts.com/tutorials/linux-commands/ifconfig.html
8. Torus Network. Recuperado 12/2014 de http://en.wikipedia.org/wiki/Torus_interconnect
9. OpenFlow SDN, Recuperado 01/2014 de http://www.xinguard.com/en/content.aspx?id=70
10. Archivo OpenFLow, Recuperado 01/2014 de www.archive.openflow.org
11. Codigo cliente servidor, Recuperado 01/2014 de http://www.bogotobogo.com
12. Mininet, Recuperado 01/2014 de http://mininet.org/
13. Mininet Mailist, Recuperado 01/2014 de https://mailman.stanford.edu/pipermail/mininet-discuss/
14. OpenFlow, Recuperado 01/2014 de http://archive.openflow.org/wp/learnmore/
15. SDN, Recuperado 02/2015 de https://www.opennetworking.org/sdn-resources/sdn-definition
16. Creador de Mininet, Recuperado 02/2015 de https://www.linkedin.com/in/boblantz1
17. SDN, Recuperado 02/2015 de http://es.wikipedia.org/wiki/Redes_definidas_por_software
18. Mininet Releases, Recuperado 01/2015 de https://github.com/mininet/mininet/releases
19. Mininet Information, Recuperado 01/2015 de http://whois.domaintools.com/mininet.org
20. Google aplication SDN, Recuperado 01/2015 de http://diarioti.com/google-explica-las-ventajas-de-haber-incorporado-sdn-y-nfv-en-su-propia-nube/82876

12 ANEXO: PLANIFICACION INICIAL DEL PROYECTO

Para definir las actividades a realizar para lograr los objetivos se realizó una Carta Gantt, en la cual se especifica los tiempos planificados a utilizar en cada tarea, a continuación se presenta dicha planificación, con las correspondientes fechas de inicio y fin de actividad.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1		**Proyecto de titulo**	**77 días**	**mar 26-08-14**	**mié 10-12-14**		
2		**Estado del arte**	**20 días**	**mar 26-08-14**	**lun 22-09-14**		
3		Investigación uso Mininet	10 días	mar 26-08-14	lun 08-09-14		
4		Investigación uso API	10 días	mar 09-09-14	lun 22-09-14	3	
5		**Generación de scripts creación**	**15 días**	**mar 23-09-14**	**lun 13-10-14**	**2**	
6		Diseño	9 días	mar 23-09-14	vie 03-10-14		
7		Codificación	3 días	lun 06-10-14	mié 08-10-14	6	
8		Pruebas	3 días	jue 09-10-14	lun 13-10-14	7	
9		**<Entregable I>**	0 días	mar 14-10-14	mar 14-10-14	5	
10		**Interfaz base**	**13 días**	**mar 14-10-14**	**jue 30-10-14**	**5**	
11		Diseño	5 días	mar 14-10-14	lun 20-10-14		
12		Codificación	5 días	mar 21-10-14	lun 27-10-14	11	
13		Pruebas	3 días	mar 28-10-14	jue 30-10-14	12	
14		**<Entregable II>**	0 días	jue 30-10-14	jue 30-10-14		
15		**Generación de scripts de simulación**	**11 días**	**vie 31-10-14**	**vie 14-11-14**	**10**	
16		Diseño	5 días	vie 31-10-14	jue 06-11-14		
17		Codificación	3 días	vie 07-11-14	mar 11-11-14	16	
18		Pruebas	3 días	mié 12-11-14	vie 14-11-14	17	
19		**<Entregable III>**	0 días	vie 14-11-14	vie 14-11-14		
20		**Recuperación y filtrado de datos**	**13 días**	**lun 17-11-14**	**mié 03-12-14**	**15**	
21		Diseño	7 días	lun 17-11-14	mar 25-11-14		
22		Codificación	4 días	mié 26-11-14	lun 01-12-14	21	
23		Pruebas	2 días	mar 02-12-14	mié 03-12-14	22	

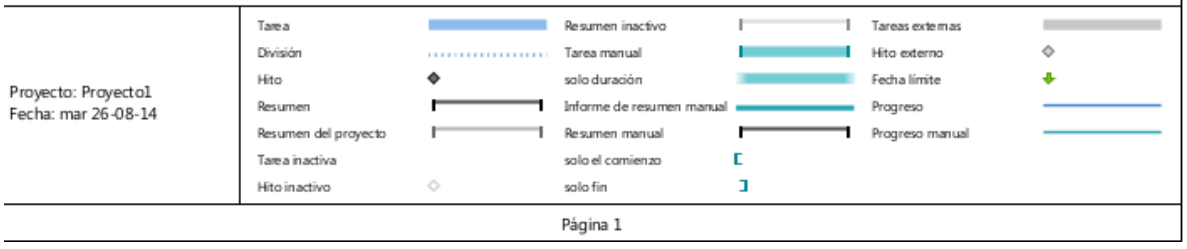
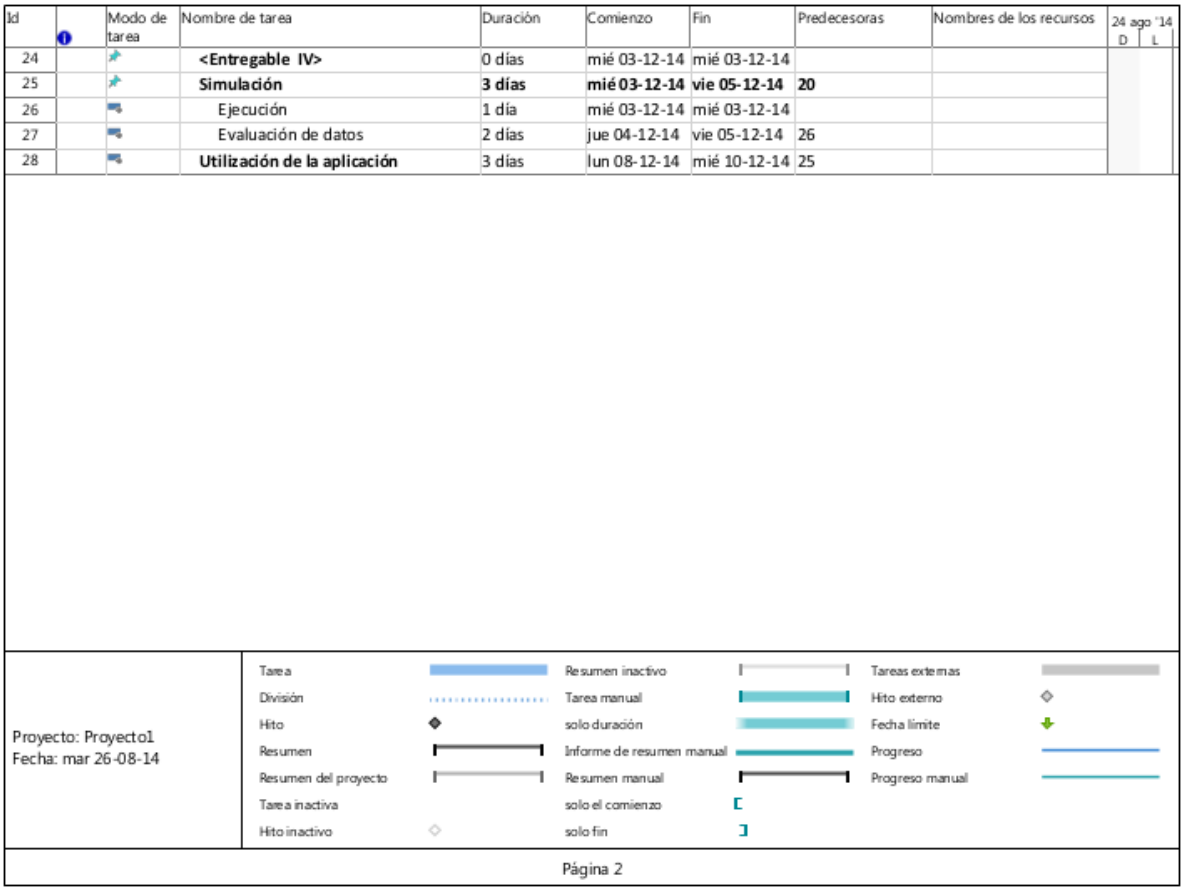


Figura 101. Carta Gantt I



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
24		**<Entregable IV>**	0 días	mié 03-12-14	mié 03-12-14		
25		**Simulación**	**3 días**	**mié 03-12-14**	**vie 05-12-14**	**20**	
26		Ejecución	1 día	mié 03-12-14	mié 03-12-14		
27		Evaluación de datos	2 días	jue 04-12-14	vie 05-12-14	26	
28		**Utilización de la aplicación**	3 días	lun 08-12-14	mié 10-12-14	25	

Figura 102. Carta Gantt II

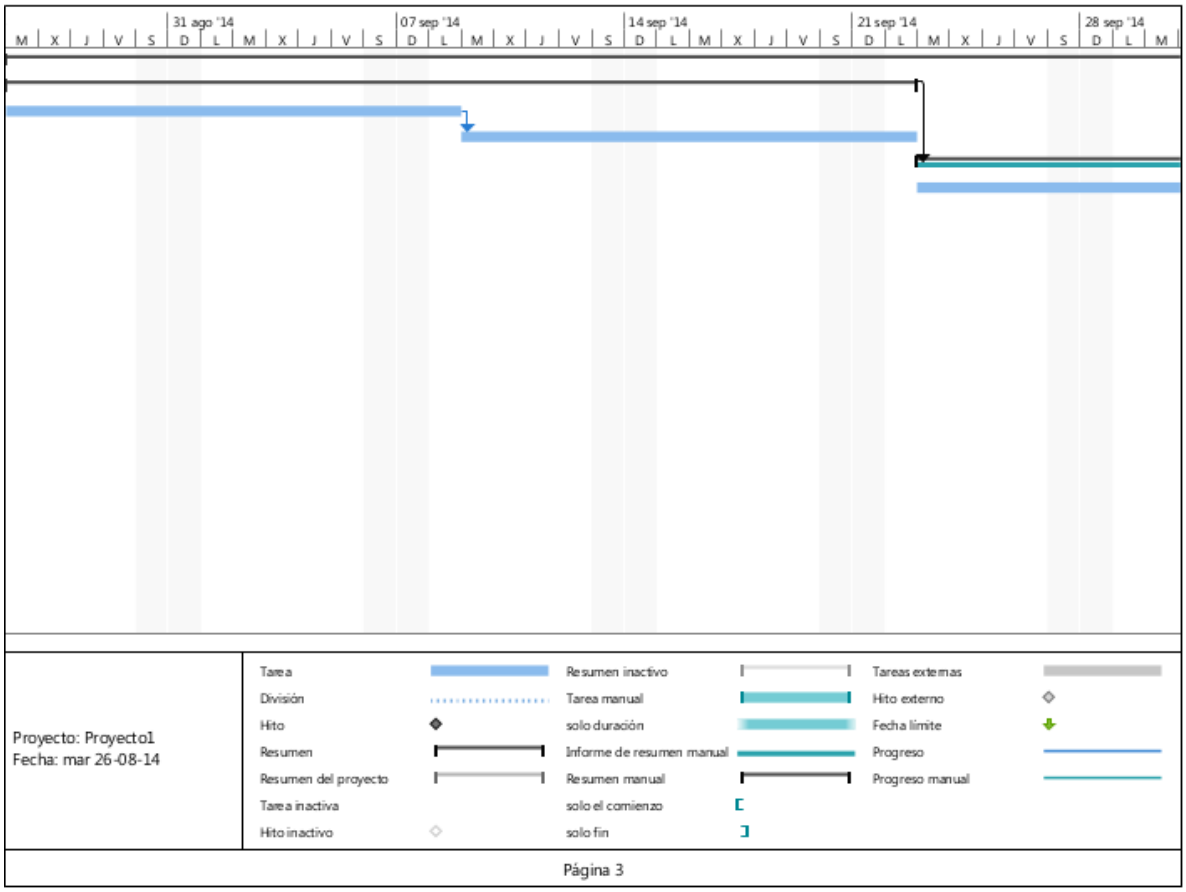


Figura 103. Carta Gantt III

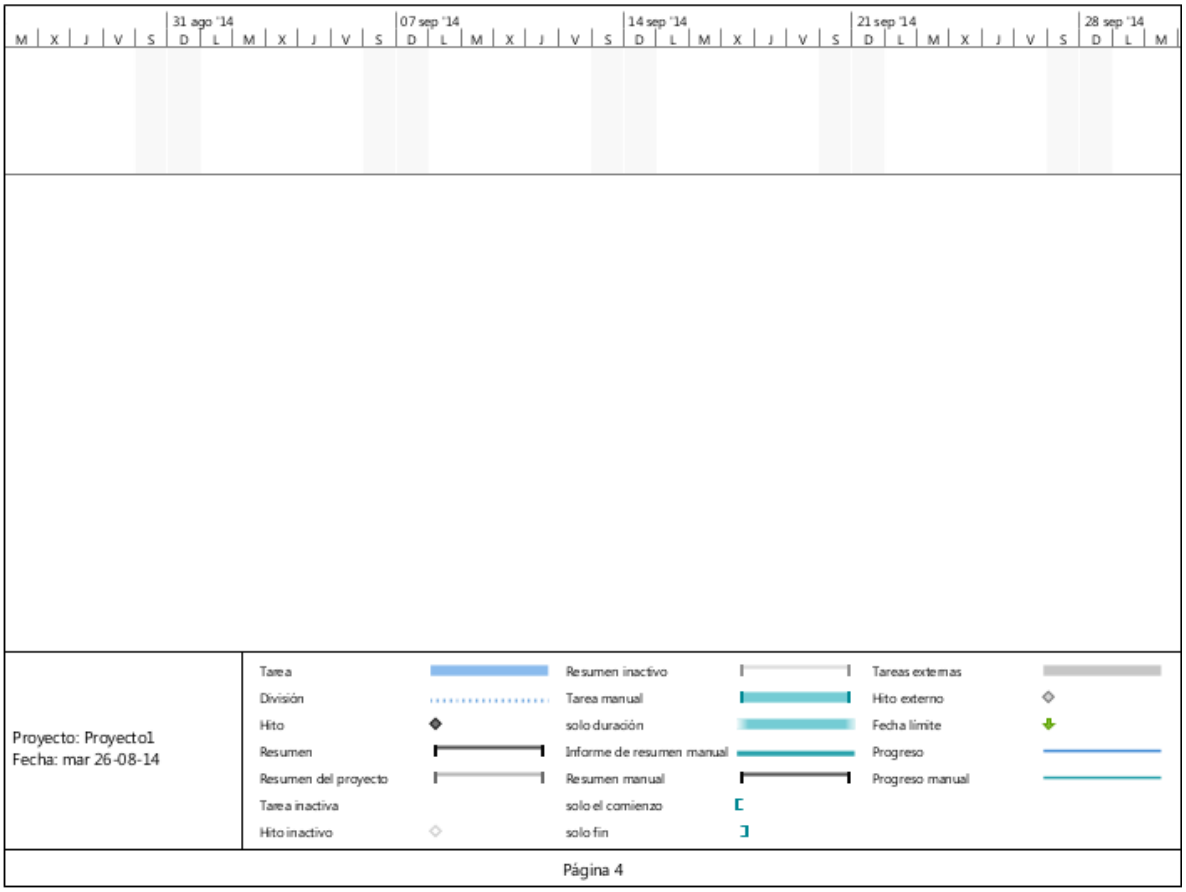


Figura 104. Carta Gantt IV

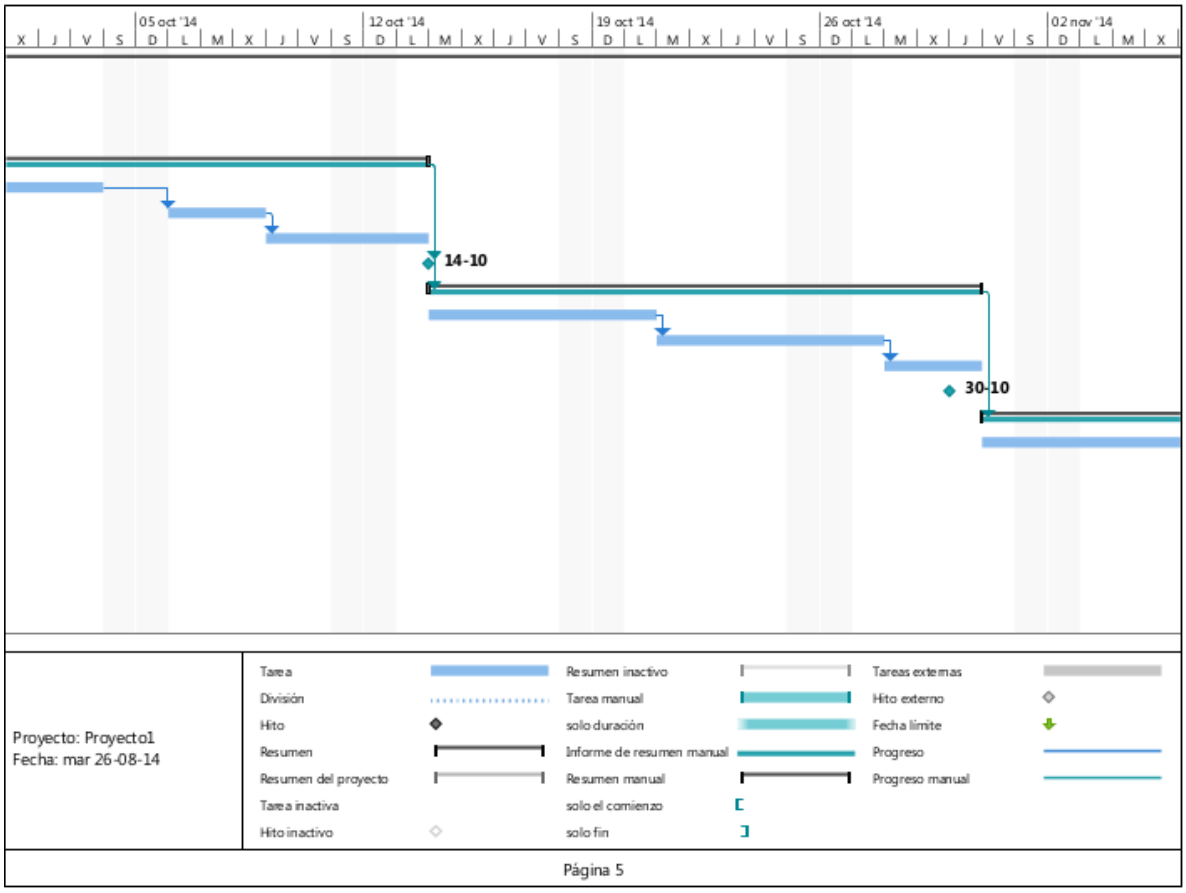


Figura 105. Carta Gantt V

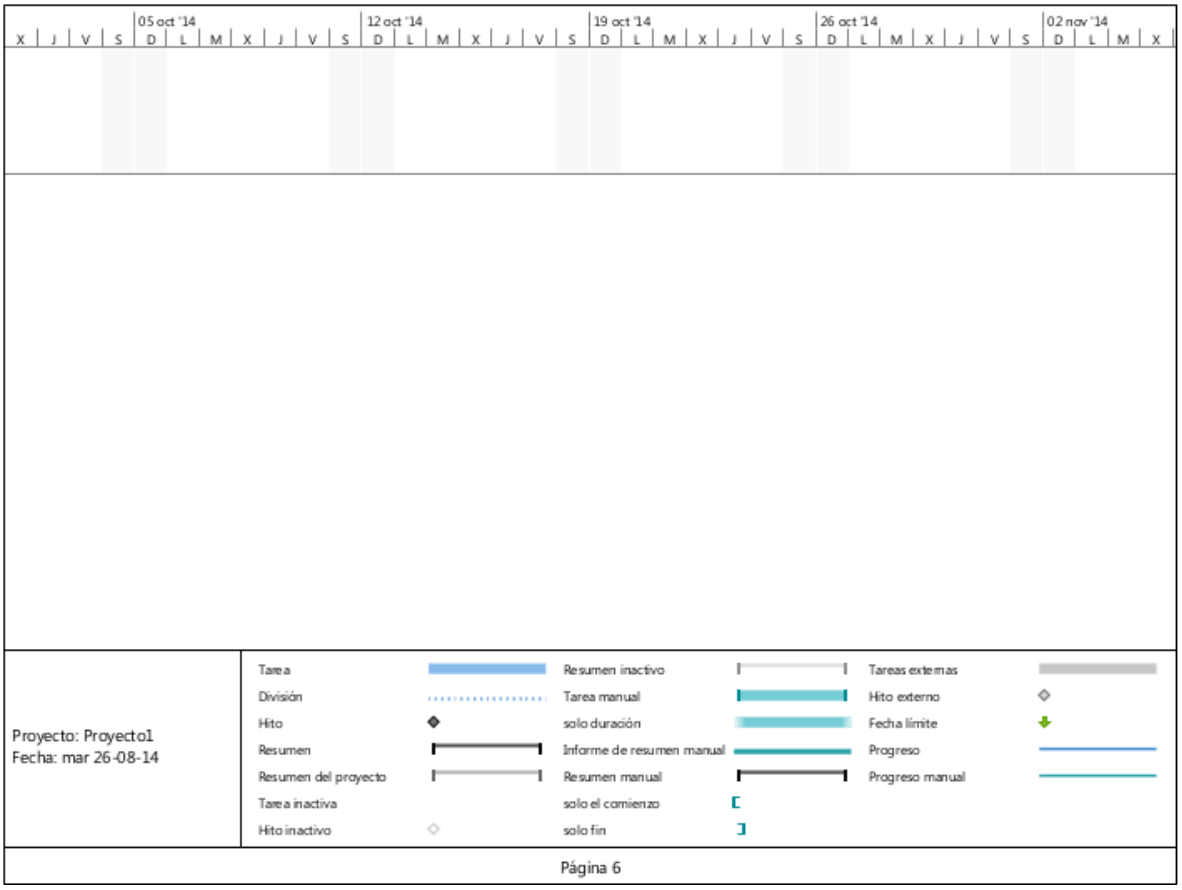


Figura 106. Carta Gantt VI

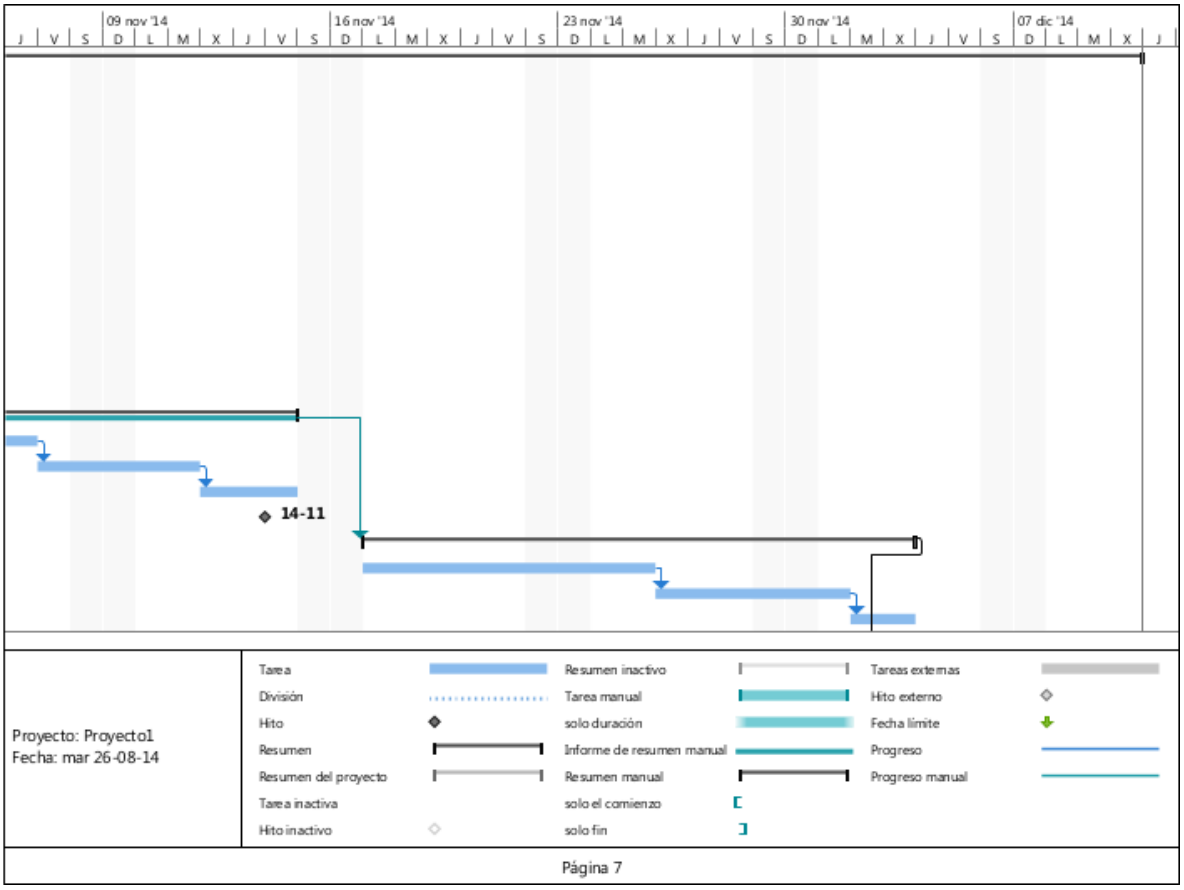


Figura 107. Carta Gantt VII

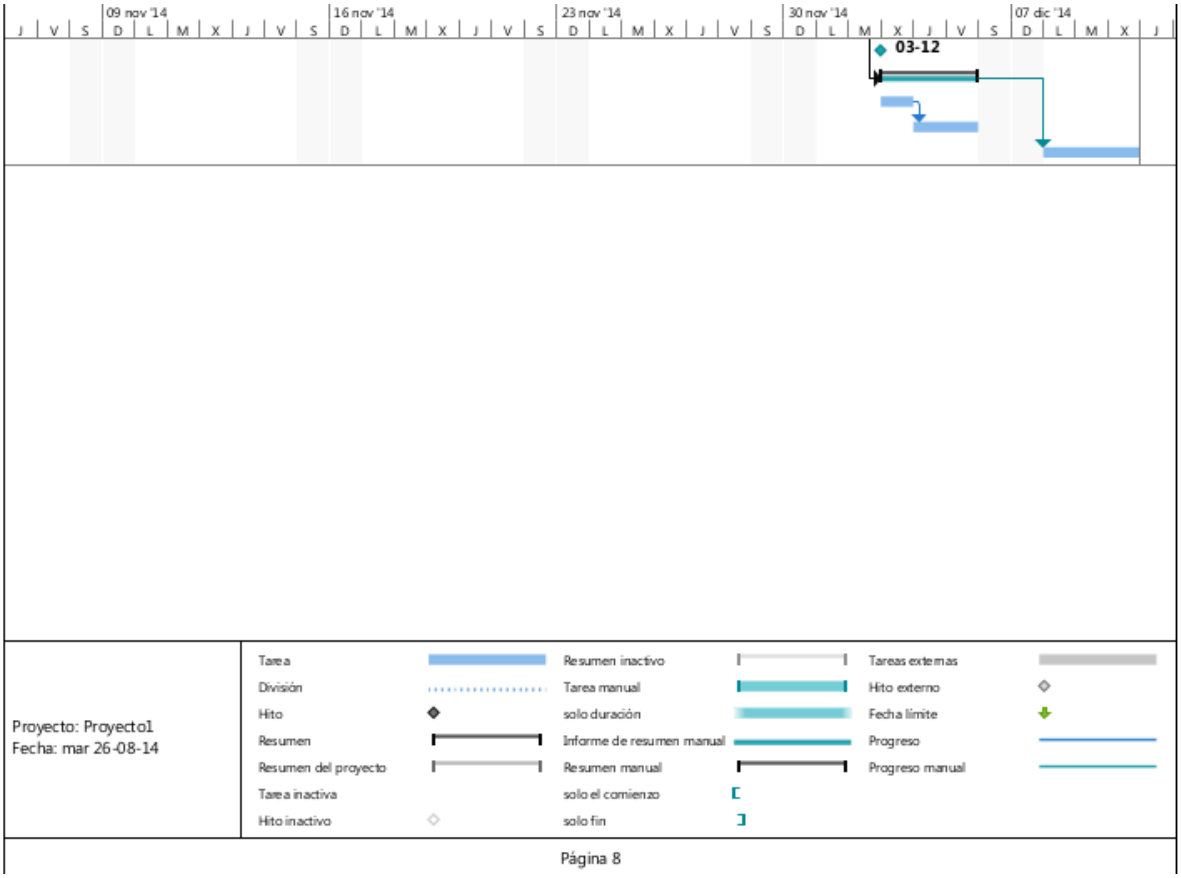


Figura 108. Carta Gantt VIII

13 ANEXO: MANUAL DE USUARIO

13.1 Instalación Mininet

Para la instalación de Mininet se requiere el uso de una máquina virtual como virtual box o un sistema operativo Ubuntus 12.02 o superior (Recomendado).

Para efectos de rendimiento se aconseja la utilización de un sistema operativo Ubuntus 14.0.4 LTS limpio (recien instalado) sobre un ordenador. Se debe instalar git para la correcta instalación de Mininet ejecutando el siguiente código en una terminal.

```
sudo apt-get install git
sudo git clone git://github.com/mininet/Mininet
```

Luego dirigirse a la carpeta Mininet e instalarlo con los siguientes comandos

```
cd mininet/util
chmod 777 install.sh
./install.sh -a
```

Para que las vlans se han ejecutadas de forma correcta es necesario la instalación del siguiente paquete.

```
sudo apt-get install vlan
```

Para la compilacion del codigo fuente es necesario instalar un IDE como Netbeans (JDK 8u40 with NetBeans 8.0.2), primero es necesario descargar el IDE en el siguiente enlace:

http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html (Es importante elegir la arquitectura correcta correspondiente al sistema operativo, x86 para 32 bits y x64 para 64 bits)

Para ejecutar el archivo ejecutable dist es necesaria la instalación de Java Runtime Enviroment 8u40, este se puede descargar del siguiente enlace: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html.

Los siguientes enlaces incorporan guias prácticas para la instalación del IDE y del JRE_

- Guia Instalacion IDE: http://ubuntuhandbook.org/index.php/2014/03/install-netbeans-ide-8-0-in-ubuntu-14-0413-1012-04/
- Guia Instalacion JRE: http://www.wikihow.com/Install-Oracle-Java-JRE-on-Ubuntu-Linux

Esto instalara Mininet de forma completa.

El correcto orden de estos pasos permitira la correcta compilacion y ejecucion del software en donde el ejectuble dist puede ser ejecutado en una consola de comandos Linux con el siguiente comando:

```
java –jar [Nombre del archivo dist]
```

O bien compilar y ejecutar el código en el IDE Netbean 8.0.2.