ANEXO A. Mininet

En este apéndice se describe el proceso de instalación llevado a cabo para poder utilizar el *software* de Mininet, necesario para el desarrollo de este TFG. De esta manera, se pretende ofrecer la posibilidad de replicar este trabajo si se desea.

A.1. Instalación

Se han contemplado cuatro posibles modos de instalación presentados a continuación.

Opción 1: Instalación de Mininet VM

Es la forma más fácil de instalar Mininet y la recomendada. Únicamente debe descargarse la imagen de la máquina virtual e importarla con cualquier software de virtualización como VirtualBox.

Opción 2: Instalación nativa

En este método se asume que se cuenta con una máquina virtual Ubuntu limpia, sobre la que se instalará el software de Mininet. Se recomienda encarecidamente el uso de las versiones más recientes de Ubuntu, ya que soportan las últimas versiones de Open vSwitch.

Para instalar de forma nativa, primero debe descargarse el código fuente:

```
git clone git://github.com/mininet/mininet
```

Mediante este código se descargará la última versión disponible de Mininet (recomendado). Para instalar cualquier otra versión de este software deberá indicarse explícitamente. Para más información visitar la web oficial de Mininet[1].

Una vez descargado, se debe entrar en su directorio raíz y proceder con la instalación mediante el siguiente comando:

```
mininet/util/install.sh [options]
```

Las opciones disponibles corresponden con los distintos paquetes que se pueden instalar

[1] http://mininet.org/download/

Opción 3: Instalación a partir de paquetes

Esta opción está disponible si se está utilizando una versión reciente de Ubuntu. Puede ser que no se proporcione la última versión disponible de Mininet. Antes de proceder con la instalación, debe eliminarse cualquier rastro de versiones anteriores de Mininet y de Open vSwitch.

Opción 4: Actualizando una instalación de Mininet existente

Si no se ha hecho ningún cambio en Mininet, basta con ejecutar los siguientes comandos:

```
cd mininet
git fetch
git checkout master
git pull
```

Cabe destacar que estos comandos tan solo actualizan Mininet. Otros componentes como Open vSwitch deben ser actualizados por separado.

A.2. Configuración de red

A continuación, se explica cómo debe configurarse la máquina virtual de Mininet para tener conectividad. Basta con dotar de dos adaptadores de red a la máquina virtual de Mininet. Para ello, se debe seleccionar el apartado de `Red` en la configuración de la VM de Mininet en VirtualBox. El primero de ellos debe ser un adaptador de tipo `NAT` (Figura A-1), que dotará a la máquina de conexión a internet. El segundo adaptador a configurar es del tipo `Adaptador sólo-anfitrión` (Figura A-2Figura A-2), que permiten la conectividad entre la máquina *host* y las máquinas virtuales asociadas a ese mismo adaptador. Este adaptador será utilizado para realizar conexiones *ssh* a Mininet.

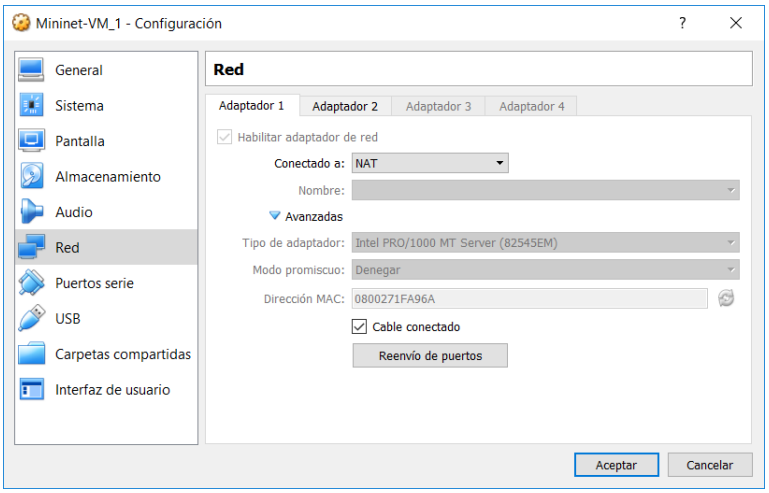

Figura A-1. Configuración de adaptador tipo NAT en Mininet

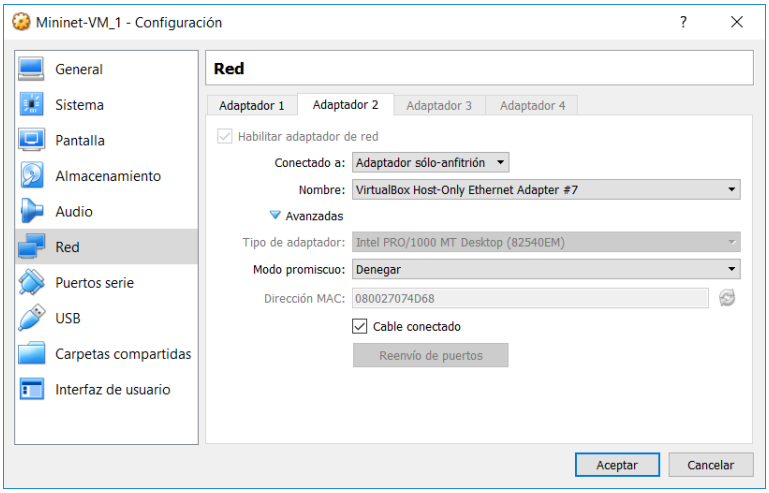

Figura A-2. Configuración de adaptador "Solo-anfitrión" en Mininet

A.3. Conexión mediante *ssh* y apertura de programas con interfaz gráfica

Para hacer posible la conexión por *ssh* desde Windows a la máquina virtual de Mininet y poder ejecutar programas con interfaz gráfica deben instalarse dos programas: PuTTy[2] y Xming.

PuTTy es un cliente *ssh* con licencia libre que hará posible esa conexión con Mininet. Tras descargarlo, debe realizarse la configuración de algunos parámetros.

En la pestaña `Session` (a la izquierda de la ventana inicial), debe indicarse el host al que se quiere conectar y el tipo de conexión (Figura A-3Figura A-3).

[2] https://www.putty.org/

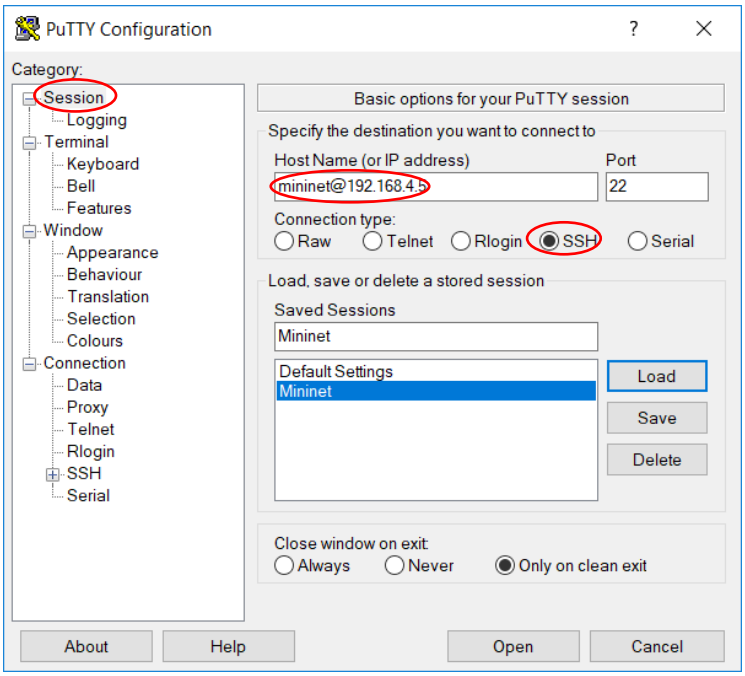

Figura A-3. Ventana "Session" en PuTTy

A continuación, en `Connection > SSH > X11` debe habilitarse la casilla de `Enable X11 forwarding` (Figura A-4). Esto es muy importante, ya que sin marcar esa casilla el sistema X-window no podrá encontrar el sistema para mostrar la pantalla.

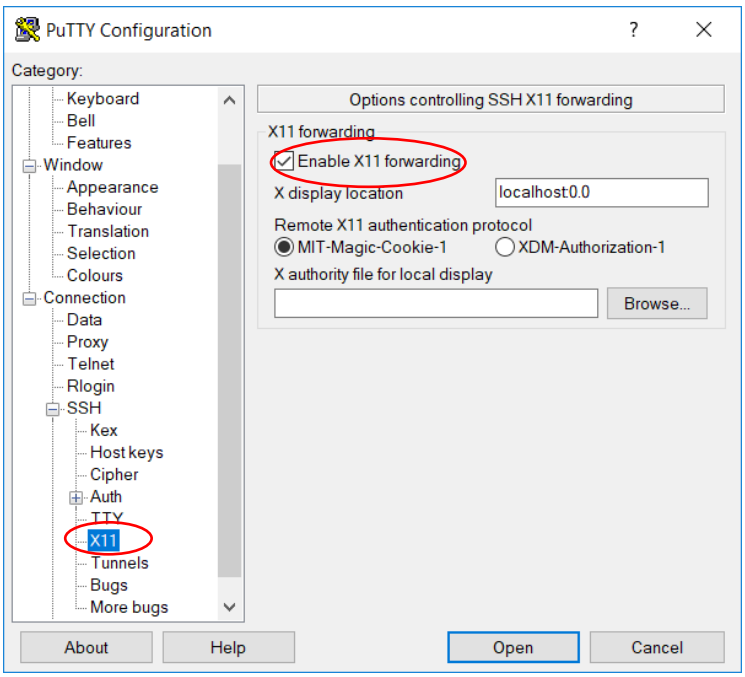

Figura A-4. Ventana "X11" en PuTTy

Una vez instalado Xming, debe lanzarse la aplicación Xlaunch y configurarlo como se muestra a continuación (Figura A-5, Figura A-6, Figura A-7, Figura A-8):

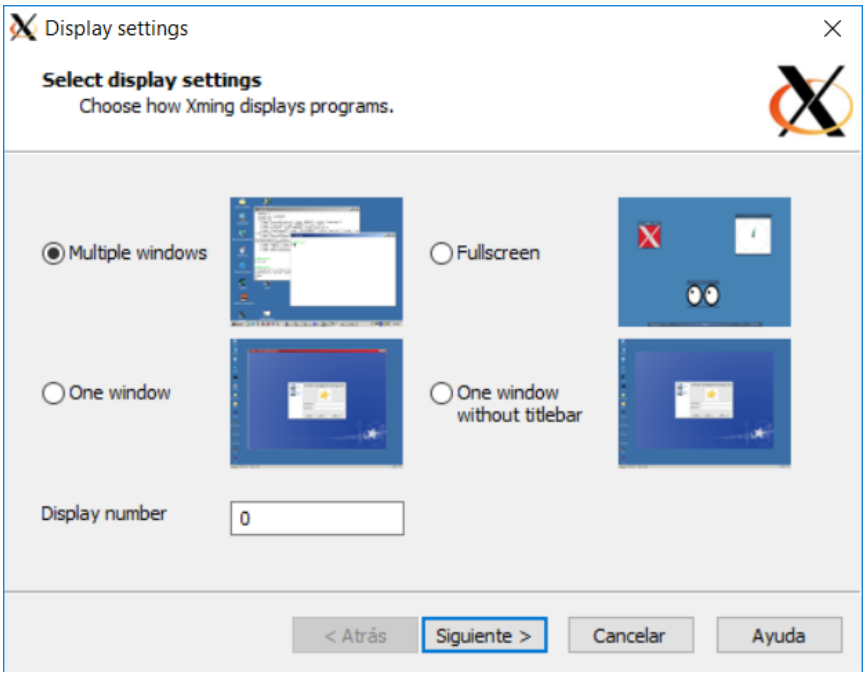

Figura A-5. Ventana 1 de configuración de XLaunch

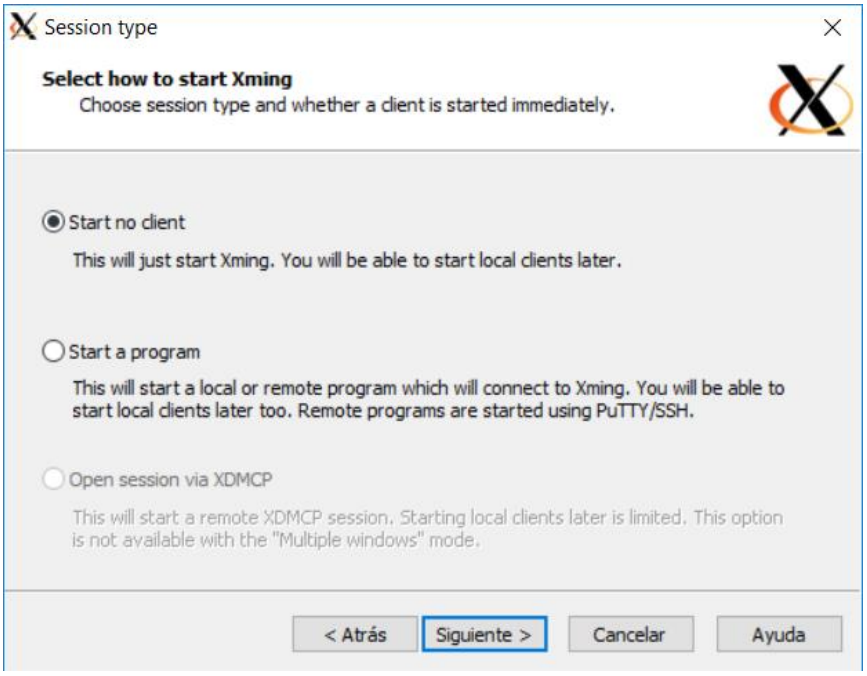

Figura A-6. Ventana 2 de configuración de XLaunch

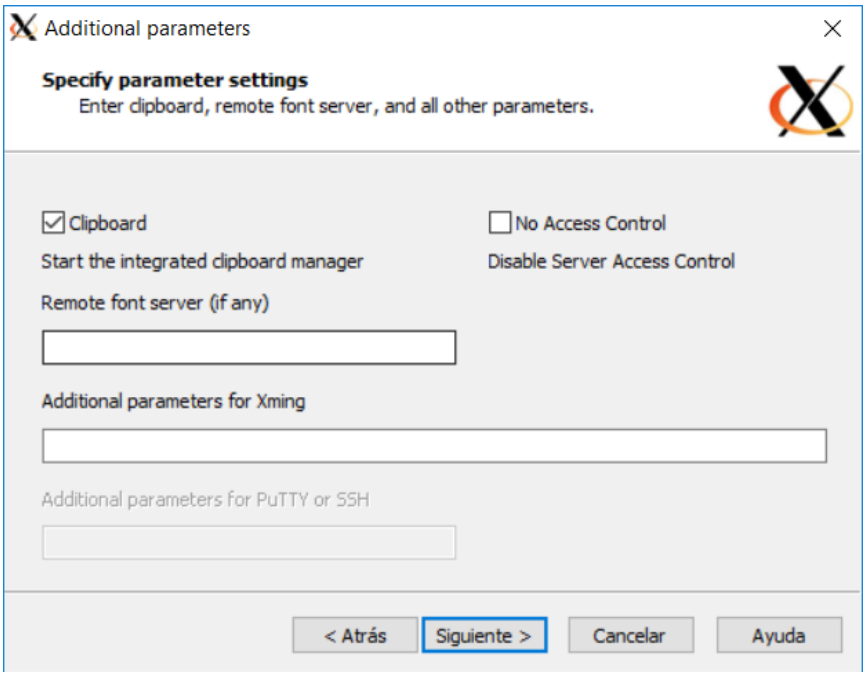

Figura A-7. Ventana 3 de configuración de XLaunch

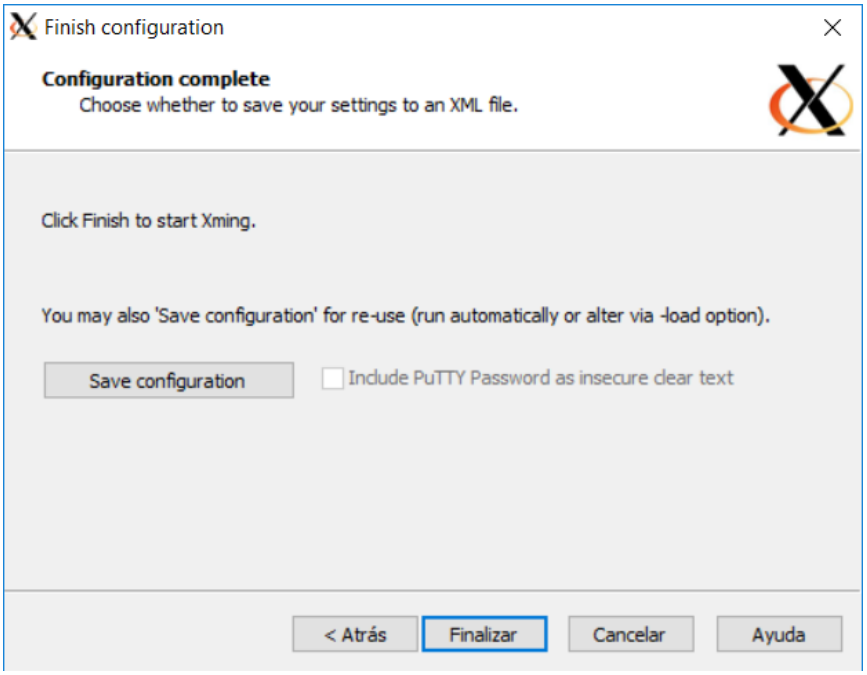

Figura A-8. Ventana 4 de configuración de XLaunch

Si se quiere utilizar alguna herramienta con interfaz gráfica debe lanzarse Xming y conectarse con PuTTy a Mininet. Una vez hecho esto, al ejecutar los comandos necesarios para abrir programas con interfaz gráfica como Miniedit o Wireshark se abrirá una ventana con el programa en cuestión.

A.4. Script de ejemplo

```
#!/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.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call
import socket
import time
import os

def myNetwork():
        #Creacion de socket para conectar con STEM
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        address = ('192.168.4.1', 12001 )

        net = Mininet(topo=None, build=False, ipBase='10.0.0.0/8', autoStaticArp=True)

        info('***Agregando controlador ODL\n')
        OpenDayLight = net.addController(name='OpenDayLight', controller=RemoteController,
                        ip='192.168.4.4', protocol='tcp', port=6633)

        info('***Agregando switches\n')
        s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
        s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
        s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
        s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
        s5 = net.addSwitch('s5', cls=OVSKernelSwitch)

        info('***Agregando hosts\n')
        h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
        h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
        h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
        h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)
        h5 = net.addHost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)

        info('***Agregando enlaces\n')
        net.addLink(s1,s2)
        net.addLink(s1,s5)
        net.addLink(s2,s5)
        net.addLink(s2,s3)
        net.addLink(s3,s4)
        net.addLink(s4,s5)
```

```
    info('***Agregando switches\n')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
    s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
    s5 = net.addSwitch('s5', cls=OVSKernelSwitch)

    info('***Agregando hosts\n')
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
    h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)
    h5 = net.addHost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)

    info('***Agregando enlaces\n')
    net.addLink(s1,s2)
    net.addLink(s1,s5)
    net.addLink(s2,s5)
    net.addLink(s2,s3)
    net.addLink(s3,s4)
    net.addLink(s4,s5)
    net.addLink(h1,s1)
    net.addLink(h2,s2)
    net.addLink(h3,s3)
    net.addLink(h4,s4)
    net.addLink(h5,s5)

    info('***Iniciando la red\n')
    net.build()
    info('***Iniciando controlador\n')
    OpenDayLight.start()

    info('***Iniciando switches\n')
    net.get('s1').start([OpenDayLight])
    net.get('s2').start([OpenDayLight])
    net.get('s3').start([OpenDayLight])
    net.get('s4').start([OpenDayLight])
    net.get('s5').start([OpenDayLight])

    info('***Esperando a que ODL reconozca la topologia\n')
    info('***Cuando vea la  topologia completa en la GUI de ODL pulse enter\n')
    raw_input()

    info('***Probando conectividad\n\n')
```

```
        net.addLink(h2,s2)
        net.addLink(h3,s3)
        net.addLink(h4,s4)
        net.addLink(h5,s5)

        info('***Iniciando la red\n')
        net.build()
        info('***Iniciando controlador\n')
        OpenDayLight.start()

        info('***Iniciando switches\n')
        net.get('s1').start([OpenDayLight])
        net.get('s2').start([OpenDayLight])
        net.get('s3').start([OpenDayLight])
        net.get('s4').start([OpenDayLight])
        net.get('s5').start([OpenDayLight])

        info('***Esperando a que ODL reconozca la topologia\n')
        info('***Cuando vea la  topologia completa en la GUI de ODL pulse enter\n')
        raw_input()

        info('***Probando conectividad\n\n')
        net.pingAll()
        info('***Cuando vea a todos los hosts en ODL pulse enter')
        raw_input()

        info('***Esperando lanzamiento de PCE y STEM\n')
        info('***Cuando sean lanzados pulse enter\n')
        raw_input()
        info('***Ejecutando test de envio de trafico\n')
        info('***Enviando peticion a STEM para flujo de node1 a node4 mediante Dijkstra\n')
        message = '1 4 1'
        sock.sendto(message.encode(), address)
        info('***Esperando recibir confirmacion de flujo instalado\n')
        data,address = sock.recvfrom(4096)
        data = data.decode()
        while data != 'INSTALLED':
                data,address = sock.recvfrom(4096)
                data  = data.decode()
        info('***Confirmacion recibida\n')
        info('***Enviando trafico\n')
        info('h4 escuchando')
        result_h4 = h4.cmd('cd /home/D-ITG-2.8.1-r1023/bin && ./ITGRecv &')
        print(result_h4)
        time.sleep(2)
```

```
        info('***Cuando vea a todos los hosts en ODL pulse enter')
        raw_input()

        info('***Esperando lanzamiento de PCE y STEM\n')
        info('***Cuando sean lanzados pulse enter\n')
        raw_input()
        info('***Ejecutando test de envio de trafico\n')
        info('***Enviando peticion a STEM para flujo de node1 a node4 mediante Dijkstra\n')
        message = '1 4 1'
        sock.sendto(message.encode(), address)
        info('***Esperando recibir confirmacion de flujo instalado\n')
        data,address = sock.recvfrom(4096)
        data = data.decode()
        while data != 'INSTALLED':
                data,address = sock.recvfrom(4096)
                data  = data.decode()
        info('***Confirmacion recibida\n')
        info('***Enviando trafico\n')
        info('h4 escuchando')
        result_h4 = h4.cmd('cd /home/D-ITG-2.8.1-r1023/bin && ./ITGRecv &')
        print(result_h4)
        time.sleep(2)
        info('h1 generando trafico')
        result_h1 = h1.cmd('cd /home/D-ITG-2.8.1-r1023/bin && ./ITGSend -T UDP -a 10.0.0.4 -c 100 -C 10 -t 180000 -l sender.log -x receiver.log &')
        print(result_h1)
        time.sleep(10)
        info('Tirando link s4-s5')
        net.configLinkStatus('s4','s5','down')'''

        CLI(net)
        net.stop()

if __name__ == '__main__':
        setLogLevel('info')
        myNetwork()
```

ANEXO B. OpenDaylight

La instalación de ODL se ha realizado sobre una máquina virtual Ubuntu Server en su versión 17.10.1. La distribución de ODL utilizada es la Beryllium en la versión 0.4.0.

B.1. Configuración de red

Antes de proceder con la descarga e instalación de OpenDaylight, se debe dotar a la máquina virtual de conectividad. Basta con configurar dos adaptadores de red en la máquina virtual. El primero de ellos debe ser un adaptador de tipo `NAT`, que dotará a la máquina de conexión a internet. El segundo adaptador a configurar es del tipo `Adaptador sólo-anfitrión`, que permiten la conectividad entre la máquina host y las máquinas virtuales asociadas a ese mismo adaptador. Este segundo adaptador posibilitará la conexión entre el controlador ODL y los *switch* OpenFlow de Mininet. Consultar Figura A-1 y Figura A-2.

B.2. Instalación

OpenDaylight es un programa escrito en Java, por lo que antes de nada debe instalarse `Java run-time`:

```
sudo apt−get update
sudo apt−get install default−jre−headless
```

A continuación, debe editarse el archivo “bashrc:

```
sudo nano ˜ /.bashrc
```

Y se añade la siguiente línea:

```
export JAVA HOME=/usr/lib/jvm/default−java
```

Por último, se ejecuta el archivo:

```
source ˜/.bashrc
```

Una vez hecho esto, se ejecuta el siguiente comando para comenzar con la descarga de ODL:

```
wget https://nexus.opendaylight.org/content/groups/
        public/org/opendaylight/integration/
        distribution−karaf/0.4.0−Beryllium/
        distribution−karaf−0.4.0−Beryllium.tar.gz
```

Se extrae el archivo descargado:

```
tar −xvf distribution −karaf −0.4.0−Beryllium.tar.gz
```

Tras la extracción se genera un archivo denominado `distribution-karaf-0.4.0-Beryllium`. Se podrá arrancar el controlador mediante la siguiente acción:

```
./distribution−karaf−0.4.0−Beryllium/bin/karaf
```

Una vez iniciado el entorno, aparece un nuevo terminal en el que introducir comandos propios del controlador. Recién instalado, ODL no cuenta con ningún complemento, por lo que a continuación se explica cómo se han instalado las características necesarias para poder hacer uso de este controlador durante el desarrollo del trabajo. Se instalarán las siguientes funcionalidades:

- `odl-restconf`: Da acceso a la API RESTCONF de ODL.
- `odl-l2switch-switch`: Proporciona funcionalidades de un *switch* de nivel 2.
- `odl-mdsal-apidocs`: Da acceso a la API Yang de ODL
- `odl-dlux-all`: Interfaz gráfica de ODL accesible a través de un navegador web.

Para instalar cualquiera de estos complementos basta con ejecutar el siguiente comando:

```
feature:install <complemento>
```

Si se quiere obtener un listado de todos los posibles complementos a instalar:

```
feature:list
```

Por otro lado, si lo que se pretende es visualizar los complementos ya instalados en el controlador:

```
feature:list --installed
```

Para detener el controlador basta con escribir en línea de comandos:

```
logout
```

B.3. Funcionamiento

Una vez instalado el controlador y las funcionalidades descritas en el apartado anterior, es posible acceder a la interfaz gráfica de este, así como empezar a interactuar con los equipos compatibles con el protocolo OpenFlow.

En primer lugar, se lanza la máquina virtual donde se ha instalado ODL. A continuación, se ejecuta el comando visto anteriormente para iniciar el controlador. Una vez hecho esto se podrá acceder a la GUI de ODL a través de un navegador. Para ello, se escribe la URL con la IP asignada a la interfaz de `Sólo-anfitrión` de la máquina virtual de ODL, así como el puerto por defecto en el que escucha la aplicación (Figura B-1).

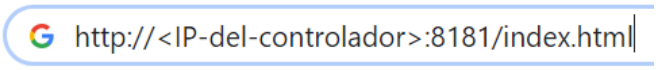

Figura B-1. URL del controlador ODL

Se pedirá la introducción de un usuario y contraseña, `admin` en ambos casos. Una vez pulsado el botón de `login`, se accede a la aplicación (Figura B-2).

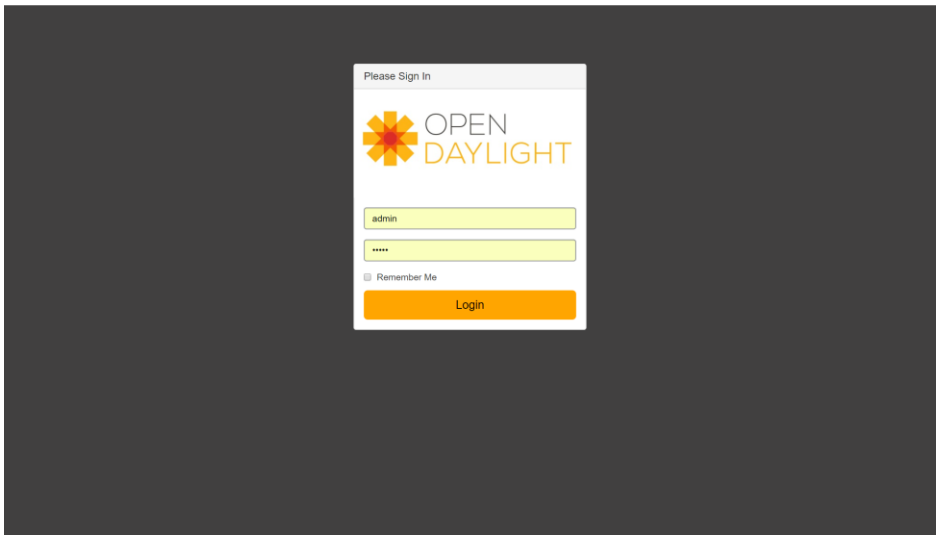

Figura B-2. Pantalla de inicio de sesión de ODL

Esta consta de un menú a la izquierda de la pantalla con distintas opciones a las que acceder y de una zona central, en la que se muestra el contenido de la opción seleccionada en el menú.

Desde la pestaña `Topology` (Figura B-3Figura B-3) se puede ver gráficamente la topología de la red con la que se está trabajando. Es una opción muy útil, sobre todo si se está trabajando con un simulador de red sin interfaz gráfica como Mininet, ya que permite verificar que la topología creada es la que se pretendía. Situando el cursor del ratón sobre los distintos elementos de la red, permite visualizar un resumen de la información de estos, como son direcciones IP y MAC de los hosts, identificadores de los *switch* o puertos por los que estos acceden a un enlace determinado de la red.

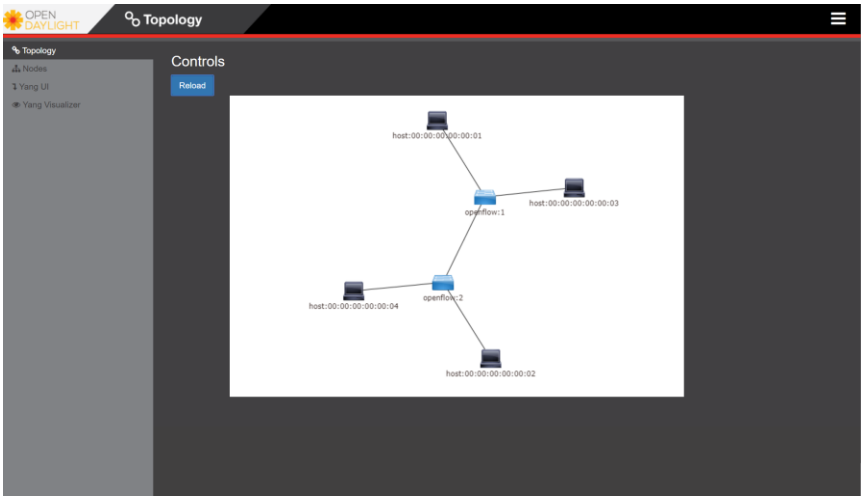

Figura B-3. Pestaña "Topology" del menú de ODL

La pestaña `Nodes` (Figura B-4) aporta información acerca de los *switch* presentes en la red.

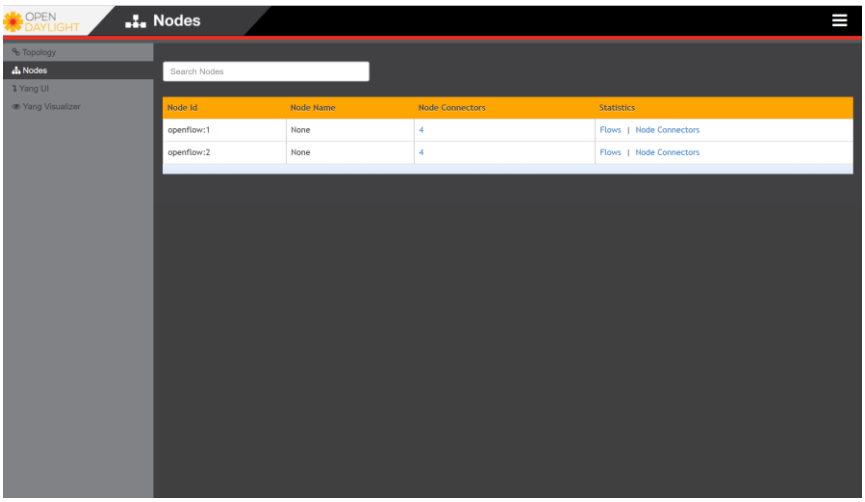

Figura B-4. Pestaña "Nodes" del menú de ODL

Si se pulsa sobre `Node Connectors` (Figura B-5) puede consultarse información relativa a los puertos del *switch* seleccionado.

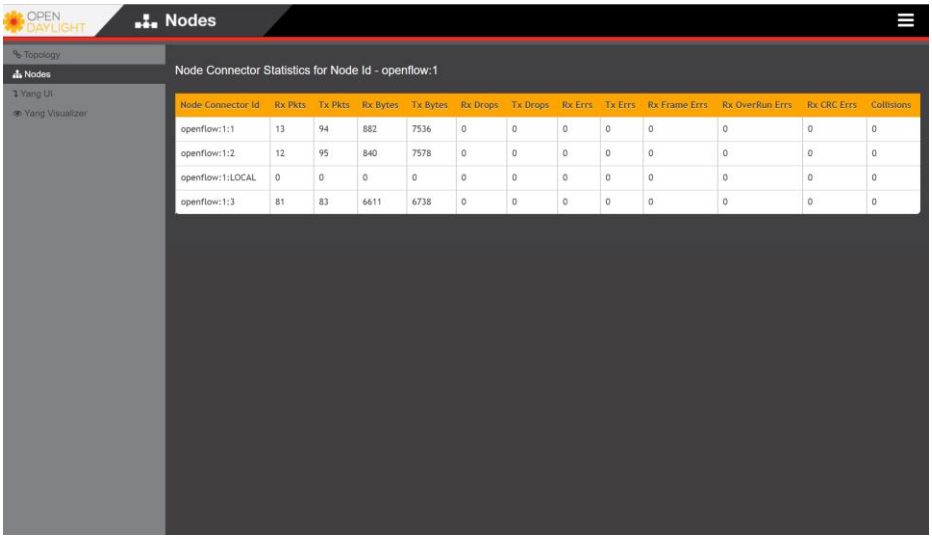

Figura B-5. "Node Connectors" de un switch en ODL

La pestaña `Yang UI` (Figura B-6) es muy importante, ya que es la que permite al usuario interactuar con los *switch* OpenFlow y obtener información de estos, así como instalar y eliminar flujos. Se trata de un cliente REST que, mediante el modelado de estructuras de datos en Yang a través de una interfaz gráfica permite la comunicación con la interfaz REST del controlador OpenDaylight.

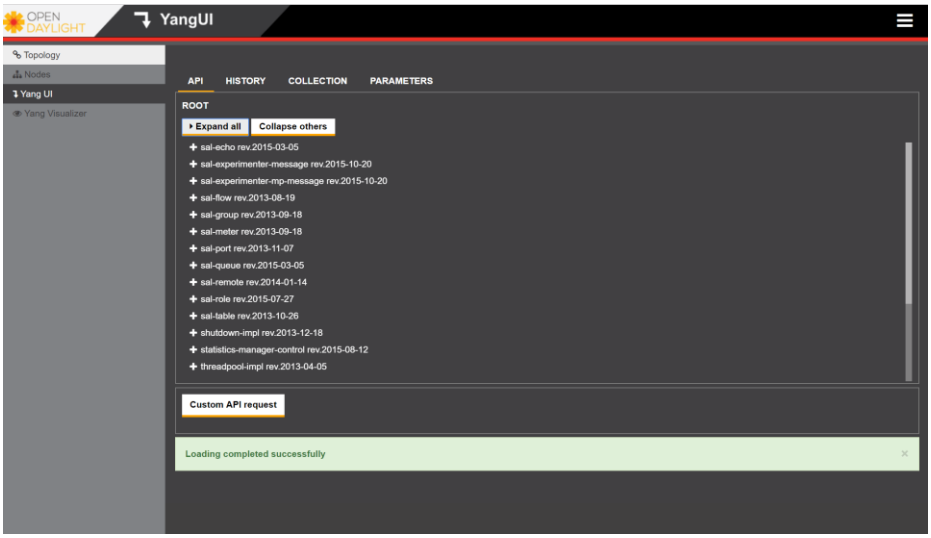

Figura B-6. Pestaña "Yang UI" del menú de ODL

Aparecen en el área de trabajo todas las API disponibles en OpenDaylight, aunque no todas se encuentran operativas, ya que requieren de la instalación de algunas características concretas. En el caso de este proyecto se han utilizado dos API:

- network-topology
- opendaylight-inventory

La API `network-topology` proporciona información referente a la topología, como pueden ser los nodos que la conforman y los enlaces que unen cada uno de estos nodos. Por otro lado, `opendaylight-inventory` hace posible la comunicación con los *switch* OpenFlow a través de ODL. Esta última API cuenta con dos subárboles, `config` y `operational` (Figura B-7).

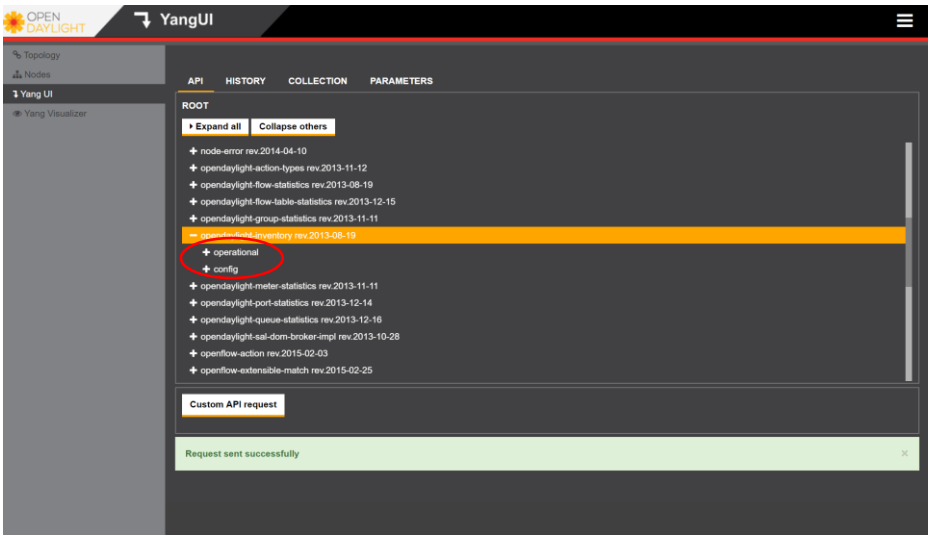

Figura B-7. Subárboles de "opendaylight-inventory"

La base de datos de `config` admite comandos PUT, GET y DELETE, lo que permite al usuario escribir en ella, pedir información sobre su contenido y eliminar entradas. Desplegando `config` se pueden observar todas las opciones disponibles.

Para el caso de instalar un flujo tan sólo debe seleccionarse la acción PUT, rellenar los campos asociados al *switch* destino y completar los campos de identificación del flujo. Además, deberá rellenarse el cuerpo del mensaje, que contendrá la información del flujo a instalar. Notar que los campos de identificación de tabla y de flujo deben ser iguales tanto en los campos presentados en la Figura B-8 como en el cuerpo del mensaje (Figura B-9). Una vez completada toda la información bastará con pulsar el botón `Send`.

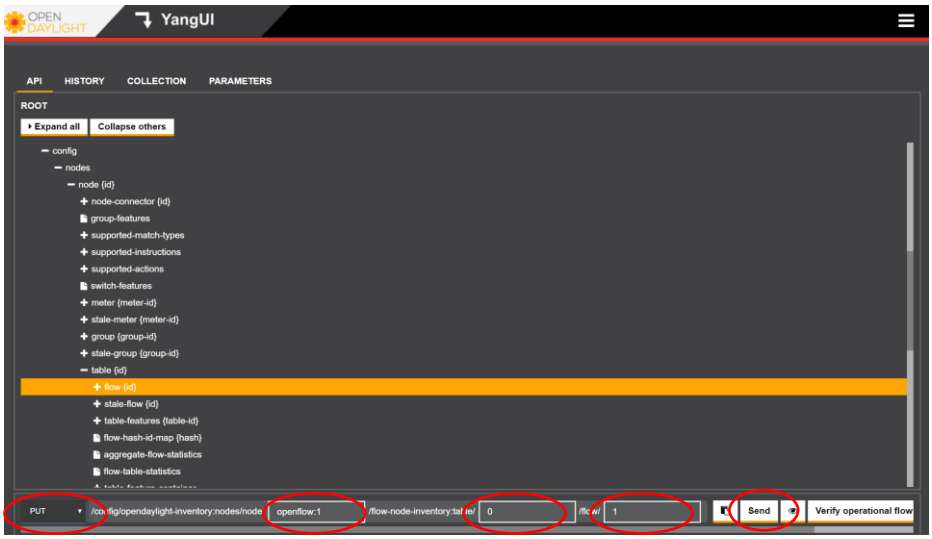

Figura B-8. Haciendo el "PUT" de un flujo

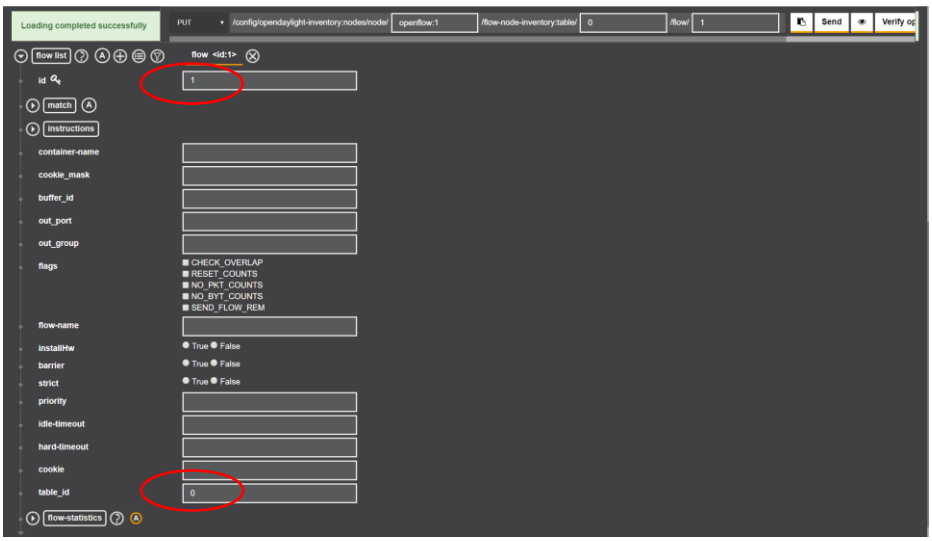

Figura B-9. Cuerpo del mensaje

OpenDaylight copia todos los datos de configuración presentes en `config` y, una vez instalados los flujos en los *switch* pertinentes, traslada toda la información a `operational`. De esta manera, siempre que se quiera verificar si un flujo se ha instalado correctamente en un *switch*, deberá seleccionarse `operational` y rellenar los campos con los datos referentes al flujo que se quiera consultar (Figura B-10).

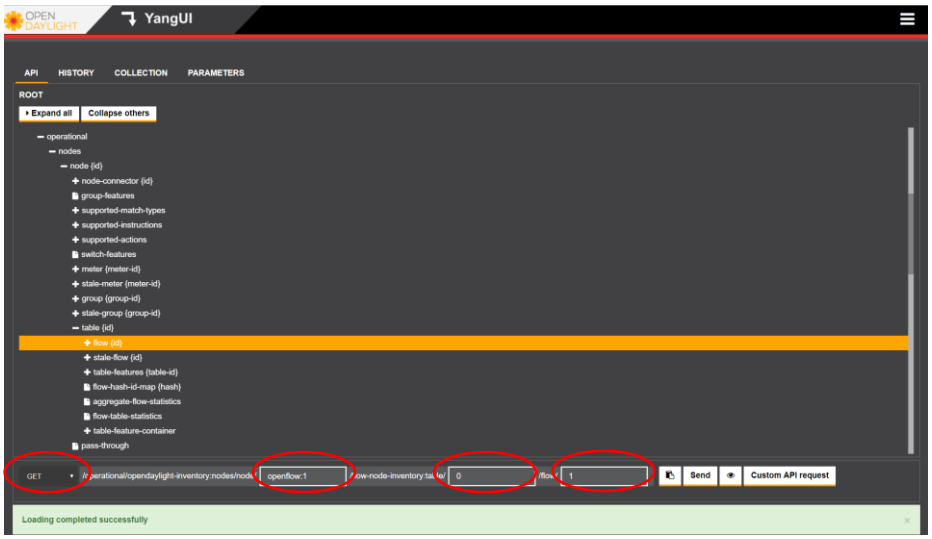

Figura B-10. Consulta de un flujo

La pestaña de `Yang Visualizer` no se ha utilizado en este proyecto, por lo que no se comentará nada al respecto.

ANEXO C. Generador de tráfico D-ITG

En este apéndice se explica el proceso llevado a cabo para la instalación del generador de tráfico D-ITG. Se trata de una herramienta capaz de generar tráfico IPv4 e IPv6, además de permitir medir las principales características de la red, como son el *thoughput*, retardo, *jitter*, pérdida de paquetes, etc.

C.1. Instalación

La instalación de D-ITG va a realizarse sobre la máquina virtual de Mininet. Cabe destacar que una vez lanzada una topología en Mininet, todos los hosts presentes en ella comparten el árbol de directorios con la máquina virtual, por lo que todo software instalado en esta puede ser ejecutado por los hosts de la topología.

En primer lugar, se descarga el programa desde el sitio oficial:

```
sudo wget http://www.grid.unina.it/software/ITG/codice/D-ITG- 2.8.1-r1023-src.zip
```

Una vez descargado, se descomprime el '.zip' obtenido:

```
unzip D-ITG-2.8.1-r1023-src.zip
```

Tras descomprimir el archivo, se generará un directorio con el nombre `D-ITG-2.8.1-r1023`. Entrar en `/D-ITG-2.8.1-r1023/src` y ejecutar:

```
make
```

De esta manera se compilan los binarios del programa y se generarán los ejecutables en el directorio `/D-ITG-2.8.1-r1023/bin`.

Para más información consultar la web oficial[3].

C.2. Generación de tráfico

Una vez obtenidos los binarios de la forma explicada en el aparatado anterior, puede empezar a utilizarse el programa. Va a presentarse un ejemplo sencillo de generación de tráfico de un host a otro. Entrar en ambos hosts en el directorio donde se encuentran los binarios (/D-ITG-2.8.1-r1023/bin). En el host destino de la comunicación debe ejecutarse:

[3] http://www.grid.unina.it/software/ITG/

```
./ITGRecv
```

De esta manera, el host quedará a la escucha de posibles conexiones. En el equipo que genera el tráfico se ejecuta un comando similar al que se expone a continuación:

```
./ITGSend -T UDP -a 127.0.0.1 -c 100 -C 10 -t 15000 -
sender.log -x receiver.log
```

Todo lo que prosigue a `ITGSend` son parámetros configurables de la conexión. En este caso se genera tráfico UDP con una cantidad de datos por paquete constante (100 bytes) y 10 paquetes por segundo durante 15 segundos. Además, se generan dos archivos de *log* en los que se recoge información referente al tráfico enviado, uno para el generador (opción `-l`) y otro para el receptor (opción `-x`).

Se pueden consultar los ficheros de logs mediante:

```
./ITGDec <archivo.log>
```

Para más información sobre los posibles parámetros con los que acompañar estos comandos consultar manual[4].

[4] http://www.grid.unina.it/software/ITG/documentation.php