



**Instituto Politécnico Nacional  
Escuela Superior de Cómputo**

**Práctica 3  
Administración de configuración**

**Integrantes del equipo:**

**Castro Flores Marcela  
Sánchez Cruz Rosa María  
Santiago Mancera Arturo Samuel**

**M. en C. Tanibet Pérez de los Santos Mondragón**

**México, Ciudad de México a 05 de diciembre de 2018**

---

## Índice general

---

<b>1. Introducción</b>	<b>5</b>
<b>2. Sensores implementados</b>	<b>6</b>
2.1. Supervisión de servidor de correo electrónico (SMTP) . . . . .	6
2.1.1. Instalación . . . . .	6
2.1.2. Sensor SMTP . . . . .	10
2.1.2.1. Funcionamiento . . . . .	11
2.2. Supervisión de servidor web (HTTP) . . . . .	12
2.2.1. Instalación . . . . .	12
2.2.2. Funcionamiento . . . . .	14
2.3. Supervisión de servidor de archivos (FTP/FTP Server File Count) . . . . .	16
2.3.1. Sensor FTP . . . . .	16
2.3.2. Instalación . . . . .	16
2.3.3. Funcionamiento . . . . .	18
2.3.4. Sensor FTP Server File Count . . . . .	22
2.3.5. Instalación . . . . .	22
2.3.6. Funcionamiento . . . . .	22
2.4. Supervisión de servidor de impresión (SNMP) . . . . .	23
2.4.0.1. Configuracion . . . . .	23
2.4.0.2. Monitorizacion de impresoras . . . . .	30
2.4.0.3. Funcionamiento . . . . .	36
2.5. Supervisión de servidor de servidor de acceso remoto (SSH) . . . . .	37
2.5.1. Instalación . . . . .	37
2.5.2. Sensor SSH . . . . .	38
2.5.2.1. Funcionamiento . . . . .	38
2.6. Administración de archivos de configuración . . . . .	40
2.6.0.1. Importación . . . . .	40
2.6.0.2. Exportación . . . . .	41
<b>3. Configuración de routers</b>	<b>43</b>
<b>4. Conclusiones</b>	<b>47</b>

---

## Índice de figuras

---

2.1.	Archivo de configuración /etc/host.conf.	6
2.2.	Archivo de resolución de nombres de dominio.	7
2.3.	Configuración final de Postfix.	7
2.4.	Login de configuración de Webmail.	8
2.5.	Configuración de dominio.	9
2.6.	Bandeja de entrada.	10
2.7.	Código en Python del sensor de correo.	11
2.8.	Funcionamiento del sensor de correo.	12
2.9.	Comando de instalación HTTP apache2.	13
2.10.	Verificación de status servidor HTTP.	13
2.11.	Visualización de ip.	14
2.12.	Página de Apache2 en el navegador.	14
2.13.	Sensor de monitoreo HTTP.	15
2.14.	Código de sensor de monitoreo HTTP.	15
2.15.	Comando de instalación FTP.	16
2.16.	Archivo de configuración FTP.	17
2.17.	Reinicio de servicio y status FTP.	18
2.18.	Opciones de sensor FTP.	19
2.19.	Archivos disponibles en servidor FTP.	19
2.20.	Solicitud de descarga de archivo vía FTP.	20
2.21.	Información del archivo enviado al servidor FTP.	20
2.22.	Archivos dentro del servidor FTP.	20
2.23.	Contenido del archivo enviado al servidor FTP.	21
2.24.	Código para descargar un archivo del servidor FTP.	21
2.25.	Código para subir un archivo del servidor FTP.	22
2.26.	Código del contador de archivos en el servidor FTP.	22
2.27.	Contador de archivos en el servidor FTP.	23
2.28.	comandos.	23
2.29.	comandos.	24
2.30.	comandos.	25
2.31.	comandos.	25
2.32.	configuración del archivo cupsd.conf	26
2.33.	configuracion de la ip de la pc location	27
2.34.	configuracion de la ip de la pc admin	28
2.35.	configuracion de la ip de la pc admin/conf	29

---

2.36. reinicio y apago de firewall . . . . .	29
2.37. servicio activo . . . . .	30
2.38. code . . . . .	31
2.39. code . . . . .	31
2.40. code . . . . .	32
2.41. code . . . . .	33
2.42. code . . . . .	34
2.43. code . . . . .	35
2.44. menú . . . . .	36
2.45. sin servidor . . . . .	36
2.46. sin impresora . . . . .	37
2.47. Impresoras conectadas . . . . .	37
2.48. Código del sensor SSH. . . . .	38
2.49. Funcionamiento del sensor SSH. . . . .	39
2.50. Conexiones SSH. . . . .	39
2.51. Archivo de configuracion para importaciones. . . . .	40
2.52. Código para importar archivos. . . . .	40
2.53. Funcionamiento de la importación de archivos. . . . .	41
2.54. Archivo de configuracion para exportaciones. . . . .	41
2.55. Código para exportar archivos. . . . .	41
2.56. Funcionamiento de la exportación de archivos. . . . .	42
3.1. Computadora 1. . . . .	43
3.2. Computadora 2. . . . .	44
3.3. Computadora 3. . . . .	44
3.4. Configuración enrutamiento estático. . . . .	45
3.5. Configuración RIP. . . . .	45
3.6. Configuración OSPF. . . . .	46

# CAPÍTULO 1

---

## Introducción

---

Para la realización de la última práctica se implementaron diferentes servidores con los cuales por medio de la ejecución de diferentes sensores se obtenían tanto sus diferentes tiempos de ejecución como las diferentes respuestas que el servidor desplegaba según era el caso.

Los servidores implementados son los siguientes:

- Servidor de correo electrónico (SMTP)
- Servidor web (HTTP)
- Servidor de archivos (FTP/FTP Server File Count)
- Servidor de impresión (SNMP)
- Servidor de acceso remoto (SSH)

Esta práctica se dividió en las tres partes siguientes:

1. En la primera parte se requirió realizar la instalación y la configuración de los distintos servidores mencionados anteriormente mismos que fueron separados en diferentes máquinas virtuales con el fin de que en el paso posterior se pudiera dividir la topología dada.
2. La segunda parte fue la configuración de la topología que iba desde realizar la conexión correcta entre los dispositivos como también realizar las diferentes comprobaciones para verificar que si había una conexión entre todos los dispositivos.
3. Por último, la tercera parte correspondió a la utilización de los diferentes sensores que verificaban la correcta respuesta y funcionamiento de cada servidor y de igual manera, se utilizó el administrador de archivos de configuración con el cual se tenía tanto la posibilidad de descargar como de subir los archivos correspondientes a cada uno de los routers disponibles.

En el capítulo mostrado a continuación se observa el desarrollo de la práctica indicando tanto el código utilizado para la implementación de los sensores de cada servidor, como las pantallas que muestran paso a paso el proceso que se realizó para la configuración de la topología dada.

# CAPÍTULO 2

---

## Sensores implementados

---

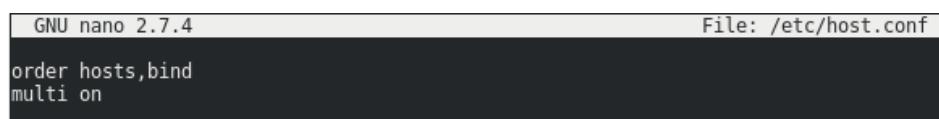
### 2.1. Supervisión de servidor de correo electrónico (SMTP)

#### 2.1.1. Instalación

Para llevar a cabo la instalación de un servidor de correo se deben tomar en cuenta 3 elementos: el servidor SMTP (*Simple Mail Transfer Protocol*) encargado de encaminar los correos electrónicos, el servidor IMAP o POP3 encargado de administrar los correos electrónicos recibidos y un cliente de correo encargado de administrar el correo electrónico de las cuentas de usuario.

Para esta instalación se utilizó Postfix como servidor SMTP, Dovecot como servidor IMAP y Rainloop Webmail como cliente de correo electrónico a través de HTTP.

El primer paso es la preparación de los servicios de DNS del servidor. Sin embargo, para esta práctica no se implementó un servidor DNS. Por lo tanto, la resolución de nombres de dominio se realizó localmente en el servidor modificando el archivo `/etc/host.conf` como se muestra en a figura 2.1.



```
GNU nano 2.7.4                                         File: /etc/host.conf
order hosts,bind
multi on
```

Figura 2.1: Archivo de configuración `/etc/host.conf`.

Así mismo, se cambió el hostname del servidor a **server1** en el archivo `/etc/hostname` se agregó una resolución de nombre de dominio al archivo `/etc/hosts` para indicar que el servidor local es el host y dominio `server1.redes3.local`:

```
root@server1:/home/samuel# cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      ss

# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.100.70.56 redes3.local server1.redes3.local
```

Figura 2.2: Archivo de resolución de nombres de dominio.

Este último paso se debe realizar en cada cliente de la topología que utilice los servicios de correo con la finalidad de que pueda reconocer la dirección IP del dominio.

Una vez configurado la resolución de nombre de dominio y el nombre de host se procede a instalar **Postfix**. Para ello se utiliza el comando:

```
apt-get install postfix
```

Durante la instalación se solicitará ingresar el tipo de configuración general de correo (*General type of mail configuration*). Para lo cual se eligió **Internet Site**. De igual forma se solicitará el nombre de sistema de correo en el cual se establece el nombre de host más el nombre de dominio configurado anteriormente: **server1.redes3.local**.

El siguiente paso es editar la configuración de Postfix desde su archivo **/etc/postfix/main.cf**. Esta configuración se puede ver reflejada con el comando:

```
postconf -n
```

La configuración final se puede observar en la figura 2.3

```
root@server1:/home/samuel# postconf -n
alias_database = hash:/etc/aliases
alias_maps = hash:/etc/aliases
append_dot_mydomain = no
biff = no
compatibility_level = 2
home_mailbox = Maildir/
inet_interfaces = all
inet_protocols = all
mailbox_size_limit = 0
mydestination = $myhostname, $mydomain, localhost.$mydomain, , localhost
mydomain = redes3.local
myhostname = server1.redes3.local
mynetworks = 127.0.0.0/8 10.0.2.0/24 [::ffff:127.0.0.0]/104 [::1]/128
myorigin = $mydomain
readme_directory = no
recipient_delimiter = +
relayhost =
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
smtpd_banner = $myhostname ESMTP
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
smtpd_tls_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file = /etc/ssl/private/ssl-cert-snakeoil.key
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtpd_use_tls = yes
root@server1:/home/samuel#
```

Figura 2.3: Configuración final de Postfix.

El paso siguiente es instalar Dovecot como servidor IMAP. Para ello se utiliza el siguiente comando:

```
apt install dovecot-core dovecot-imapd
```

Una vez instalado deberemos editar algunos de los archivos de configuración. El primero es el archivo **/etc/dovecot/dovecot.conf** en el cual se debe descomentar la línea:

```
listen = *, ::
```

El segundo archivo a editar es **/etc/dovecot/conf.d/10-auth.conf** en el cual se deben tener líneas como las siguientes:

```
disable_plaintext_auth = no
```

```
auth_mechanisms = plain login
```

El tercer archivo a editar es `/etc/dovecot/conf.d/10-mail.conf` en la siguiente línea:

```
mail_location = maildir: /Maildir
```

Y el último archivo a editar es `/etc/dovecot/conf.d/10-master.conf` en el bloque de smpt-auth:

```
# Postfix smtp-auth
unix_listener /var/spool/postfix/private/auth {
mode = 0666
user = postfix
group = postfix
}
```

Reiniciamos Postfix y Dovecot:

```
service postfix restart service dovecot restart
```

El último paso es la instalación de Rainloop Webmail como cliente de correo electrónico a través de HTTP. Para ello se instala un servidor apache y las dependencias necesarias para utilizar PHP. Para ello, usamos el siguiente comando:

```
apt install apache2 php7.0 libapache2-mod-php7.0 php7.0-curl php7.0-xml
```

Eliminamos el archivo índice del Apache y descargamos Webmail:

```
# cd /var/www/html/
```

```
# rm index.html
```

```
# curl -sL https://repository.rainloop.net/installer.php | php
```

Finalizada la instalación procedemos a ingresar al navegador web en la dirección `http://localhost/?admin` e ingresamos con las credenciales: **User: admin** y **Password: 12345** como se muestra en la figura 2.4.

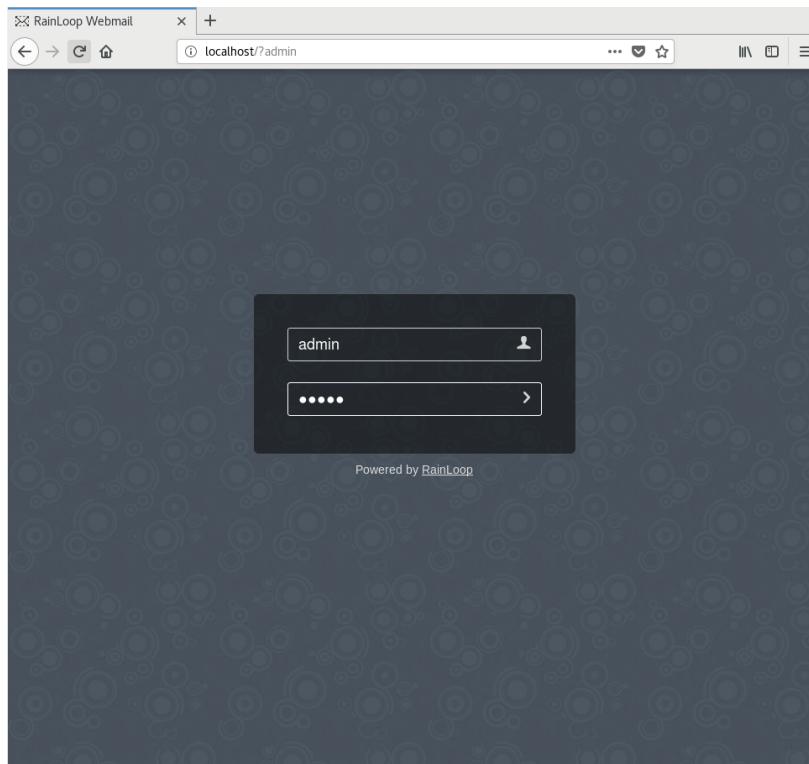


Figura 2.4: Login de configuración de Webmail.

Dentro del panel de configuración nos dirigimos a la pestaña **Domains** en donde se agrega el dominio configurado al inicio. Esto se puede ver en la figura 2.5.

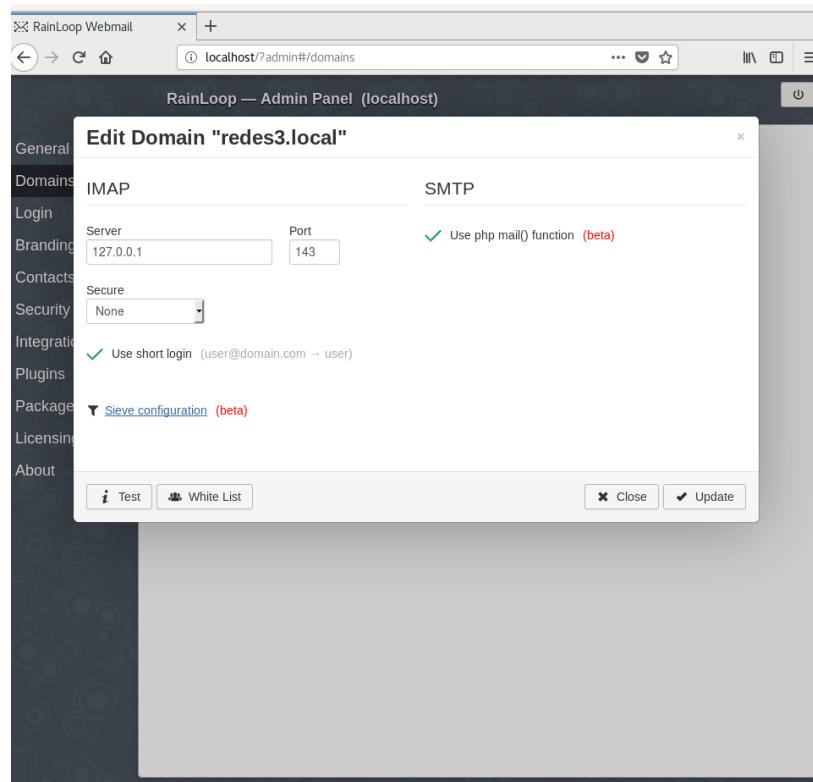


Figura 2.5: Configuración de dominio.

Finalmente, para crear un nuevo usuario en el servidor de correo (SNMP + IMAP) es necesario crear al usuario UNIX. Sin embargo, antes se debe definir una variable global que indique el directorio de almacenamiento de los correos para cada usuario. Para realizar esto se utilizan los siguientes comandos:

```
echo 'export MAIL=$HOME/Maildir' >>/etc/profile
useradd -m samuel
passwd samuel
```

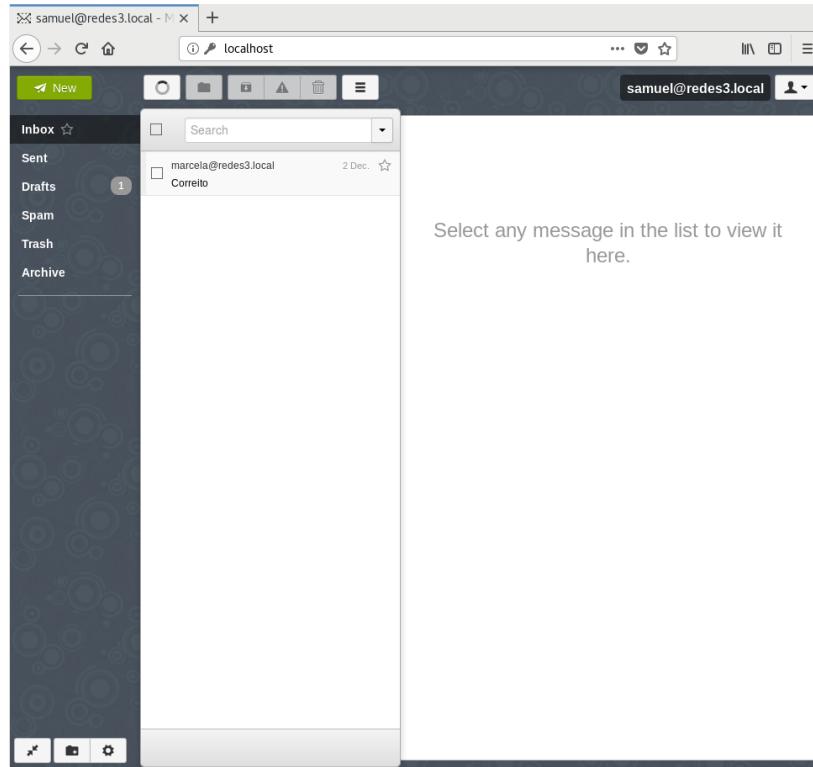


Figura 2.6: Bandeja de entrada.

### 2.1.2. Sensor SMTP

El sensor SMTP realiza mediciones de tiempo de respuesta tanto del servidor SMTP como del servidor IMAP. Para ello, se toma el tiempo (EPOCH) antes de realizar una solicitud SMTP, es decir, tiempo inicial. Después, se envía un correo de prueba a un usuario predefinido. Una vez que se terminó de ejecutar la función de solicitud de envío se vuelve a tomar el tiempo. Este tiempo será el tiempo de respuesta del servidor SMTP, puesto que el servidor SMTP ya terminó su tarea de enviar el correo al servidor IMAP. No obstante, para obtener el tiempo de respuesta del servidor IMAP se obtiene el nombre del último archivo modificado en la bandeja de entrada del usuario de prueba. Por defecto, Dovecot almacena los correos en el sistema de archivos concatenando en el nombre el tiempo (EPOCH) en el que fue recibido. Es por ello, que este valor se considera el tiempo de respuesta del servidor IMAP.

El proceso se muestra a detalle en la función `sensor_correo()` de la figura 2.7. Dicha función es parte del programa `cliente.py` escrito en Python que contiene la funcionalidad de todos los sensores solicitados en esta práctica.



```

106 def sensor_correo():
107     """
108     Antes de ejecutar la función se debe establecer una conexión
109     inicial con el servidor ssh para intercambiar firmas.
110     """
111     ssh_hostname = '192.168.0.32' # IP del servidor
112     ssh_port = 22
113     ssh_username = 'root'
114     ssh_password = 'holal23..'
115
116     sender = 'root'
117     sender_pass = 'holal23..'
118     receiver = 'marcela@redes3.local'
119     server = 'server1.redes3.local' # Cambiar la IP en /etc/hosts
120     email_port = 25
121
122     correo = MIMEText()
123     correo['From'] = sender
124     correo['To'] = receiver
125     correo['Subject'] = "sensor-SMTP-redes3.local"
126     correo.attach(MIMEText('Correo de prueba - Hacer caso omiso.'))
127
128     mailServer = smtplib.SMTP(server,25)
129     mailServer.ehlo()
130     mailServer.starttls()
131     mailServer.ehlo()
132     #mailServer.login(sender, sender_pass)
133     tiempo_inicio = int(round(time.time() * 1000)) # Milisegundos
134     #print 'Tiempo inicial: '+str(tiempo_inicio)
135     mailServer.sendmail(sender,receiver,correo.as_string()) # No so por que se puede hacer esto sin autent
136     tiempo_fin_smtp = int(round(time.time() * 1000))
137     tiempo_smtp = abs(tiempo_fin_smtp - tiempo_inicio)
138     print 'Tiempo de respuesta SMTP: '+str(tiempo_smtp) + 'ms'
139     mailServer.close()
140
141     paramiko.util.log_to_file('paramiko.log')
142     s = paramiko.SSHClient()
143     s.load_system_host_keys()
144     s.connect(ssh_hostname,ssh_port,ssh_username,ssh_password)
145     stdin, stdout, stderr = s.exec_command("find /home/marcela/Maildir/new/ -printf '%C@\n' | tail -1")
146     tiempo_imap = abs(int(round(float(stdout.read()) * 1000)) - tiempo_inicio)
147     print 'Tiempo de respuesta IMAP: ' + str(tiempo_imap) + 'ms'
148     print 'Tiempo total: '+str(tiempo_smtp + tiempo_imap) + 'ms'
149
150     stdin, stdout, stderr = s.exec_command("find /home/marcela/Maildir/new/ | tail -1")
151     archivo = stdout.read()
152     stdin, stdout, stderr = s.exec_command("rm "+archivo)
153     s.close()

```

Figura 2.7: Código en Python del sensor de correo.

De la línea 122 a la línea 131 se realiza la preparación del correo de prueba. En la línea 133 se toma el tiempo inicial y posteriormente, en la línea 135 se realiza la solicitud de envío SMTP. Una vez que termina, en la línea 136 se vuelve a tomar el tiempo que indica la respuesta SMTP. A partir de la línea 141 a la línea 153 se realiza una conexión mediante SSH para obtener el tiempo EPOCH del nombre del último archivo modificado en la bandeja de entrada del usuario de prueba. Finalmente, utilizando SSH se elimina el correo de prueba.

### 2.1.2.1. Funcionamiento

Inciamos corriendo el programa **cliente.py** de la siguiente manera:

**python cliente.py**

Se despliega el menú de opciones y elegimos la opción 1 que corresponde al sensor SMTP. Con ello, el sensor realizará todas las operaciones antes mencionadas y mostrará en pantalla los tiempos obtenidos como se puede observar en la figura 2.8

```
root@ss:/home/samuel/redes3/Redes3/TercerParcial# python cliente.py

----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):1

----- SENSOR SMTP -----
Tiempo de respuesta SMTP: 498ms
Tiempo de respuesta IMAP: 328706ms
Tiempo total: 329204ms
```

Figura 2.8: Funcionamiento del sensor de correo.

## 2.2. Supervisión de servidor web (HTTP)

### 2.2.1. Instalación

Para la supervisión del funcionamiento de este servidor, se realizó primero la instalación que es muy sencilla en una máquina virtual Ubuntu Server, en la cual se ingresó el comando **sudo apt-get install apache2**, mostrado en la figura 2.9 mismo que instalaba las librerías necesarias de este servidor.

```
x - □ march@march-Lenovo-Y50-70:~  
march@march-Lenovo-Y50-70:~$ sudo apt-get install apache2  
[sudo] password for march:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1  
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0  
Suggested packages:  
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom  
The following NEW packages will be installed:  
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1  
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0  
0 upgraded, 9 newly installed, 0 to remove and 40 not upgraded.  
Need to get 1 540 kB of archives.  
After this operation, 6 373 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 libapr1 amd64 1.5.2-  
3 [86,0 kB]  
Get:2 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1 amd64 1.  
5.4-1build1 [77,1 kB]  
Get:3 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1-dbd-sqlit  
e3 amd64 1.5.4-1build1 [10,6 kB]  
Get:4 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 libaprutil1-ldap amd  
64 1.5.4-1build1 [8 720 B]  
Get:5 http://fr.archive.ubuntu.com/ubuntu xenial/main amd64 liblua5.1-0 amd64 5.  
1.5-8ubuntu1 [102 kB]  
Get:6 http://fr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-bin  
amd64 2.4.18-2ubuntu3.9 [925 kB]  
Get:7 http://fr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-util  
s amd64 2.4.18-2ubuntu3.9 [81,8 kB]  
Get:8 http://fr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-data  
all 2.4.18-2ubuntu3.9 [162 kB]  
Get:9 http://fr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2 amd6
```

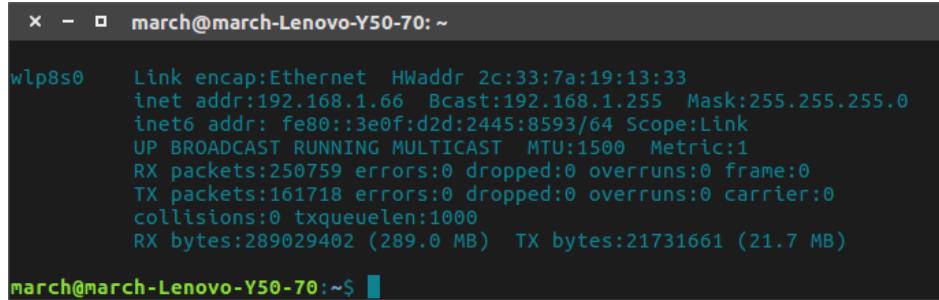
Figura 2.9: Comando de instalación HTTP apache2.

Una vez que se realizó la instalación de los paquetes correspondientes a dicho servidor, se ejecutó el comando **sudo systemctl status apache2** con el cual se verificaba que el servidor estuviera activo como se observa en la figura 2.10.

```
x - □ march@march-Lenovo-Y50-70:~  
march@march-Lenovo-Y50-70:~$ sudo systemctl status apache2  
● apache2.service - LSB: Apache2 web server  
  Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)  
  Drop-In: /lib/systemd/system/apache2.service.d  
            └─apache2-systemd.conf  
    Active: active (running) since lun. 2018-12-03 20:18:59 CST; 42s ago  
      Docs: man:systemd-sysv-generator(8)  
    CGroup: /system.slice/apache2.service  
           ├─4987 /usr/sbin/apache2 -k start  
           ├─4990 /usr/sbin/apache2 -k start  
           ├─4991 /usr/sbin/apache2 -k start  
  
déc. 03 20:18:58 march-Lenovo-Y50-70 systemd[1]: Starting LSB: Apache2 web serve  
déc. 03 20:18:58 march-Lenovo-Y50-70 apache2[4965]: * Starting Apache httpd web  
déc. 03 20:18:58 march-Lenovo-Y50-70 apache2[4965]: AH00558: apache2: Could not  
déc. 03 20:18:59 march-Lenovo-Y50-70 apache2[4965]: *  
déc. 03 20:18:59 march-Lenovo-Y50-70 systemd[1]: Started LSB: Apache2 web server
```

Figura 2.10: Verificación de status servidor HTTP.

Y una vez que se verifica que su status es activo, se visualiza en terminal la IP de la máquina con el fin de ingresarla en el navegador mediante el cual se obtiene la página de inicio del servidor de Apache (figuras 2.11 y 2.12).



```
wlp8s0      Link encap:Ethernet  HWaddr 2c:33:7a:19:13:33
            inet  addr: 192.168.1.66  Bcast: 192.168.1.255  Mask: 255.255.255.0
            inet6 addr: fe80::3e0f:d2d:2445:8593/64 Scope:Link
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:250759 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:161718 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:289029402 (289.0 MB)  TX bytes:21731661 (21.7 MB)

march@march-Lenovo-Y50-70:~$
```

Figura 2.11: Visualización de ip.

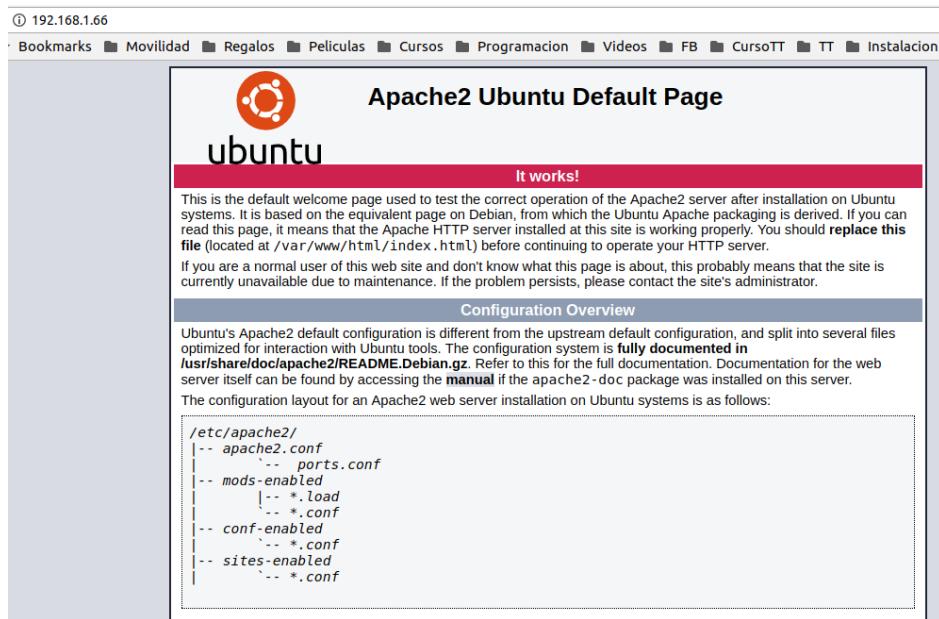


Figura 2.12: Página de Apache2 en el navegador.

## 2.2.2. Funcionamiento

Ya que se ha realizado correctamente toda la instalación del servidor, se prueba este mediante el sensor HTTP. Al ejecutar el programa de python, nuevamente nos aparece el menú de opciones y se selecciona la opción 2 la cual nos despliega la información perteneciente a dicho servidor como lo es el tiempo de respuesta, los bytes recibidos y la velocidad del ancho de banda (figura 2.13).

```
----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):2

----- SENSOR HTTP -----
Ingresa host del servidor: '192.168.1.69'

Inicio de solicitud: 2018-12-04 21:46:13.406109
Recepcion de solicitud: 2018-12-04 21:46:13.406220
Tiempo de respuesta de solicitud: 0:00:00.000111 segundos
Bytes recibidos: 11321 bytes
Velocidad de ancho de banda 133.0 Kb/s
```

Figura 2.13: Sensor de monitoreo HTTP.

La imagen de la figura 2.14, muestra el código del funcionamiento de esta sección en el cual se solicita primero el host del servidor HTTP y posteriormente se realiza una petición en este caso GET, misma que puede cambiar por POST o cualquier otra que se desee y con base en esto, se obtienen el tiempo de respuesta de la solicitud, los bytes transmitidos y la velocidad del ancho de banda.

```
def HTTPmethod():
    ip = str(input('Ingresa host del servidor: '))
    print
    #ip = '192.168.1.69'
    #nf = urllib.urlopen('http://'+ip).read()
    #Tiempo de respuesta
    conn = httplib.HTTPConnection(ip)
    start = datetime.datetime.now()
    print 'Inicio de solicitud: ', start
    end = datetime.datetime.now()
    print 'Recepcion de solicitud: ', end
    print 'Tiempo de respuesta de solicitud: ', end - start, 'segundos'
    #Bytes recibidos
    conn.request('GET', '/')
    res = conn.getresponse()
    data = res.read()
    print 'Bytes recibidos: ', (len(data)), 'bytes'
    #Ancho de banda
    start = time.time()
    file = requests.get('http://'+ip)
    end = time.time()
    time_difference = end - start
    file_size = int(file.headers['Content-Length'])/1000
    print 'Velocidad de ancho de banda', round(file_size / time_difference), 'Kb/s'
```

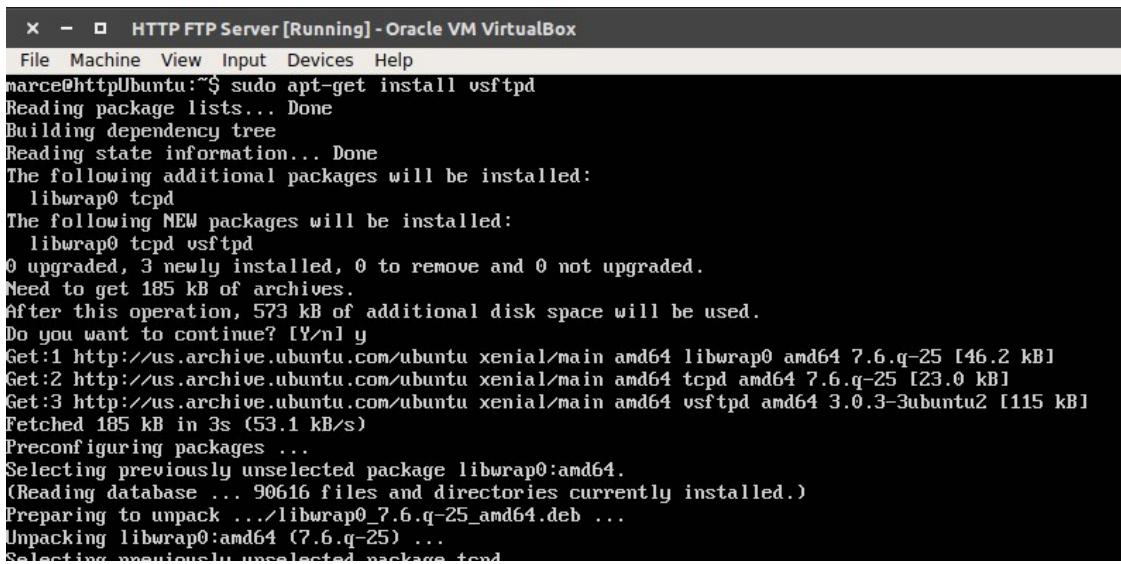
Figura 2.14: Código de sensor de monitoreo HTTP.

## 2.3. Supervisión de servidor de archivos (FTP/FTP Server File Count)

### 2.3.1. Sensor FTP

### 2.3.2. Instalación

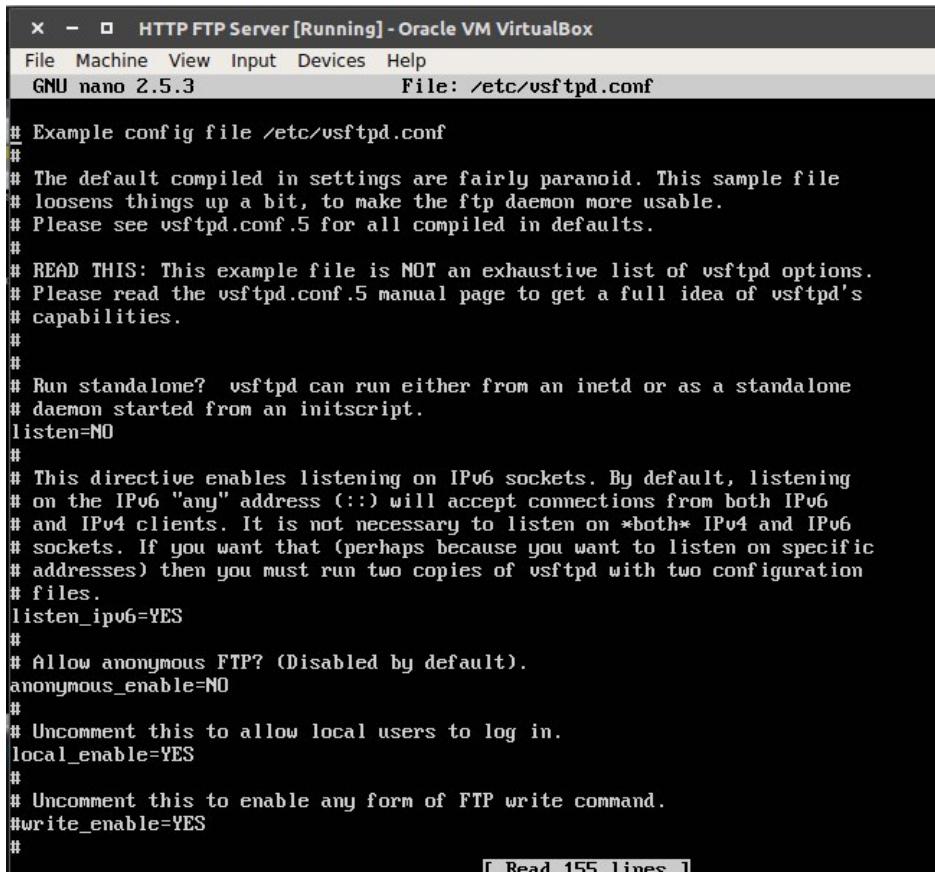
Para la utilización de este sensor, se realizó primero la instalación del servidor FTP en una máquina virtual Ubuntu Server, en la cual se ingresó el comando **sudo apt-get install vsftpd**, mostrado en la figura 2.15 mismo que instalaba las librerías necesarias de este servidor.



```
X - □ HTTP FTP Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
narce@httpUbuntu:~$ sudo apt-get install vsftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libwrap0 tcpd
The following NEW packages will be installed:
  libwrap0 tcpd vsftpd
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 185 kB of archives.
After this operation, 573 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 libwrap0 amd64 7.6.q-25 [46.2 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 tcpd amd64 7.6.q-25 [23.0 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 vsftpd amd64 3.0.3-3ubuntu2 [115 kB]
Fetched 185 kB in 3s (53.1 kB/s)
Preconfiguring packages ...
Selecting previously unselected package libwrap0:amd64.
(Reading database ... 90616 files and directories currently installed.)
Preparing to unpack .../libwrap0_7.6.q-25_amd64.deb ...
Unpacking libwrap0:amd64 (7.6.q-25) ...
Selecting previously unselected package vsftpd.
```

Figura 2.15: Comando de instalación FTP.

Posteriormente, ya que se había realizado toda la descarga de paquetes, se ingreso mediante el comando **sudo nano /etc/vsftpd.conf**, como lo muestra la figura 2.16 con el fin de eliminar el comentario de la línea **write\_enable = YES** y de esta manera permitir la escritura de archivos dentro del servidor.



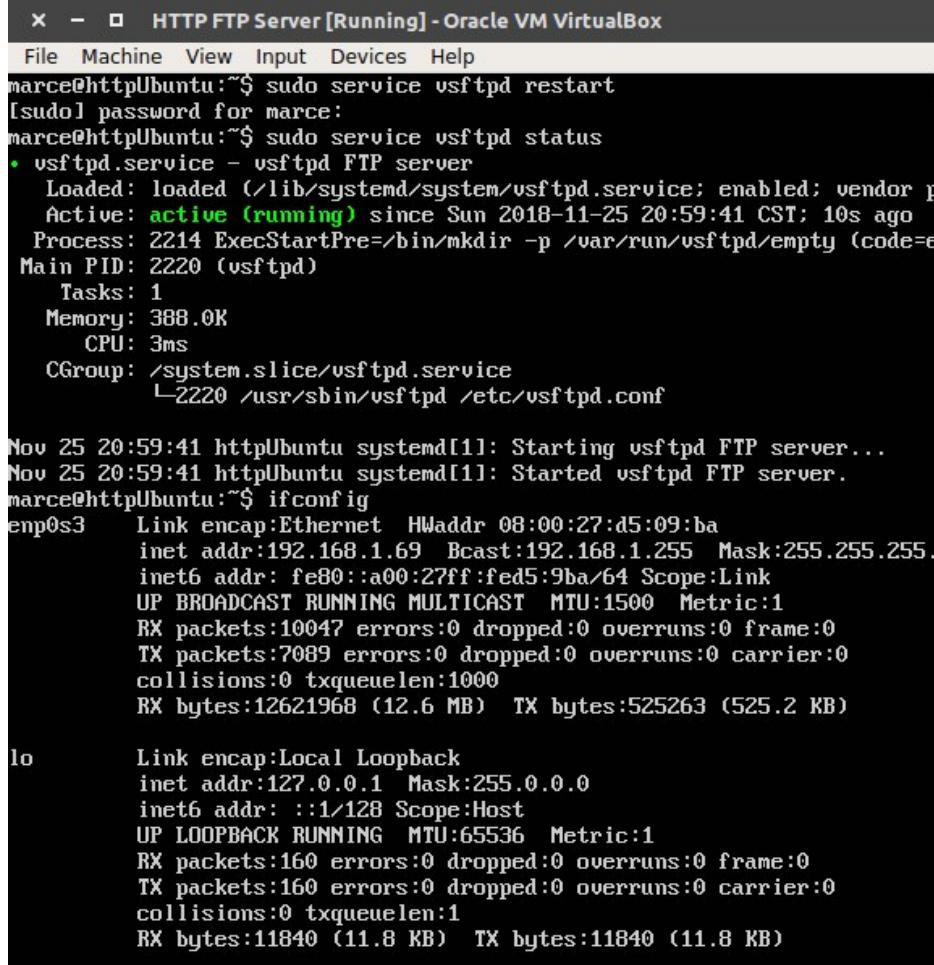
The screenshot shows a terminal window titled "HTTP FTP Server [Running] - Oracle VM VirtualBox". The window contains the contents of the /etc/vsftpd.conf file. The file is a configuration for the vsftpd daemon, containing various directives like listen=NO, listen\_ipv6=YES, anonymous\_enable=NO, and local\_enable=YES. The terminal window has a dark background with white text, and the nano editor interface is visible at the top.

```
# Example config file /etc/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
#
# Run standalone?  vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
listen=NO
#
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
#

```

Figura 2.16: Archivo de configuración FTP.

Por último, se ejecutó el comando **sudo service vsftpd restart** con el cual se reinicia el servicio FTP y posteriormente el comando **sudo service vsftpd status** para asegurarnos de que el estado del servicio sea activo. Por último únicamente se obtuvo la ip de la máquina virtual para saber el host al cual se enviarían y se recibirían los datos almacenados en dicho servidor (figura 2.17).



```

X - □ HTTP FTP Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
marce@httpUbuntu:~$ sudo service vsftpd restart
[sudo] password for marce:
marce@httpUbuntu:~$ sudo service vsftpd status
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor pr
   Active: active (running) since Sun 2018-11-25 20:59:41 CST; 10s ago
     Process: 2214 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=e
   Main PID: 2220 (vsftpd)
      Tasks: 1
     Memory: 388.0K
        CPU: 3ms
       CGroup: /system.slice/vsftpd.service
               └─2220 /usr/sbin/vsftpd /etc/vsftpd.conf

Nov 25 20:59:41 httpUbuntu systemd[1]: Starting vsftpd FTP server...
Nov 25 20:59:41 httpUbuntu systemd[1]: Started vsftpd FTP server.
marce@httpUbuntu:~$ ifconfig
enp0s3    Link encap:Ethernet HWaddr 08:00:27:d5:09:ba
          inet addr:192.168.1.69 Bcast:192.168.1.255 Mask:255.255.255.
          inet6 addr: fe80::a00:27ff:fed5:9ba/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:10047 errors:0 dropped:0 overruns:0 frame:0
            TX packets:7089 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:12621968 (12.6 MB) TX bytes:525263 (525.2 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:65536 Metric:1
            RX packets:160 errors:0 dropped:0 overruns:0 frame:0
            TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:11840 (11.8 KB) TX bytes:11840 (11.8 KB)

```

Figura 2.17: Reinicio de servicio y status FTP.

### 2.3.3. Funcionamiento

En cuanto al funcionamiento de este sensor, se tenían dos diferentes opciones que se desplegaban en el menú al ejecutar el programa cliente.py, dichas opciones eran tanto importar como exportar un archivo al servidor FTP (figura 2.18).

```
----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):3

----- SENSOR FTP -----
1. Descargar archivo desde servidor
2. Subir archivo a servidor
3. Salir
Selecciona una opcion (numero entero 1-3):1
```

Figura 2.18: Opciones de sensor FTP.

Ya que se seleccionó en este caso la opción 1, se solicitan ciertos datos para poder acceder al servidor y obtener la información de este (figura 2.19).

```
----- SENSOR FTP -----
1. Descargar archivo desde servidor
2. Subir archivo a servidor
3. Salir
Selecciona una opcion (numero entero 1-3):1
Ingresa host del servidor: '192.168.1.69'
Ingresa el usuario: 'marce'
Ingresa la contraseña: '1596'
```

Figura 2.19: Archivos disponibles en servidor FTP.

Una vez que se ingresó al servidor, se despliegan todos los archivos disponibles en una carpeta en particular y se solicita al usuario el archivo que desea importar desde el servidor y por último se muestra toda la información perteneciente al archivo descargado como el nombre, los bytes y el tiempo que se tardó en transmitirse dicho archivo (figura 2.20).

```
Los archivos disponibles son:
-rw----- 1 1000 1000 40 Dec 04 11:53 archivo.txt
-rw----- 1 1000 1000 228139 Nov 25 22:03 consultarFolleto.jpg
-rw----- 1 1000 1000 94909 Dec 03 00:25 imagen1.jpg
-rw----- 1 1000 1000 6088 Dec 03 00:25 imagen2.png
-rw----- 1 1000 1000 21093 Dec 03 00:25 imagen3.png
-rw----- 1 1000 1000 22089 Dec 03 00:25 map-icon.png
-rw----- 1 1000 1000 61798 Dec 03 20:37 prueba.jpg
-rw----- 1 1000 1000 81028 Dec 03 00:25 ruta.jpg
Ingresa el nombre del archivo que deseas: 'ruta.jpg'

226 Transfer complete.
Tamaño de archivo recibido: 81028 bytes
Tiempo de respuesta: 0.47061 segundos
```

Figura 2.20: Solicitud de descarga de archivo vía FTP.

Por otro lado, en caso de que se presione la opción 2 para subir un archivo al servidor, se muestra únicamente el tiempo de ejecución y de igual manera la respuesta del servidor y la información del archivo transmitido (figura 2.21).

```
---- SENSOR FTP ----
1. Descargar archivo desde servidor
2. Subir archivo a servidor
3. Salir
Selecciona una opcion (numero entero 1-3):2
Ingresa host del servidor: '10.100.67.210'
Ingresa el usuario: 'marce'
Ingresa la contraseña: '1596'

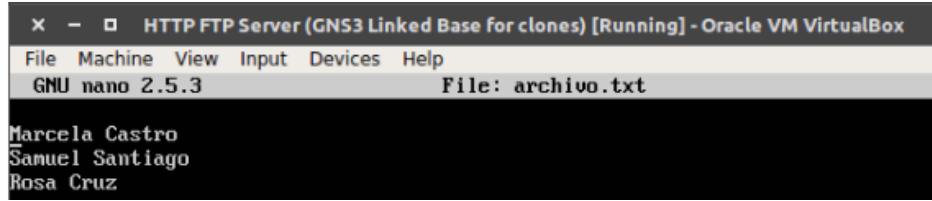
226 Transfer complete.
Archivo transferido: archivo.txt
Tamaño de archivo transferido: 40 bytes
Tiempo de respuesta: 0.00509 segundos
```

Figura 2.21: Información del archivo enviado al servidor FTP.

Si verificamos en el servidor, se observa que el archivo fue transmitidos correctamente y cuando se abre el archivo enviado, se visualizan los nombres de los alumnos integrantes del equipo (figuras 2.22 y 2.23).

```
marce@httpUbuntu:~$ ls
archivo.txt      imagen1.jpg  imagen3.png  prueba.jpg
consultarFolleto.jpg  imagen2.png  map-icon.png  ruta.jpg
marce@httpUbuntu:~$
```

Figura 2.22: Archivos dentro del servidor FTP.



```
GNU nano 2.5.3          File: archivo.txt

Marcela Castro
Samuel Santiago
Rosa Cruz
```

Figura 2.23: Contenido del archivo enviado al servidor FTP.

Por último se muestra el código de ejecución tanto para obtener un archivo desde el servidor, como para subir un nuevo archivo a este (figuras 2.24 y 2.25).

```
def FTP_receive_method():
    ip = str(input('Ingresa host del servidor: '))
    user = str(input('Ingresa el usuario: '))
    pss = str(input('Ingresa la contraseña: '))
    #ip = '192.168.1.69'
    ftp = FTP(ip)
    ftp.login(user,pss)
    #ftp.login('marce','1596')
    ftp.cwd('/home/marce/')
    print 'Los archivos disponibles son: '
    ftp.retrlines('LIST')
    try:
        filename = str(input('Ingresa el nombre del archivo que deseas: '))
        start = time.time()
        tr_response = ftp.retrbinary('RETR '+filename, open(filename, 'wb').write)
        print
        end = time.time()
        print tr_response
        time_difference = end - start
        print 'Tamaño de archivo recibido:', ftp.size(filename), 'bytes'
        print 'Tiempo de respuesta:', "{0:.5f}".format(time_difference), 'segundos'
    except:
        print 'Ocurrió un error al recibir el archivo'
    ftp.quit()
    ftp.close()
```

Figura 2.24: Código para descargar un archivo del servidor FTP.

```

def FTP_upload_method():
    ip = str(input('Ingresa host del servidor: '))
    user = str(input('Ingresa el usuario: '))
    pss = str(input('Ingresa la contraseña: '))
    #ip = '192.168.1.69'
    ftp = FTP(ip)
    ftp.login(user,pss)
    #ftp.login('marce','1596')
    start = time.time()
    ftp.cwd('/home/marce/')
    try:
        ftp.storbinary("STOR " + 'archivo.txt', open('archivo.txt', 'r'))
        end = time.time()
        time_difference = end - start
        print
        print ftp_response
        print 'Archivo transferido: archivo.txt'
        print 'Tamaño de archivo transferido:', ftp.size('/home/marce/archivo.txt'), 'bytes'
        print 'Tiempo de respuesta:', "{0:.5f}".format(time_difference), 'segundos'
    except:
        print 'Ocurrió un error al recibir el archivo'
    #ftp.delete('prueba.jpg')
    ftp.quit()
    ftp.close()

```

Figura 2.25: Código para subir un archivo del servidor FTP.

### 2.3.4. Sensor FTP Server File Count

### 2.3.5. Instalación

Para la implementación del contador de archivos que almacena el servidor FTP se realiza una conexión SSH y se enlistan y se cuentan los archivos en el directorio de almacenamiento del servidor. Para ello se utiliza el siguiente comando:

`ls -l /home/ftp/ | wc -l`

El código se puede observar en la figura 2.26.

```

155 def ftp_counter():
156     ssh_hostname = '10.100.70.39' # IP del servidor
157     ssh_port = 22
158     ssh_username = 'root'
159     ssh_password = 'holal23..'
160
161     paramiko.util.log_to_file('paramiko.log')
162     s = paramiko.SSHClient()
163     s.load_system_host_keys()
164     s.connect(ssh_hostname,ssh_port,ssh_username,ssh_password)
165     stdin, stdout, stderr = s.exec_command("ls -l /home/samuel/ | wc -l")
166     print 'Número de archivos alojados en el servidor FTP: '+stdout.read()
167     s.close()

```

Figura 2.26: Código del contador de archivos en el servidor FTP.

### 2.3.6. Funcionamiento

El funcionamiento se puede ver en la figura 2.27.

```
root@ss:/home/samuel/redes3/Redes3/TercerParcial# python cliente.py
```

```
----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):4

----- SENSOR FTP Server File Count -----
Numero de archivos alojados en el servidor FTP: 10
```

Figura 2.27: Contador de archivos en el servidor FTP.

## 2.4. Supervisión de servidor de impresión (SNMP)

### 2.4.0.1. Configuracion

El proceso de configuracion será aplicado a una distribucion CentOS, el servicio de impresion con el cual se exemplifica el documento es: CUPS. Se inicia el sistema con las credenciales de super usuario root, con la restriccion de tener internet introducimos el comando:

```
yum -y install cups
```

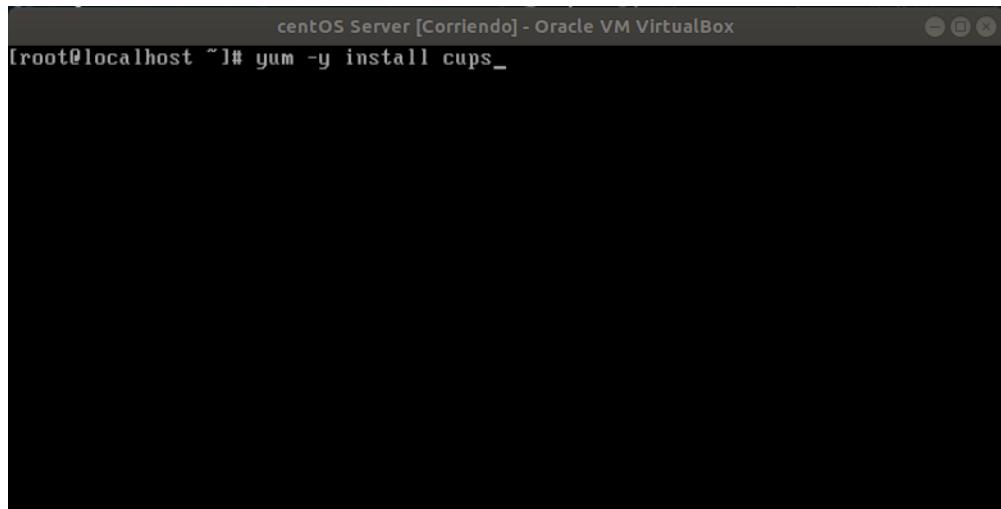


Figura 2.28: comandos.

Con lo cual habremos instalado todo lo necesario para iniciar el servicio de impresion cups, posterior-

mente debemos cambiar el archivo de configuracion por dos razones, la primera de ellas es para poner a la escucha el servidor y que otra pc pueda conectarse, el segundo motivo es para permitir que nuestra pc que ocupamos para el desarrollo pueda utilizar el servicio web que ofrece cups, ya que al tener una pc sin ambiente grafico, no es posible ver el servidor web. Para lograr lo anterior primeramente de forma opcional iniciamos una sesion ssh para conectarnos de forma remota y no tener que cambiar constantemente de equipo, para ello desde nuestra pc de desarrollo introducimos el comando:

```
ssh root@direccion ip servidor ej. ssh root@10.100.76.245
```

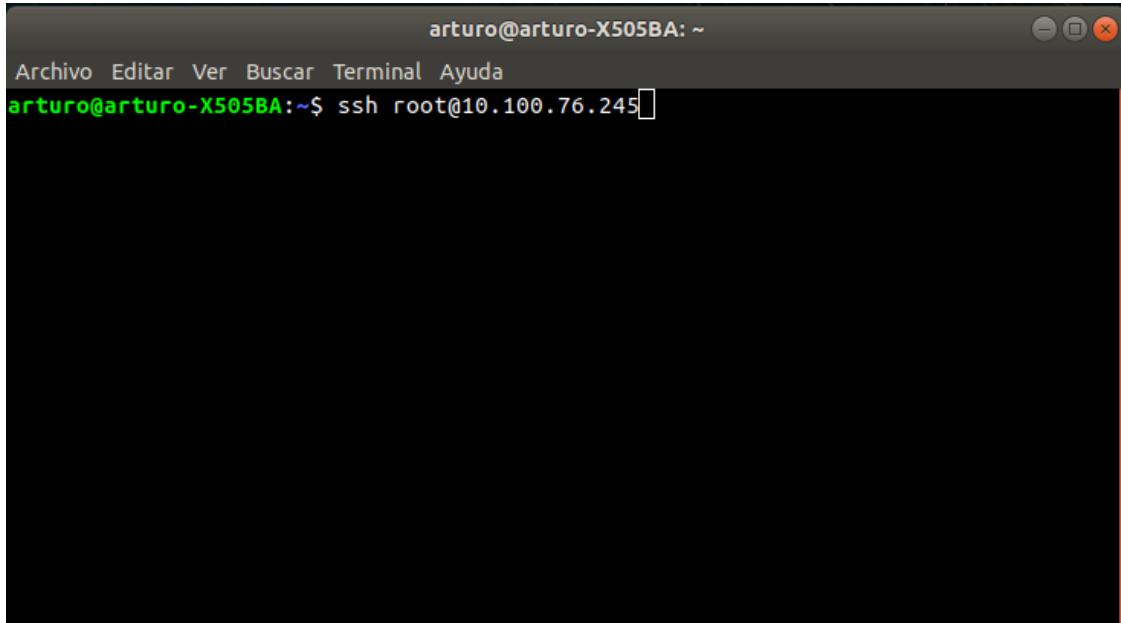
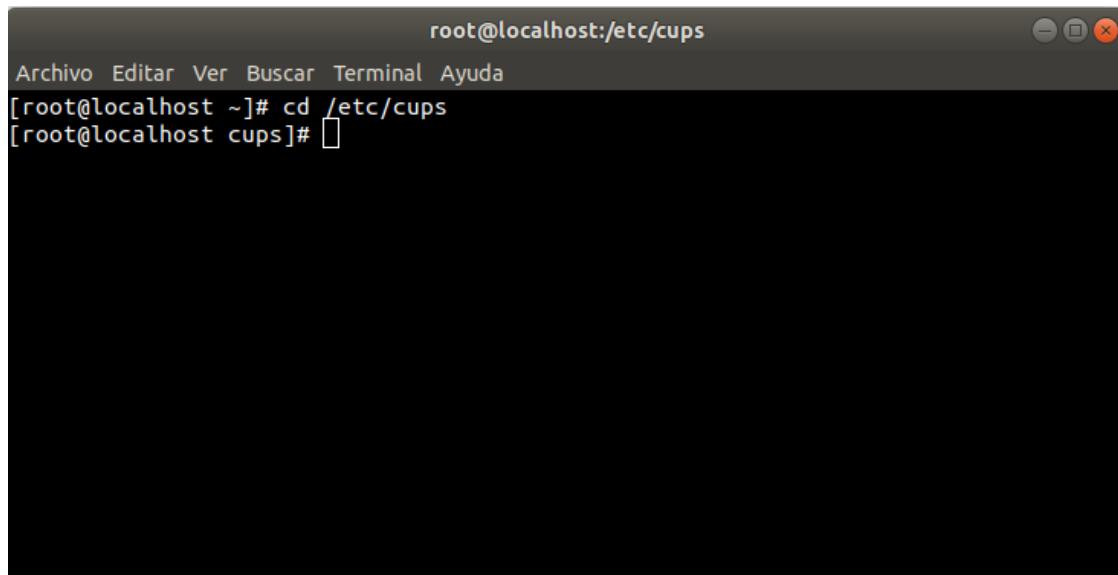


Figura 2.29: comandos.

Posteriormente introducimos la contraseña y ya podemos trabajar en nuestra pc. El siguiente paso es acceder al directorio de archivos de cups:

```
cd /etc/cups
```

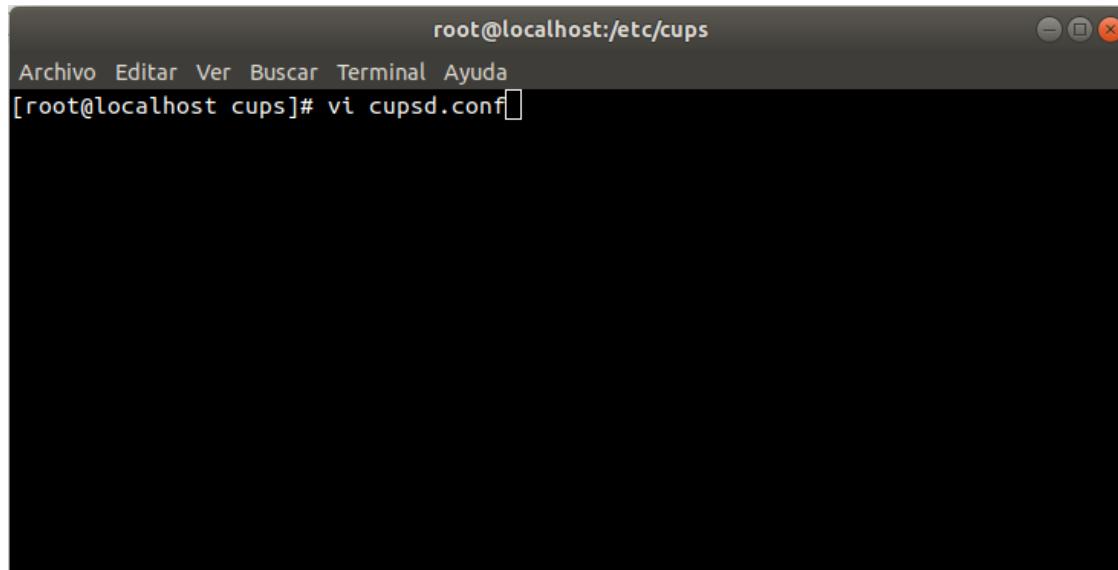


```
root@localhost:/etc/cups
Archivo Editar Ver Buscar Terminal Ayuda
[root@localhost ~]# cd /etc/cups
[root@localhost cups]#
```

Figura 2.30: comandos.

Abrimos el archivo de configuracion cupsd.conf

```
vi cupsd.conf
```



```
root@localhost:/etc/cups
Archivo Editar Ver Buscar Terminal Ayuda
[root@localhost cups]# vi cupsd.conf
```

Figura 2.31: comandos.

Agregamos una linea LISTEN para poner a la escucha nuestro servidor instroduciendo como parametro la direccion ip del servidor que tiene instalado el servicio de impresiones, indicando el puerto por defecto del servicio de impresion que es el 631.

**LISTEN ip servidor:631 ej. LISTEN 10.100.76.245:631**

```
root@localhost:/etc/cups
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@localhost:/etc/cups x arturo@arturo-X505BA: ~ x + ▾
#
# "$Id: cupsd.conf.in 7888 2008-08-29 21:16:56Z mike $"
#
# Sample configuration file for the CUPS scheduler. See "man cupsd.conf" for a
# complete description of this file.
#
# Log general information in error_log - change "warn" to "debug"
# for troubleshooting...
LogLevel warn

# Only listen for connections from the local machine.
Listen 10.100.76.245:631
Listen localhost:631
Listen /var/run/cups/cups.sock

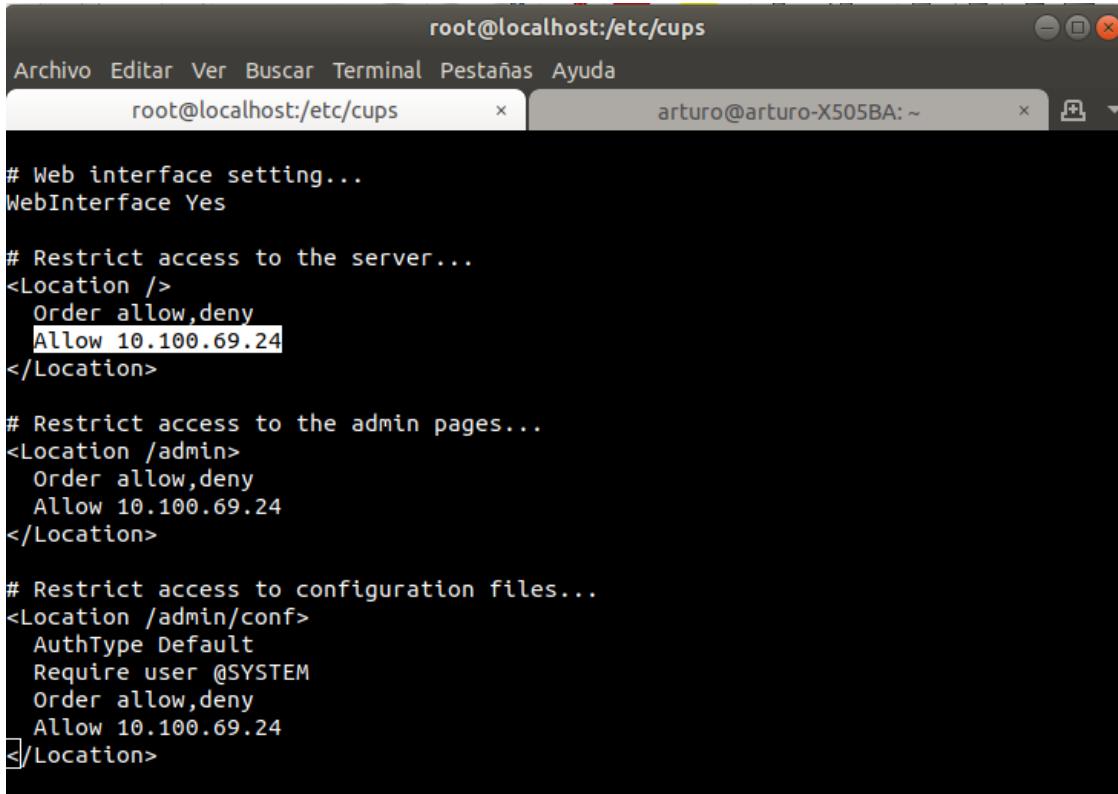
# Show shared printers on the local network.
Browsing On
BrowseLocalProtocols dnssd

# Default authentication type, when authentication is required...
DefaultAuthType Basic
```

Figura 2.32: configuración del archivo cupsd.conf

Posteriormente agregamos tres linea mas que implican el permiso a cierta dirección ip para que pueda utilizar o acceder al servicio y configuración, lo logramos introduciendo una instrucción Allow seguido de la dirección ip de la pc de desarrollo

Allow ip pc desarrollo ej. Allow 10.100.69.24



```
# Web interface setting...
WebInterface Yes

# Restrict access to the server...
<Location />
    Order allow,deny
    Allow 10.100.69.24
</Location>

# Restrict access to the admin pages...
<Location /admin>
    Order allow,deny
    Allow 10.100.69.24
</Location>

# Restrict access to configuration files...
<Location /admin/conf>
    AuthType Default
    Require user @SYSTEM
    Order allow,deny
    Allow 10.100.69.24
</Location>
```

Figura 2.33: configuración de la ip de la pc location

```
root@localhost:/etc/cups
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@localhost:/etc/cups x arturo@arturo-X505BA: ~ x + ▾

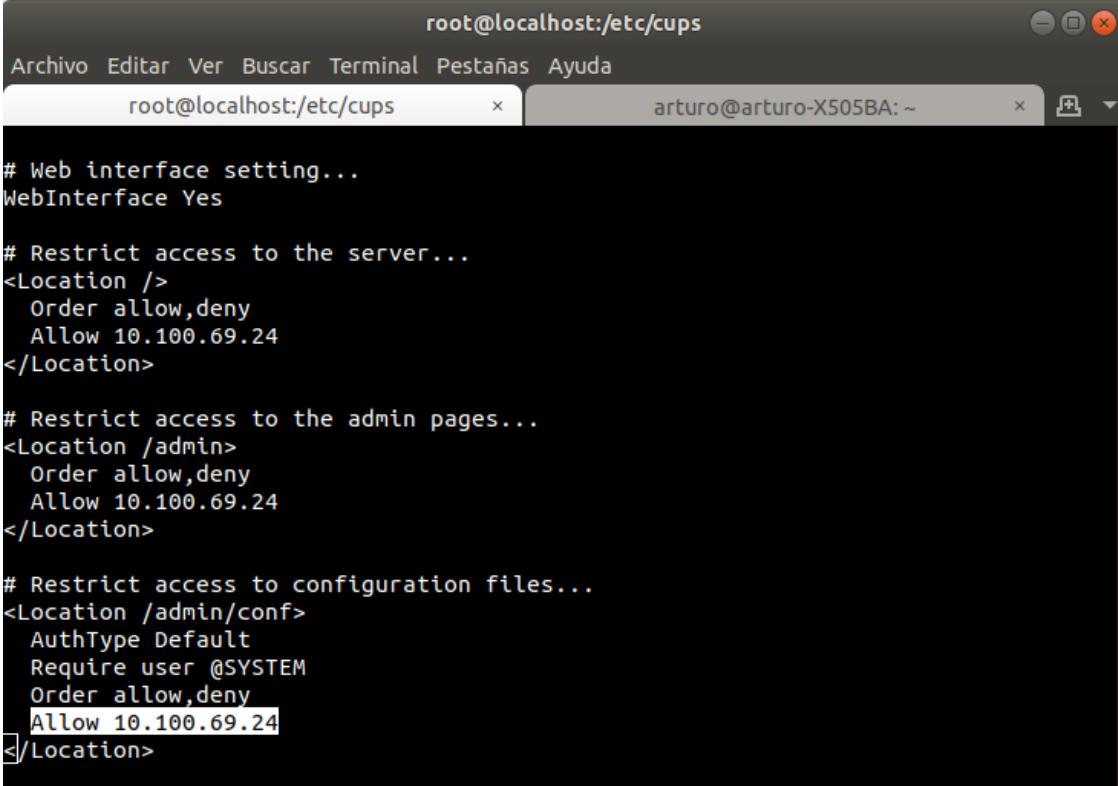
# Web interface setting...
WebInterface Yes

# Restrict access to the server...
<Location />
  Order allow,deny
  Allow 10.100.69.24
</Location>

# Restrict access to the admin pages...
<Location /admin>
  Order allow,deny
  Allow 10.100.69.24
</Location>

# Restrict access to configuration files...
<Location /admin/conf>
  AuthType Default
  Require user @SYSTEM
  Order allow,deny
  Allow 10.100.69.24
</Location>
```

Figura 2.34: configuracion de la ip de la pc admin



```

root@localhost:/etc/cups
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@localhost:/etc/cups x arturo@arturo-X505BA: ~ x
# Web interface setting...
WebInterface Yes

# Restrict access to the server...
<Location />
  Order allow,deny
  Allow 10.100.69.24
</Location>

# Restrict access to the admin pages...
<Location /admin>
  Order allow,deny
  Allow 10.100.69.24
</Location>

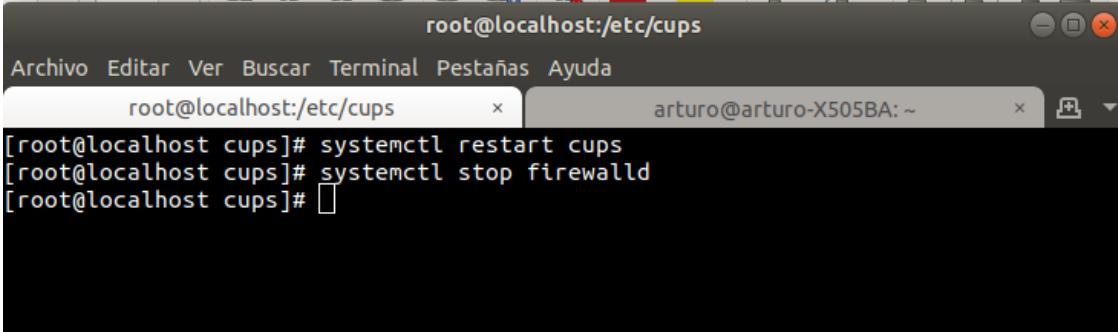
# Restrict access to configuration files...
<Location /admin/conf>
  AuthType Default
  Require user @SYSTEM
  Order allow,deny
  Allow 10.100.69.24
</Location>

```

Figura 2.35: configuracion de la ip de la pc admin/conf

Guardamos el archivo, reiniciamos el servicio de impresion y apagamos el firewall.

**systemctl restart cups systemctl stop firewalld**



```

root@localhost:/etc/cups
Archivo Editar Ver Buscar Terminal Pestañas Ayuda
root@localhost:/etc/cups x arturo@arturo-X505BA: ~ x
[root@localhost cups]# systemctl restart cups
[root@localhost cups]# systemctl stop firewalld
[root@localhost cups]#

```

Figura 2.36: reinicio y apago de firewall

Ahora probamos el servicio en nuestra pc de desarrollo introduciendo en un navegador la ip del servidor y el puerto. Con esto hemos completado la configuración:

**ej. <http://192.168.100.12:631>**

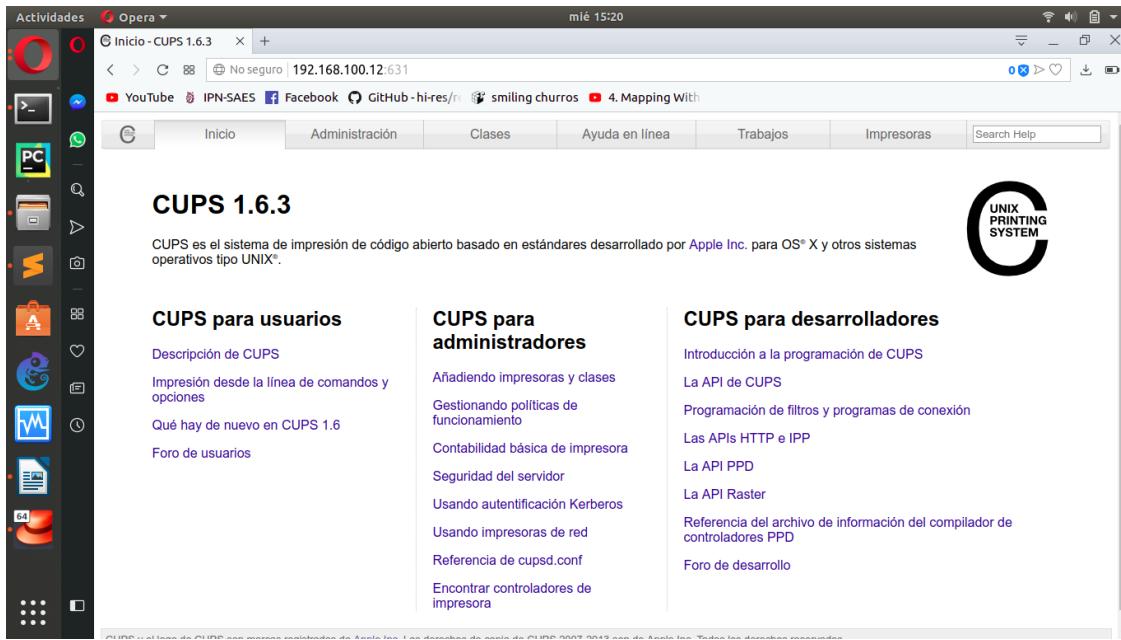


Figura 2.37: servicio activo

#### 2.4.0.2. Monitorizacion de impresoras

Anexamos las capturas de pantalla del código generado, con una nota extra, de que no contábamos con impresoras con tarjeta de red por lo que no fue posible obtener su información vía SNMP, posteriormente se explica cómo se solucionó el problema mediante SSH.

La función **monitorear** en la línea 46 2.39 tiene todo el procedimiento de tres aspectos fundamentales que son la conexión, la captura de datos y finalmente el tratamiento de la información para ser mostrada.

La función **sshHilo** dentro de monitorear 2.43, que podemos encontrar en la línea 181, establecemos una conexión SSH al servidor que cuenta con el servicio de impresión mediante la **librería pexpect**, introducimos las credenciales de inicio de sesión, lo más importante recae en la línea 188, en donde se ejecuta el comando **lpstat -l -t** que nos junta la información de las impresoras que tiene conectadas, posteriormente cerramos esa sesión y abrimos un archivo llamado **impresoras.txt** donde se almacena la información obtenida del comando **lpstat**.

La evaluación del **if** en la línea 51 2.39 nos indica si hubo un problema en la conexión y nos muestra una ventana de error. De otro modo la función **establecer info** nos retorna un indicador si es que el servicio de impresión tiene conectada alguna impresora, en caso de no encontrar, nos muestra otra ventana indicando que no tiene impresoras por el momento.

Finalmente al pasar por estos filtros se organiza la información para tener cierto formato y el usuario pueda obtener el estado de las impresoras.

```

0
7     _server = '192.168.100.12'
8     _user = 'root'
9     _pass = '123'
10    reporte = []
11    informacion = []
12    num_impresoras = 0
13    _Background = '#BDC3C7' #'#410be2'
14
15    class Aplicacion():
16        def __init__(self):
17            self.raiz = Tk()
18            self.raiz.title('Impresiones')
19            self.raiz.geometry('300x200+300+300')
20            self.raiz.configure(background=_Background)
21            self.estilos()
22            self.variables()
23            self.insertar()
24            self.raiz.mainloop()
25
26        def estilos(self):
27            self.style = ttk.Style()
28            self.style.configure("LBL.TLabel", foreground="Black", background=_Background, font='verdana 12')
29            self.style.configure("ENT.TCombobox", background=_Background, font='verdana 12')
30            self.style.map("BTN.TButton",
31                           foreground=[('pressed', '#6f16ce'), ('active', 'white')],
32                           background=[('pressed', '!disabled', '#726e77'), ('active', '#d11b91')] # 38c0ed
33                           )
34            self.style.configure("BTN.TButton", font='verdana 12', foreground='white', background='#d11b91')
35
36        def variables(self):
37            self.bandera = BooleanVar(value=True)
38            self.listo = BooleanVar(value=False)
39
40        def insertar(self):
41            botonEntrar = ttk.Button(self.raiz, text='Monitorear', command=self.monitorear, style='BTN.TButton')
42            botonCerrar = ttk.Button(self.raiz, text='Salir', command=self.cerrar, style='BTN.TButton')
43            botonEntrar.pack(pady=15)

```

Figura 2.38: code

```

44
45        botonCerrar.pack(pady=20)
46
47    def monitorear(self):
48
49        self.cads = []
50        global informacion
51        res = self.sshHilo()
52        if res == -1:
53            self.ventanaError()
54        else:
55            res = self.establecer_info()
56            if res == -1:
57                self.ventanaError()
58            else:
59                print 'num: ' + str(num_impresoras)
60                for i in range(num_impresoras):
61                    informacion.append([])
62                    self.cads.append('Sin informacion')
63                print 'test'
64                for i in self.cads:
65                    print i
66                self.establece_cadenas()
67
68                self.raiz.withdraw()
69                ventana = Toplevel()
70                ventana.geometry('530x450+200+200')
71                ventana.title('Servicio de impresion')
72                ventana.configure(background=_Background)
73
74                etiquetas = []
75                for i in range(num_impresoras):
76                    etiquetas.append(ttk.Label(ventana, text=self.cads[i], style='LBL.TLabel'))
77                for i in range(num_impresoras):
78                    etiquetas[i].grid(row=i, column=0, pady=15)
79                boton = ttk.Button(ventana, text='Cerrar', command=lambda: self.cerrarVentana(ventana), style='BTN.TButton')
80                boton.grid(row=num_impresoras, column=0)

```

Figura 2.39: code

```
81     def ventanaError(self):
82         self.raiz.withdraw()
83         ventana = Toplevel()
84         ventana.geometry('530x150+200+200')
85         ventana.title('Error')
86         ventana.configure(background=_Background)
87
88         et = ttk.Label(ventana, text='Ocurrió un error con la conexión en el servidor :(', style='LBL.TLabel')
89         et.pack(pady=20)
90
91         boton = ttk.Button(ventana, text='Cerrar', command=lambda: self.cerrarVentana(ventana), style='BTN.TButton')
92         boton.pack()
93
94     def ventanaError2(self):
95         self.raiz.withdraw()
96         ventana = Toplevel()
97         ventana.geometry('530x150+200+200')
98         ventana.title('Sin impresoras')
99         ventana.configure(background=_Background)
100
101        et = ttk.Label(ventana, text='Por el momento no hay impresoras conectadas :(', style='LBL.TLabel')
102        et.pack(pady=20)
103
104        boton = ttk.Button(ventana, text='Cerrar', command=lambda: self.cerrarVentana(ventana), style='BTN.TButton')
105        boton.pack()
106
107    def cerrarVentana(self, ventana):
108        reporte = []
109        información = []
110        num_impresoras = 0
111        ventana.destroy()
112        self.raiz.deiconify()
113
114
115
116    def establece_cadenas(self):
117        ...
```

Figura 2.40: code

```
115 def establece_cadenas(self):
116     #while (self.bandera.get()):
117     global informacion
118     global reporte
119     if num_impresoras != 0:
120         for i in range(num_impresoras):
121             tmp = ''
122             tmp = tmp + reporte[i][2] + '\n'
123             tmp = tmp + reporte[i][1] + '\n'
124             tmp = tmp + reporte[i][3] + '\n'
125             tmp = tmp + reporte[i][0]
126
127             self.cads[i] = tmp
128         print 'informacion:' + str(num_impresoras)
129
130
131
132 def establecer_info(self):
133     fd = open('impresoras.txt', 'r')
134     lista = fd.read().splitlines()
135     fd.close()
136     tam = len(lista)
137     impresoras = []
138     conexiones = []
139     global reporte
140     tam_impresoras = 0
141
142     # obtener las conexiones
143     for i in range(tam):
144         if lista[i].find('usb') != -1:
145             tam_impresoras += 1
146             impresoras.append('usb' + str(lista[i].split('usb')[1]))
147             if lista[i].find('la impresora') != -1:
148                 impresoras.append(str(lista[i].split('.')[0].split(' ')[-1]))
149                 if lista[i].find('Descripci') != -1:
150                     impresoras.append(str(lista[i].split(':')[1]))
151                     if lista[i].find('Conexi') != -1:
```

Figura 2.41: code

```
151             if lista[i].find('Conexi') != -1:
152                 impresoras.append(str(lista[i].split(':')[1]))
153
154     global num_impresoras
155     num_impresoras = tam_impresoras
156     if num_impresoras == 0:
157         return -1
158
159     for i in range(tam_impresoras):
160         conexiones.append(impresoras[0])
161         impresoras.pop(0)
162
163     for i in range(tam_impresoras):
164         reporte.append([])
165
166     for i in range(tam_impresoras):
167         tmp = 'puerto: ' + str(conexiones[i])
168         reporte[i].append(tmp)
169         tmp = 'estado: ' + str(impresoras[0])
170         reporte[i].append(tmp)
171         impresoras.pop(0)
172         tmp = 'Impresora: ' + str(impresoras[0])
173         reporte[i].append(tmp)
174         impresoras.pop(0)
175         tmp = 'Conexion: ' + str(impresoras[0])
176         reporte[i].append(tmp)
177         impresoras.pop(0)
178     print reporte
179     return 0
180
181 def sshHilo(self):
182     try:
183         s = pxssh.pxssh()
184         hostname = _server
185         user = _user
186         password = _pass
187         s.login(hostname, user, password)
188         print s.before
189
190         for i in range(tam_impresoras):
191             if lista[i].find('Conexi') != -1:
192                 impresoras.append(str(lista[i].split(':')[1]))
193
194     global num_impresoras
195     num_impresoras = tam_impresoras
196     if num_impresoras == 0:
197         return -1
198
199     for i in range(tam_impresoras):
200         conexiones.append(impresoras[0])
201         impresoras.pop(0)
202
203     for i in range(tam_impresoras):
204         reporte.append([])
205
206     for i in range(tam_impresoras):
207         tmp = 'puerto: ' + str(conexiones[i])
208         reporte[i].append(tmp)
209         tmp = 'estado: ' + str(impresoras[0])
210         reporte[i].append(tmp)
211         impresoras.pop(0)
212         tmp = 'Impresora: ' + str(impresoras[0])
213         reporte[i].append(tmp)
214         impresoras.pop(0)
215         tmp = 'Conexion: ' + str(impresoras[0])
216         reporte[i].append(tmp)
217         impresoras.pop(0)
218     print reporte
219     return 0
```

Figura 2.42: code

```
181     def sshHilo(self):
182         try:
183             s = pxssh.pxssh()
184             hostname = _server
185             user = _user
186             password = _pass
187             s.login(hostname, user, password)
188             s.sendline('lpstat -l -t')
189             s.prompt()
190             lista = s.before
191             # print lista
192             s.logout()
193
194             fd = open('impresoras.txt', 'w')
195             fd.write(lista)
196             fd.close()
197             return 0
198         except:
199             print 'Error en la conexion SSH'
200             return -1
201
202     def cerrar(self):
203         self.raiz.destroy()
204
205
206     def main():
207         app = Aplicacion()
208
209     if __name__ == '__main__':
210         main()
```

Figura 2.43: code

#### 2.4.0.3. Funcionamiento

Primero nos encontramos con un pequeño menú



Figura 2.44: menú

Dado caso que el servidor no se encunetre activo se manda un mensaje de alerta.

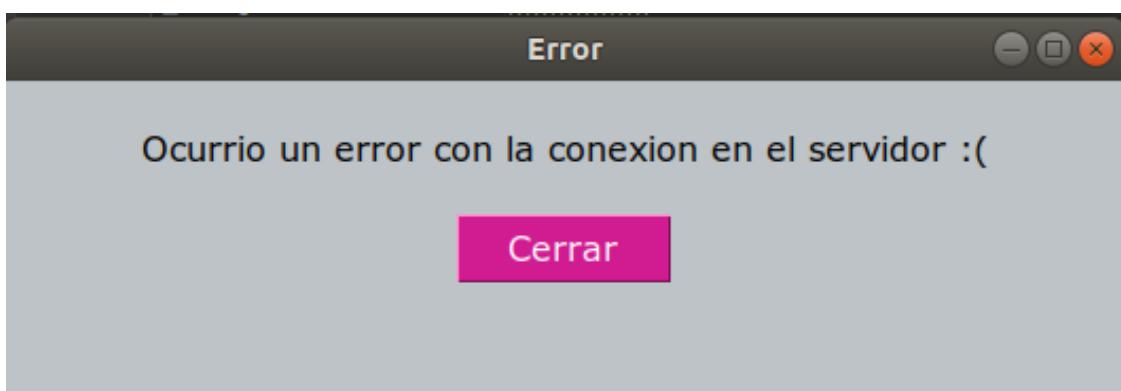


Figura 2.45: sin servidor

De igual manera si no se encuentra una impresora se mostrara el siguiente mensaje [2.46](#)



Figura 2.46: sin impresora

Cuando se tiene el servidor corriendo y las impresoras conectadas nos mostrará la siguiente información que es el nombre de la impresora, su estado, conexión y en este caso el puerto. [2.47](#)

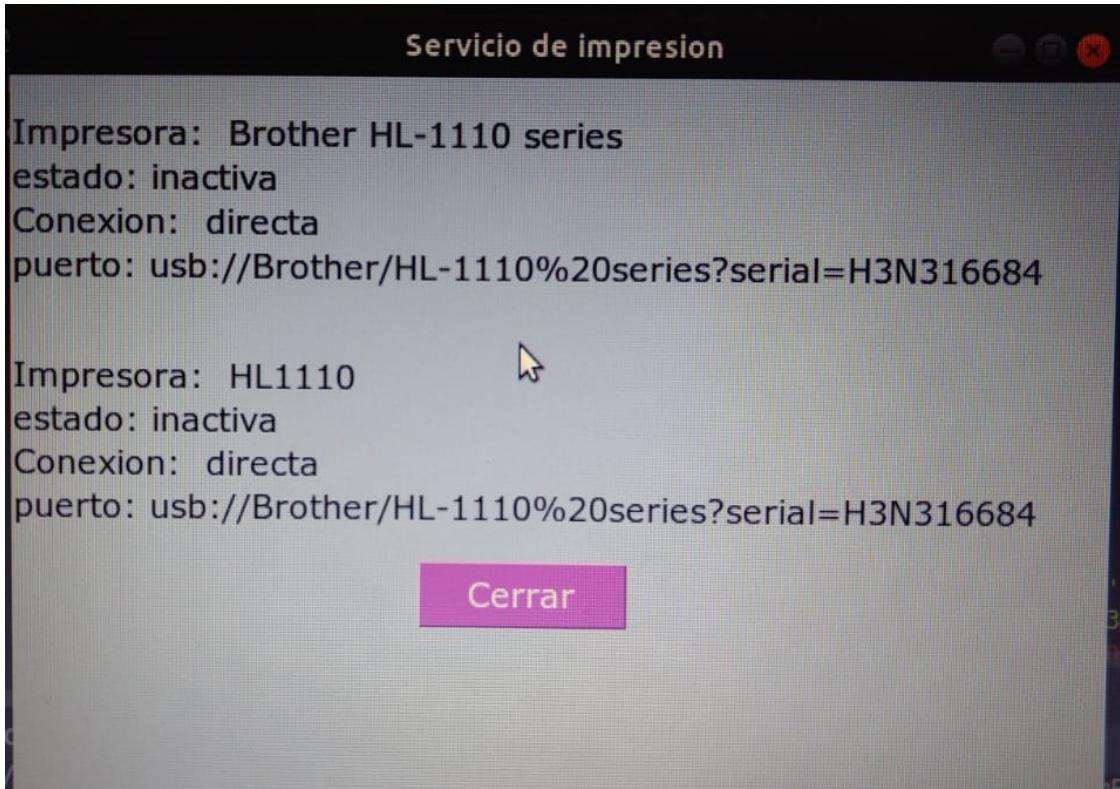


Figura 2.47: Impresoras conectadas

## 2.5. Supervisión de servidor de servidor de acceso remoto (SSH)

### 2.5.1. Instalación

Para el desarrollo de esta práctica se optó por implementar el protocolo SSH *Secure Shell*. La instalación de este servidor de acceso remoto es muy simple. Basta con ejecutar el comando:

```
apt-get install openssh-server
```

En caso de que el servicio no sea iniciado automáticamente después de la instalación se ejecuta el comando:

```
service sshd start
```

## 2.5.2. Sensor SSH

Para la implementación del sensor se monitorizaron 4 aspectos: el número total de conexiones, dirección IP origen, dirección IP destino y usuario al que se encuentra conectada la sesión. Para obtener estos datos se ejecuta la función `sensor_ssh()` del programa `cliente.py`. En la figura 2.48 se puede observar a detalle el código.

```

169 def sensor_ssh():
170     ssh_hostname = '10.100.70.39' # IP del servidor
171     ssh_port = 22
172     ssh_username = 'root'
173     ssh_password = 'holai23.'
174
175     paramiko.util.log_to_file('paramiko.log')
176     s = paramiko.SSHClient()
177     s.load_system_host_keys()
178     s.connect(ssh_hostname,ssh_port,ssh_username,ssh_password)
179     stdin, stdout, stderr = s.exec_command("netstat -tnpa | grep 'ESTABLISHED.*sshd' | wc -l")
180     num = int(stdout.read())
181     print "Número de conexiones SSH en el servidor: "+str(num)
182     if(num > 0):
183         print Destino      Origen          Usuario\n"
184         stdin, stdout, stderr = s.exec_command("netstat -tnpa | grep 'ESTABLISHED.*sshd' | tr -s ' ' | cut -d' ' -f4,5,8")
185         print stdout.read()
186     s.close()

```

Figura 2.48: Código del sensor SSH.

En esta función podemos ver que se realiza una conexión SSH al servidor para ejecutar el comando `netstat -tnpa | grep 'ESTABLISHED.*sshd' | wc -l` para obtener el total de conexiones SSH establecidas. Si el número de conexiones es mayor a cero se ejecuta el comando `netstat -tnpa | grep 'ESTABLISHED.*sshd' | tr -s ' ' | cut -d' ' -f4,5,8` con lo cual se obtiene la información requerida por cada conexión establecida.

### 2.5.2.1. Funcionamiento

La ejecución de este sensor se puede observar en la figura 2.49

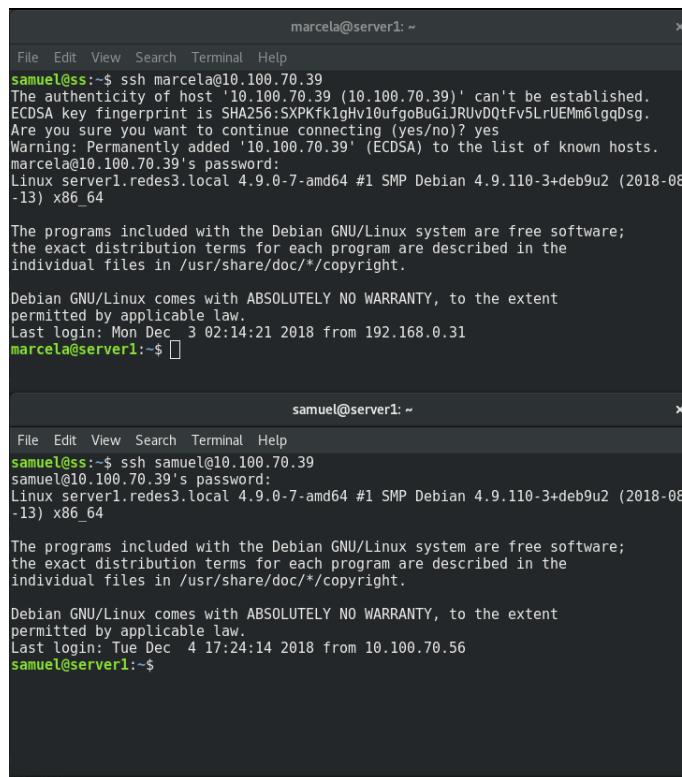
```
root@ss:/home/samuel/redes3/Redes3/TercerParcial# python cliente.py

----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):6

----- SENSOR SSH -----
Numero de conexiones SSH en el servidor: 3
Destino          Origen          Usuario
10.100.70.39:22 10.100.70.56:44370    marcela
10.100.70.39:22 10.100.70.56:44378    root@not
10.100.70.39:22 10.100.70.56:44376    samuel
```

Figura 2.49: Funcionamiento del sensor SSH.

Este resultado se obtiene al tener iniciadas dos sesiones de SSH como se muestra en la figura 2.50.



The screenshot shows two terminal windows. The top window is titled 'marcela@server1: ~' and the bottom window is titled 'samuel@server1: ~'. Both windows have a standard Linux terminal interface with a dark background and light-colored text. The top window displays the command 'ssh marcela@10.100.70.39' and its execution details, including host key fingerprint verification and password entry. The bottom window displays the command 'ssh samuel@10.100.70.39' and its execution details, also including host key fingerprint verification and password entry. Both windows show the Debian 4.9.110-3+deb9u2 (2018-08-13) x86\_64 system information at the bottom.

Figura 2.50: Conexiones SSH.

Es importante recalcar que el sensor reporta 3 sesiones ya que también cuenta la sesión que utiliza el mismo sensor para obtener los datos.

## 2.6. Administración de archivos de configuración

La administración de archivos de configuración se divide en dos partes. La importación y la exportación. Para ambos casos se utilizó FTP como protocolo de intercambio de archivos. Ambas funciones se encuentran definidas dentro del programa **cliente.py** y cada una de ellas necesita un archivo de configuración en el que se definen las direcciones IP destino u origen, rutas de archivos, usuarios y contraseñas, etc.

### 2.6.0.1. Importación

El archivo de configuración para la función de importación se puede observar en la figura 2.51. Podemos observar que en la parte superior se define la sintaxis de cada regla. Esto permite agregar los parámetros de diferentes routers y que la ejecución del programa sea fluido.

```
GNU nano 2.7.4                                         File: importar.conf

#Archivo de configuración para importar archivos desde routers
#La ruta destino FTP por defecto de los routers es /home/rcp/
#Sintaxis:
#<Direccion_IP_origen> <Archivo_origen> <Ruta_destino> <usuario> <pass> <ID>

#192.168.202.15 archivito.txt /home/samuel/redes3/archivito.txt rcp rcp R2
30.30.30.2 startup-config /home/samuel/redes3/Redes3/TercerParcial/startup-config-maestra rcp rcp RM
```

Figura 2.51: Archivo de configuracion para importaciones.

El código de la función **importar()** se puede ver en la figura 2.52. Que consiste básicamente en un ciclo que busca líneas que no comiencen con el carácter #, es decir, que no estén comentadas. Una vez que obtiene una línea descompone la cadena y obtiene los parámetros necesarios para realizar una conexión FTP y la respectiva descarga del archivo (también definido en el archivo de configuración).

```
199 ▼ def importar():
200 ▼   with open("importar.conf", 'r') as exportar:
201 ▼     for linea in exportar.readlines():
202       linea = linea.rstrip().split(' ')
203       if len(linea[0]) > 0:
204         if linea[0][0] != '#':
205           print linea
206           ftp = FTP(linea[0]) # Connect
207           ftp.login(linea[3],linea[4]) # Login
208           file = open(linea[2]+'_'+linea[5]+'.txt', 'w') # Abre archivo local
209           ftp.retrbinary('RETR '+linea[1],file.write) # Upload
210           ftp.quit()
```

Figura 2.52: Código para importar archivos.

El funcionamiento se puede observar en la figura 2.53.

```
root@ss:/home/samuel/redes3/Redes3/TercerParcial# python cliente.py

----- Menu de opciones -----
1. Sensor SMTP
2. Sensor HTTP
3. Sensor FTP
4. Sensor FTP Server File Count
5. Sensor de impresion
6. Sensor de acceso remoto
7. Administracion de archivos de configuracion
8. Salir
Selecciona una opcion (numero entero 1-8):7

----- ADMINISTRACION DE ARCHIVOS DE CONFIGURACION ----

----- Menu de opciones de archivos de configuracion-----
1. Importar archivos de configuracion
2. Exportar archivos de configuracion
3. Regresar
Selecciona una opcion (numero entero 1-3):1
```

Figura 2.53: Funcionamiento de la importación de archivos.

#### 2.6.0.2. Exportación

El archivo de configuración para la función de exportación se puede observar en la figura 2.54. Podemos observar que en la parte superior se define la sintaxis de cada regla. Al igual que la importación, esto permite agregar los parámetros de diferentes routers y que la ejecución del programa sea fluido.

```
GNU nano 2.7.4                               File: exportar.conf

#Archivo de configuración para exportar archivos a routers
#La ruta destino FTP por defecto de los routers es /home/rcp/
#Sintaxis:
#<Direccion_IP_destino> <Archivo_origen> <Nombre_destino> <usuario> <pass>

#192.168.232.1 /home/samuel/redes3/archivito.txt archivito.txt rcp rcp
30.30.30.2 /home/samuel/redes3/Redes3/TercerParcial/startup-config startup-config-equipo2 rcp rcp
```

Figura 2.54: Archivo de configuracion para exportaciones.

El código de la función **exportar()** se puede ver en la figura 2.55. Que, al igual que **importar()**, consiste básicamente en un ciclo que busca líneas que no comiencen con el carácter #, es decir, que no estén comentadas. Una vez que obtiene una línea descompone la cadena y obtiene los parámetros necesarios para realizar una conexión FTP y la respectiva subida del archivo (también definido en el archivo de configuración).

```
212 def exportar():
213     with open("exportar.conf",'r') as exportar:
214         for linea in exportar.readlines():
215             linea = linea.rstrip().split(' ')
216             if len(linea[0]) > 0:
217                 if linea[0][0] != '#':
218                     print linea
219                     ftp = FTP(linea[0]) # Connect
220                     ftp.login(linea[3],linea[4]) # Login
221                     file = open(linea[1],'rb') # Abre archivo local
222                     ftp.storbinary('STOR %s' % linea[2],file) # Upload
223                     ftp.retrlines('LIST') # Lista archivos
224                     ftp.quit()
```

Figura 2.55: Código para exportar archivos.

El funcionamiento se puede observar en la figura 2.56.

```
---- EXPORTART ARCHIVOS DE CONFIGURACION ----
'30.30.30.2', '/home/samuel/redes3/Redes3/TercerParcial/startup-config', 'startup-config-equipo2', 'rcp', 'rcp']
rw----- 1 0          0                      0 Dec  4 17:23 leases
rw-r--r-- 1 1000      1000                  714 Dec  4 18:01 startup-config-equipo2
rwxr-xr-x 2 1000      1000                  1024 Jun  6 2018 tftpboot
rw----- 1 1000      1000                  7270 Dec  4 18:00 xmlstats.xml

---- Menu de opciones de archivos de configuracion----
. Importar archivos de configuracion
. Exportar archivos de configuracion
. Regresar
elecciona una opcion (numero entero 1-3):■
```

Figura 2.56: Funcionamiento de la exportación de archivos.

# CAPÍTULO 3

## Configuración de routers

Para el desarrollo de esta práctica, se desarrolló la topología siguiente en la cuál se conectaron 3 computadoras en total siendo la primera la que tenía implementado el código de Python que fungía como cliente (figura 3.1) y las otras dos que contenían los servidores configurados (figuras 3.2 y 3.3).

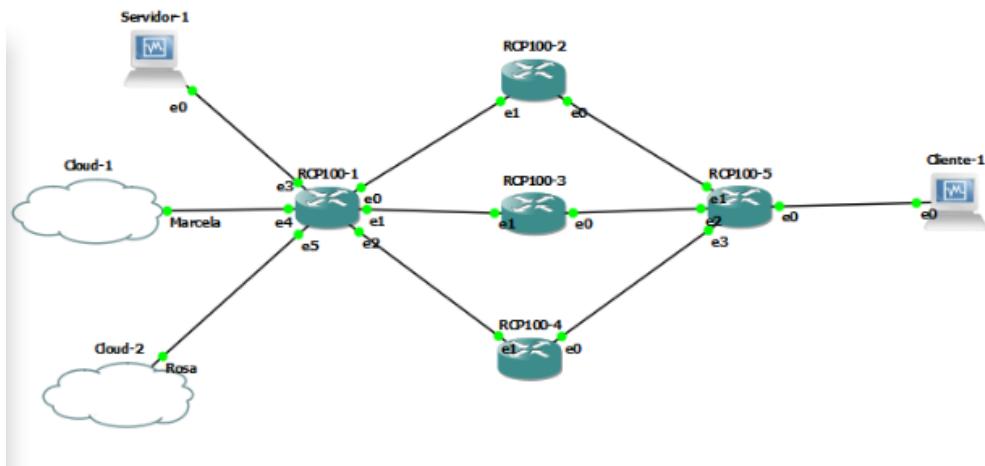


Figura 3.1: Computadora 1.

En las figuras mostradas en la parte inferior se observa únicamente una cloud conectada con su propia computadora pues se realizó la conexión por medio de un túnel UDP.

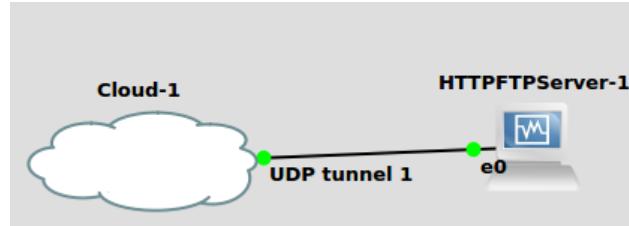


Figura 3.2: Computadora 2.

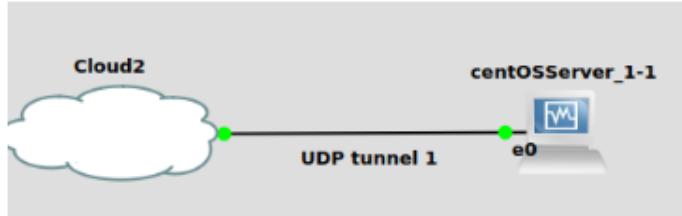


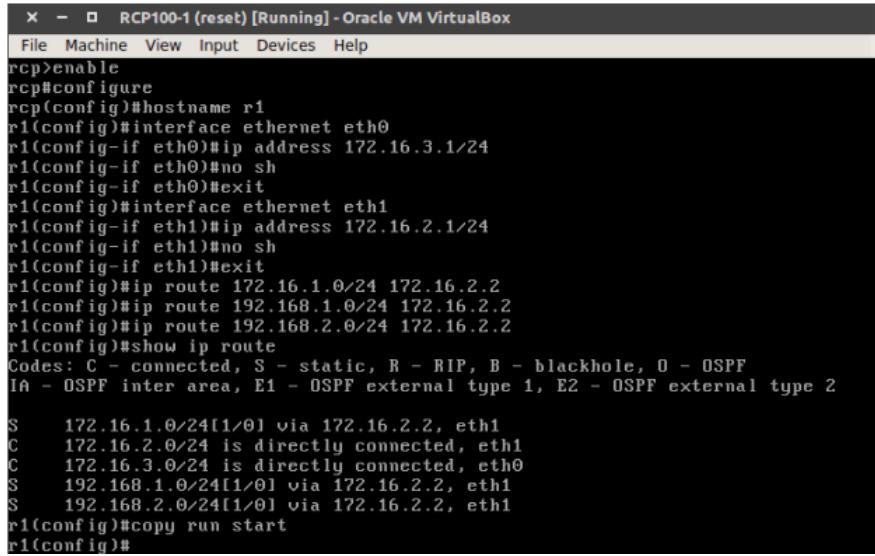
Figura 3.3: Computadora 3.

La funcionalidad de esta topología constaba en comunicar el cliente desde donde se ejecutaron los sensores que monitorizaban el funcionamiento de cada uno de servidores almacenados en las computadoras 2 y 3.

La configuración de estos routers se realizaron por los 3 diferentes métodos que eran estático, RIP y OSPF. A continuación se muestran algunos de los comandos necesarios para la realización de dicha configuración en cada uno de los enrutadores.

#### Enrutamiento estático

La tabla de enrutamiento contiene la información más importante que usan los routers. Esta tabla proporciona la información que usan los routers para reenviar los paquetes recibidos. Si la información de la tabla de enrutamiento no es correcta, el tráfico se reenviará incorrectamente y posiblemente no llegue al destino. Para que se comprendan las rutas de tráfico, la resolución de problemas y la manipulación del tráfico, es absolutamente necesario que se tengan conocimientos sólidos sobre cómo leer y analizar una tabla de enrutamiento [1].



```

X - RCP100-1 (reset) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
rcp>enable
rcp#configure
rcp(config)#hostname r1
r1(config)#interface ethernet eth0
r1(config-if eth0)#ip address 172.16.3.1/24
r1(config-if eth0)#no sh
r1(config-if eth0)#exit
r1(config)#interface ethernet eth1
r1(config-if eth1)#ip address 172.16.2.1/24
r1(config-if eth1)#no sh
r1(config-if eth1)#exit
r1(config)#ip route 172.16.1.0/24 172.16.2.2
r1(config)#ip route 192.168.1.0/24 172.16.2.2
r1(config)#ip route 192.168.2.0/24 172.16.2.2
r1(config)#show ip route
Codes: C - connected, S - static, R - RIP, B - blackhole, O - OSPF
IA - OSPF inter area, E1 - OSPF external type 1, E2 - OSPF external type 2

S 172.16.1.0/24[1/0] via 172.16.2.2, eth1
C 172.16.2.0/24 is directly connected, eth1
C 172.16.3.0/24 is directly connected, eth0
S 192.168.1.0/24[1/0] via 172.16.2.2, eth1
S 192.168.2.0/24[1/0] via 172.16.2.2, eth1
r1(config)#copy run start
r1(config)#

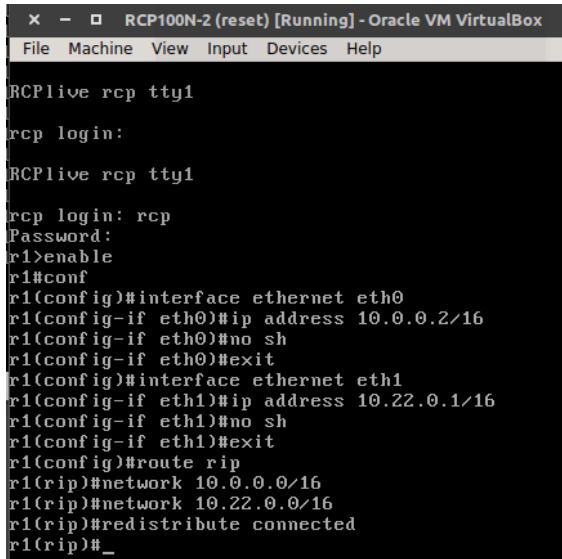
```

Figura 3.4: Configuración enrutamiento estático.

### Enrutamiento RIP

RIP es un protocolo dinámico y tiene 2 versiones (RIP y RIPv2). La versión 1 de RIP es con clase (no soporta VLSM), no utiliza autenticación y utiliza broadcast. La versión 2 de RIP es sin clase (soporta VLSM), añade la autenticación y utiliza multicast. La base de la configuración del protocolo RIP en Cisco se encuentra en el comando network [2]. Dicho comando cumple con dos propósitos:

- Informar a RIP sobre qué interfaces intervienen en el envío y recepción de actualizaciones de enrutamiento.
- Pedir a RIP que anuncie a los demás routers la existencia de la red.



```

X - RCP100N-2 (reset) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
RCPlive rcp tty1
rcp login:
RCPlive rcp tty1
rcp login: rcp
Password:
r1>enable
r1#conf
r1(config)#interface ethernet eth0
r1(config-if eth0)#ip address 10.0.0.2/16
r1(config-if eth0)#no sh
r1(config-if eth0)#exit
r1(config)#interface ethernet eth1
r1(config-if eth1)#ip address 10.22.0.1/16
r1(config-if eth1)#no sh
r1(config-if eth1)#exit
r1(config)#route rip
r1(rip)#network 10.0.0.0/16
r1(rip)#network 10.22.0.0/16
r1(rip)#redistribute connected
r1(rip)#

```

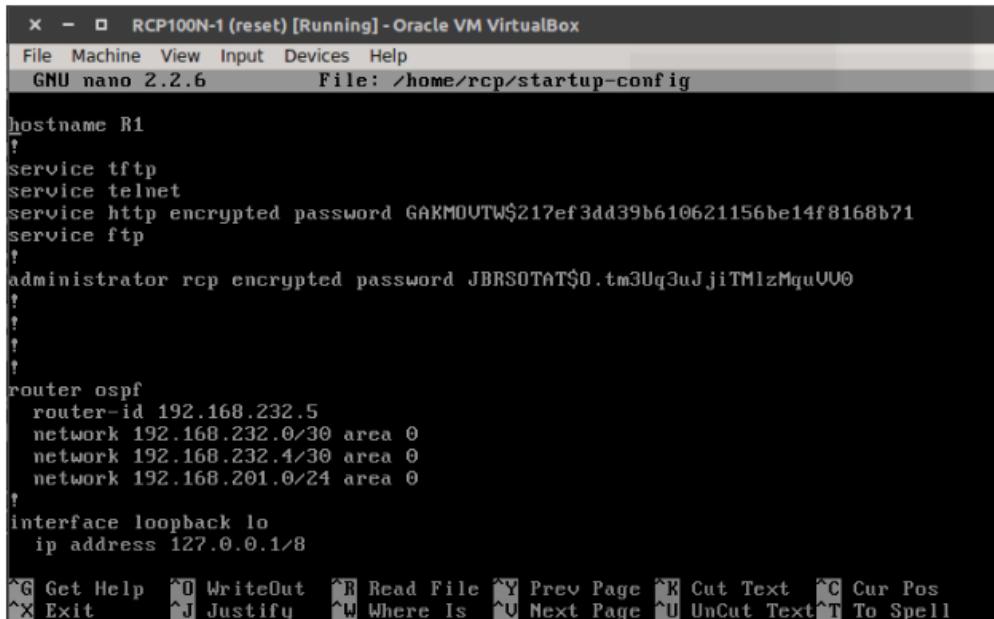
Figura 3.5: Configuración RIP.

## Enrutamiento OSPF

Open Shortest Path First (OSPF), camino más corto primero, es un protocolo de red para encaminamiento jerárquico de pasarela interior o Interior Gateway Protocol (IGP), que usa el algoritmo SmoothWall Dijkstra enlace-estado para calcular la ruta idónea entre dos nodos cualesquiera de un sistema autónomo.

Su medida de métrica se denomina cost, y tiene en cuenta diversos parámetros tales como el ancho de banda y la congestión de los enlaces. OSPF construye además una base de datos enlace-estado (Link-State Database, LSDB) idéntica en todos los routers de la zona.

Una red OSPF se puede descomponer en regiones (áreas) más pequeñas. Hay un área especial llamada área backbone que forma la parte central de la red a la que se encuentran conectadas el resto de áreas de la misma. Las rutas entre las diferentes áreas circularán siempre por el backbone, por lo tanto todas las áreas deben conectar con el backbone. Si no es posible hacer una conexión directa con el backbone, se puede hacer un enlace virtual entre redes [3].



```

x - RCP100N-1 (reset) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 2.2.6      File: /home/rcp/startup-config

hostname R1
!
service tftp
service telnet
service http encrypted password GAKMOUTW$217ef3dd39b610621156be14f8168b71
service ftp
!
administrator rcp encrypted password JBR5OTAT$0.tm3Uq3uJj1TM1zMquUU0
!
!
!
!
!
router ospf
  router-id 192.168.232.5
  network 192.168.232.0/30 area 0
  network 192.168.232.4/30 area 0
  network 192.168.201.0/24 area 0
!
interface loopback 0
  ip address 127.0.0.1/8

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^U Next Page ^U UnCut Text ^T To Spell

```

Figura 3.6: Configuración OSPF.

# CAPÍTULO 4

---

## Conclusiones

---

- **Castro Flores Marcela**

El desarrollo de esta práctica fue un poco difícil debido a las múltiples formas en las cuales se pueden configurar las topologías y de las cuales se obtenían los archivos de configuración de los diferentes routers, mismos que posteriormente, podían ser importados nuevamente a ellos con ciertas modificaciones y funcionalidades. Sin embargo, al realizar la implementación de las diferentes librerías de python tanto para realizar el monitoreo de los diferentes servidores, como también para obtener los archivos de configuración mencionados anteriormente, me di cuenta que era mucho más sencilla y mucho más rápida la obtención de la información como los tiempos de respuesta o bytes transferidos y también se optimizaban muchos de los pasos requeridos para simplemente obtener un archivo.

Creo que la implementación de este tipo de topologías nos acerca a una visión más próxima de como están realmente configuradas y conectadas las grandes redes tanto a nivel local como a un nivel más industrial.

- **Sánchez Cruz Rosa María**

Concluyo que esta práctica fué retadora ya que es una aplicación bastante grande debido a que maneja todo lo que hemos visto en los parciales anteriores y además agrega el manejo de otros protocolos, finalmente integrar todo en GNS3 fue algo retador. Pero a cambio considero que fue importante ver estos temas ya que no nos detenemos a pensar como es que se administran las redes, que protocolos son los que intervienen en su funcionamiento y mucho menos de que forma nosotros como ingenieros en sistemas podemos desarrollar una herramienta la cual nos permita manipular de una forma más sencilla todos estos conceptos. Es importante mencionar que tambien aprendí a configurar enrutadores, resolver problemas de conectividad y a interpretar tablas de enrutamiento. De la misma forma se logró incluir diferentes servicios, obtener información y mostrarla, todo en un mismo gestor.

- **Santiago Mancera Arturo Samuel**

La práctica fue muy interesante porque nos permitió experimentar en GNS3 y enrutamiento. Lo cual son conocimientos básicos que deberían tomarse con más detalle. Así mismo, cabe resaltar que la implementación de servicios fue uno de los puntos más complicados. Sobre todo con respecto al servidor de correo.

Por otra parte, para la implementación de los sensores es importante señalar que se utilizaron soluciones simples como conexiones SSH y FTP. Sólo se tuvieron complicaciones con el sensor SSH para obtener los bytes transferidos por sesión y con el sensor de impresión al no contar con una impresora real.

---

## Referencias y bibliografías

---

- [1] FACULTAD DE ESTADÍSTICA E INFORMÁTICA., *Universidad Veracruzana*, (2018). Disponible en: <https://www.uv.mx/personal/ocruz/files/2014/01/Enrutamiento-estatico.pdf>.
- [2] RIP Cisco, *Aprende a configurar este protocolo facilmente.* (2018). Disponible en: <https://aplicacionesysistemas.com/rip-cisco-version2-de-manera-facil-y-sencilla/>.
- [3] FERNÁNDEZ, R., (2018). *Enrutamiento dinámico OSPF con Packet Tracer.* Disponible en: <https://www.raulprietofernandez.net/blog/packet-tracer/enrutamiento-dinamico-ospf-con-packet-tracer>