

# Práctica 2.1

## Enrutamiento múltiple

### Redes 3

Domingo, 29 de noviembre 2020

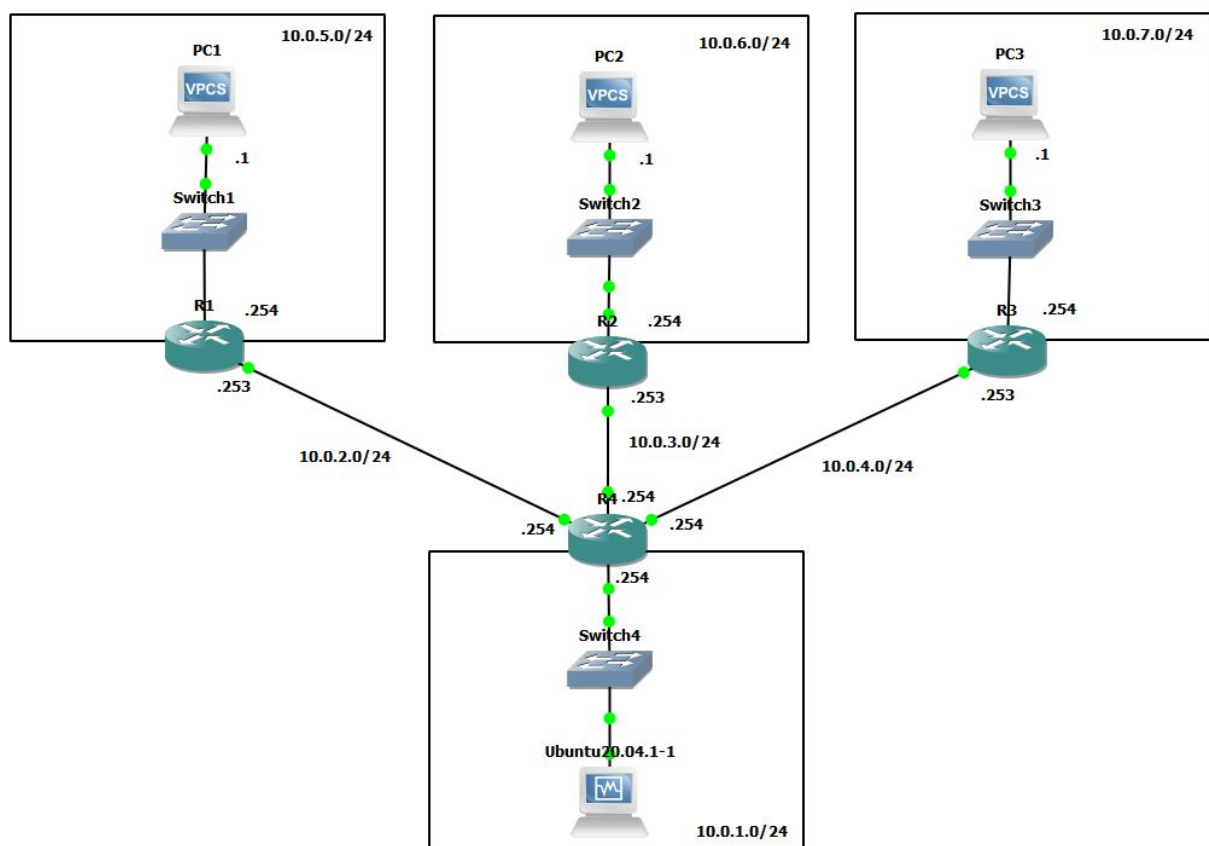
Joel Harim Hernández Javier

### Descripción

Desarrollar un programa en Python que sea capaz de levantar el enrutamiento estático y dinámico de la columna vertebral de una topología con múltiples métodos de enrutamiento.

Implementar en GNS3 la siguiente topología, donde únicamente se van a configurar las interfaces indicadas en enrutadores, MV y VPCS y un usuario admin con contraseña admin en ssh de los enrutadores.

Desarrollar un programa en Python que correrá en la máquina virtual host1 y que sea capaz de levantar los distintos métodos de enrutamiento indicados en la topología y permitir que exista conectividad en toda la red.



El programa no contendrá ninguna dirección IP y deberá de ser capaz de ir localizando los diferentes router (por su nombre), así como levantar los tipos de enrutamiento según la siguiente tabla:

| Enrutador | Estático | RIP | OSPF |
|-----------|----------|-----|------|
| R1        | Sí       |     |      |
| R2        |          | Sí  |      |
| R3        |          |     | Sí   |
| R4        | Sí       | Sí  | Sí   |

## Desarrollo

### 1. Creación de la topología

Se creó la topología indicada y se configuraron solamente las direcciones de cada interfaz de los routers y un usuario ssh; así como las ips de las máquinas virtuales.

Se muestra la tabla de enrutamiento del router R1:

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 2 subnets
C       10.0.2.0 is directly connected, FastEthernet0/1
C       10.0.5.0 is directly connected, FastEthernet0/0
R1#
```

Se muestra la tabla de enrutamiento del router R2:

```
R2#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 2 subnets
C       10.0.3.0 is directly connected, FastEthernet0/1
C       10.0.6.0 is directly connected, FastEthernet0/0
```

Se muestra la tabla de enrutamiento del router R3:

```
R3#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 2 subnets
C       10.0.7.0 is directly connected, FastEthernet0/0
C       10.0.4.0 is directly connected, FastEthernet0/1
R3#
```

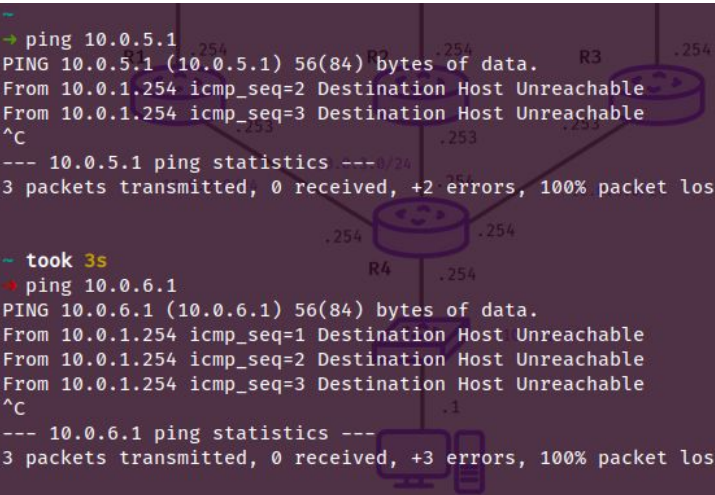
Se muestra la tabla de enrutamiento del router R4:

```
R4#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 4 subnets
C       10.0.2.0 is directly connected, FastEthernet0/0
C       10.0.3.0 is directly connected, FastEthernet0/1
C       10.0.1.0 is directly connected, FastEthernet1/1
C       10.0.4.0 is directly connected, FastEthernet1/0
R4#
```

Intento de conexión con las redes remotas



```
--
-> ping 10.0.5.1
PING 10.0.5.1 (10.0.5.1) 56(84) bytes of data.
From 10.0.1.254 icmp_seq=2 Destination Host Unreachable
From 10.0.1.254 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.5.1 ping statistics ---
3 packets transmitted, 0 received, +2 errors, 100% packet loss, time 2021ms

-- took 3s
-> ping 10.0.6.1
PING 10.0.6.1 (10.0.6.1) 56(84) bytes of data.
From 10.0.1.254 icmp_seq=1 Destination Host Unreachable
From 10.0.1.254 icmp_seq=2 Destination Host Unreachable
From 10.0.1.254 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.6.1 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2004ms

-- took 2s
-> ping 10.0.7.1
PING 10.0.7.1 (10.0.7.1) 56(84) bytes of data.
From 10.0.1.254 icmp_seq=1 Destination Host Unreachable
From 10.0.1.254 icmp_seq=2 Destination Host Unreachable
^C
--- 10.0.7.1 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1002ms
EJECUTAR
```

## 2. Ambiente virtual y estructura de la práctica

En la carpeta de la práctica se creó un ambiente virtual de python, donde se copió el `requirements.txt`:

```
python3 -m venv env
source env/bin/activate
pip -r requirements.txt
```

Después de creó la siguiente estructura de directorios:

```
.
├── app
│   ├── __init__.py
│   ├── static
│   │   ├── css
│   │   ├── fonts
│   │   ├── images
│   │   └── js
│   ├── templates
│   └── views
│       └── __init__.py
├── router_configuration
│   └── __init__.py
└── run.py
```

Es decir, se crearon dos paquetes, `app/` que es el servidor Flask y el `router_configuration/` que es el paquete encargado de realizar la configuración en los routers, mientras que el archivo `run.py` es el archivo principal de la práctica.

### 3. Algoritmo de configuración

El algoritmo de configuración consiste en iniciar sesión ssh en el router 4, el cual es el gateway de la computadora virtual, y desde ahí se inicia sesión ssh en cada uno de los demás routers, es decir, una sesión ssh dentro de otra sesión ssh.

La función principal inicia sesión con el R4, cuya dirección ip encuentra consultando el gateway configurado en la máquina virtual:

```
from router_configuration import router, shared, net
from pexpect import pxssh

from router_configuration import test

def configure():
    shared.username = "admin"
    shared.password = "admin"

    gateway = net.get_default_gateway()

    session = pxssh.pxssh()

    print("Connecting to " + gateway + "...")

    session.login(gateway, shared.username, shared.password,
auto_prompt_reset=False)
    session.sendline("term length 0")
    session.expect("#")

    router.config_r4(session)

    session.logout()
```

La configuración del router 4 se realiza en la función `config_r4`:

```
def config_r4(session):

    me = net.get_hostname(session.before)

    shared.set_name(me)

    connections = net.get_connections(session)

    commands = ["conf t"]
    for route in config_routers(session, connections, me):
        commands.append(
```

```

        "ip route " + route["net"] + " " + route["mask"] + " " +
route["ip"]
    )

    shared.set_name(me)

    configuration.send_commands(session, commands, exits=1)

    configuration.ospf(session, connections)
    configuration.rip(session, connections)

    configuration.redistribuite_ospf(session)
    configuration.redistribuite_rip(session)

```

La configuración de los demás routers se logra consultando la tabla de enrutamiento directa y calculando el siguiente salto, que se supone como última o penúltima dirección ip de la red conectada directamente:

```

def config_routers(session, connections, me):
    routes = []

    for hop in net.get_next_hops(session, connections):
        shared.set_name(me)

        if not net.check_conn(session, hop["hop"]):
            print(shared.name + "Without connection: " + hop["hop"])
            continue

        open_inner_session(session, hop["hop"], shared.username,
shared.password)

        hostname = net.get_hostname(session.before)

        shared.set_name(me, hostname)

        if hostname == "R1":
            routes = config_r1(session, hop["source"])
            for route in routes:
                route["ip"] = hop["hop"]

        elif hostname == "R2":
            config_r2(session)
        else:
            config_r3(session)

        close_inner_session(session)

```

```
return routes
```

Enrutamiento del router R1:

```
def config_r1(session, source):  
  
    commands = ["conf t", "ip route 0.0.0.0 0.0.0.0 " + source]  
  
    configuration.send_commands(session, commands, exits=1)  
  
    connections = net.get_connections(session)  
  
    return [c for c in connections if net.get_next_hop(c) != source]
```

Enrutamiento del router R2:

```
def config_r2(session):  
    connections = net.get_connections(session)  
    configuration.rip(session, connections)
```

Enrutamiento del router R4:

```
def config_r3(session):  
    connections = net.get_connections(session)  
    configuration.ospf(session, connections)
```

## 4. Configuración del servidor de Flask

Se configuró las siguientes rutas para el frontend en `app/views/configure_routing.py`:

```
import time, json  
  
from flask import render_template, request, jsonify, make_response  
from app import app  
from router_configuration import configure  
  
@app.route("/configure_routing")  
def configure_routing():  
    return render_template("configure_routing.html")
```



```
@app.route("/configure_routing/configure", methods=["POST"])
def configuring():
    req = request.get_json()

    print(req)

    try:
        configure()
        return make_response(jsonify(req), 200)
    except Exception as e:
        obj = {"message": str(e)}
        return make_response(json.dumps(obj), 500)
```

## 5. FrontEnd

La página dedicada a la práctica tiene un botón que invoca al evento remoto:

joel@ubuntuvm:/configure\_routing

### Networking management

- Menu**
- Index
  - Configure routing

## Configure routing

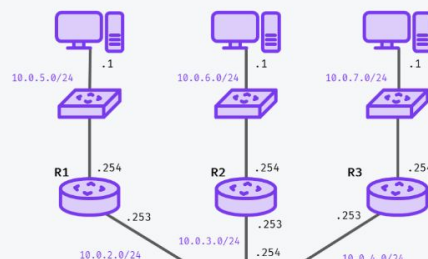
### Descripción

Este programa escrito en python se encarga de levantar el enrutamiento en 4 routers que lo único que tienen configurado son sus interfaces.

Se tienen que configurar de la siguiente manera los routers:

- R1: Enrutamiento estático
- R2: Enrutamiento usando RIP
- R3: Enrutamiento usando OSPF
- R4: Enrutamiento múltiple que permita la interoperabilidad de los otros 3 routers.

### Topología



### Ejecución

Este botón ejecuta el script de forma remota

EJECUTAR



## 5. Ejecución

El servidor es ejecutado:

```
network_management/practices/practice_2.1/server on ↵ main
→ flask run --host=0.0.0.0
* Serving Flask app "run.py" (lazy loading)
* Environment: development
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 522-736-198
```

Se ingresa a `localhost:5000`, la ejecución se observa como sigue:

### Ejecución

Este botón ejecuta el script de forma remota

CARGANDO... 🐞

En la consola se observa como Flask ejecuta el script:

```
Connecting to 10.0.1.254...
[R4] sh run | inc interface | ip address
[R4] ping 10.0.2.253 r 3
[R4] [R1] conf t
[R4] [R1] ip route 0.0.0.0 0.0.0.0 10.0.2.254
[R4] [R1] exit
[R4] [R1] sh run | inc interface | ip address
[R4] ping 10.0.3.253 r 3
[R4] [R2] sh run | inc interface | ip address
[R4] [R2] conf t
[R4] [R2] router rip
[R4] [R2] version 2
[R4] [R2] no auto
[R4] [R2] net 10.0.6.0
[R4] [R2] net 10.0.3.0
[R4] [R2] exit
[R4] [R2] exit
[R4] ping 10.0.4.253 r 3
[R4] [R3] sh run | inc interface | ip address
[R4] [R3] conf t
[R4] [R3] router ospf 1
[R4] [R3] net 10.0.7.0 0.0.0.255 area 0
[R4] [R3] net 10.0.4.0 0.0.0.255 area 0
[R4] [R3] exit
[R4] [R3] exit
[R4] ping 10.0.1.253 r 3
[R4] Without connection: 10.0.1.253
[R4] conf t
[R4] ip route 10.0.5.0 255.255.255.0 10.0.2.253
[R4] exit
[R4] conf t
[R4] router ospf 1
[R4] net 10.0.2.0 0.0.0.255 area 0
[R4] net 10.0.3.0 0.0.0.255 area 0
[R4] net 10.0.4.0 0.0.0.255 area 0
[R4] net 10.0.1.0 0.0.0.255 area 0
```

Finalmente se muestra el siguiente mensaje:

## Ejecución

Este botón ejecuta el script de forma remota

**Routers configurados!**

**EJECUTAR**

## 7. Verificación de conexión

Ping a las máquinas remotas:

```
~  
→ ping 10.0.5.1  
PING 10.0.5.1 (10.0.5.1) 56(84) bytes of data.  
64 bytes from 10.0.5.1: icmp_seq=2 ttl=62 time=25.5 ms  
64 bytes from 10.0.5.1: icmp_seq=3 ttl=62 time=32.4 ms  
^C  
--- 10.0.5.1 ping statistics ---  
3 packets transmitted, 2 received, 33.3333% packet loss, time 2011ms  
rtt min/avg/max/mdev = 25.500/28.944/32.389/3.444 ms  
  
~ took 3s  
→ ping 10.0.6.1  
PING 10.0.6.1 (10.0.6.1) 56(84) bytes of data.  
64 bytes from 10.0.6.1: icmp_seq=2 ttl=62 time=40.2 ms  
64 bytes from 10.0.6.1: icmp_seq=3 ttl=62 time=34.3 ms  
^C  
--- 10.0.6.1 ping statistics ---  
3 packets transmitted, 2 received, 33.3333% packet loss, time 2033ms  
rtt min/avg/max/mdev = 34.337/37.280/40.224/2.943 ms  
  
~ took 3s  
→ ping 10.0.7.1  
PING 10.0.7.1 (10.0.7.1) 56(84) bytes of data.  
64 bytes from 10.0.7.1: icmp_seq=2 ttl=62 time=30.4 ms  
64 bytes from 10.0.7.1: icmp_seq=3 ttl=62 time=26.5 ms  
^C  
--- 10.0.7.1 ping statistics ---  
3 packets transmitted, 2 received, 33.3333% packet loss, time 2029ms  
rtt min/avg/max/mdev = 26.542/28.489/30.437/1.947 ms  
  
~ took 3s  
→
```

Con esto podemos comprobar que la conexión y la configuración de los routers fue completamente exitosa.

El código fuente completo se puede encontrar en mi github:

<https://github.com/JoelHernandez343/networking-administration>