



PRACTICA 2

TOPOLOGYZOO

PEDRO MIGUEL CARMONA BRONCANO

Trabajo a realizar.....	2
Estructura de ficheros de la aplicación.....	2
Recogida de datos.....	3
Cálculo de datos necesarios.....	4
Peso de las interfaces en OSPF.....	4
Direcciones IP de las redes e interfaces.....	5
Generación de ficheros para la aplicación Kathara.....	5
Pruebas realizadas.....	7
Conectividad total.....	7
Forzando a calcular otras rutas.....	7

Trabajo a realizar

En esta segunda práctica se seleccionará una topología de red de las librerías [SNDLib](#) o [TopologyZoo](#) y se creará un script que cree un laboratorio de Kathara de manera automática. Tras su ejecución, se realizará documentación de las diversas pruebas que permitan comprobar el funcionamiento de los protocolos RIP OSPF estudiados en clase. Se deben describir cada uno de los pasos realizados en una memoria técnica, además de entregar los archivos y los directorios de los laboratorios.

Estructura de ficheros de la aplicación

En este apartado se va a proceder a explicar la estructura en la que se ha establecido la aplicación.

- **classes** (Clases utilizadas para el proyecto)
- **files** (Contiene las plantillas para el enrutamiento en RIP o OSPF)
- **generate** (Contendrá una carpeta por cada topología generada)
 - **<Nombre de la topología>** (Archivos generados para la topología elegida)
 - **<Nombre de la topología>_topology.txt** (Archivo de texto que ilustra las conexiones entre los nodos)
- **parser** (Parser utilizado para pasar la topología)
- **topology** (Ficheros de las topologías)
- **main.py** (Archivo principal de ejecución)

Recogida de datos

```
TPC (Workspace) - main.py

def parse_command_line_args(args):
    if len(args) != 5:
        print("Use: python main.py <dataset> <format> <routing> <path_file>")
        print()
        print("Argumentos validos:")
        print("Dataset -> [topologyzoo]")
        print("Format -> [gml]")
        print("Routing -> [rip ospf]")
        sys.exit(1)

    dataset = args[1]
    file_format = args[2]
    routing = args[3]
    file_path = args[4]

    if dataset == "topologyzoo":
        if file_format == "gml":
            print(f"Procesando TopologyZoo dataset en formato {file_format} desde {file_path}.")
            parser = TopologyZooGMLParser(file_path)
            nodes = parser.parse_nodes()
            edges = parser.parse_edges()
        else:
            print("Formato no válido para TopologyZoo. Use 'gml'.")
            sys.exit(1)

        if routing != "rip" and routing != "ospf" :
            print("Formato no válido para Routing. Use 'rip' o 'ospf'.")
            sys.exit(1)
    else:
        print("Conjunto de datos no válido. Use 'topologyzoo'.")
        sys.exit(1)

    lab_name = file_route(file_path)

    return nodes, edges, routing, lab_name
```

Entre las dos librerías disponibles para esta práctica se ha optado por elegir *TopologyZoo* debido a su mayor número de topologías disponibles y por tener formato *GML*, el cual es más comprensible por el ojo humano y más fácil de construir el *parser* de datos. No obstante, se ha dejado preparado el programa realizado para una futura ampliación de más librerías y formatos.

El primer paso que hace el programa es recoger en la entrada lo que queremos:

- El dataset elegido
- El formato de entrada de los datos
- El protocolo de enrutamiento que queremos para la topología
- La ruta del fichero que contiene la topología

Como se indicaba antes, ahora mismo el programa soporta solo los siguientes datos:

- Dataset [topologyzoo]
- Formato [gml]
- Protocolo de enrutamiento [rip ospf]

Para arrancar la aplicación deberemos por tanto introducir el siguiente comando.

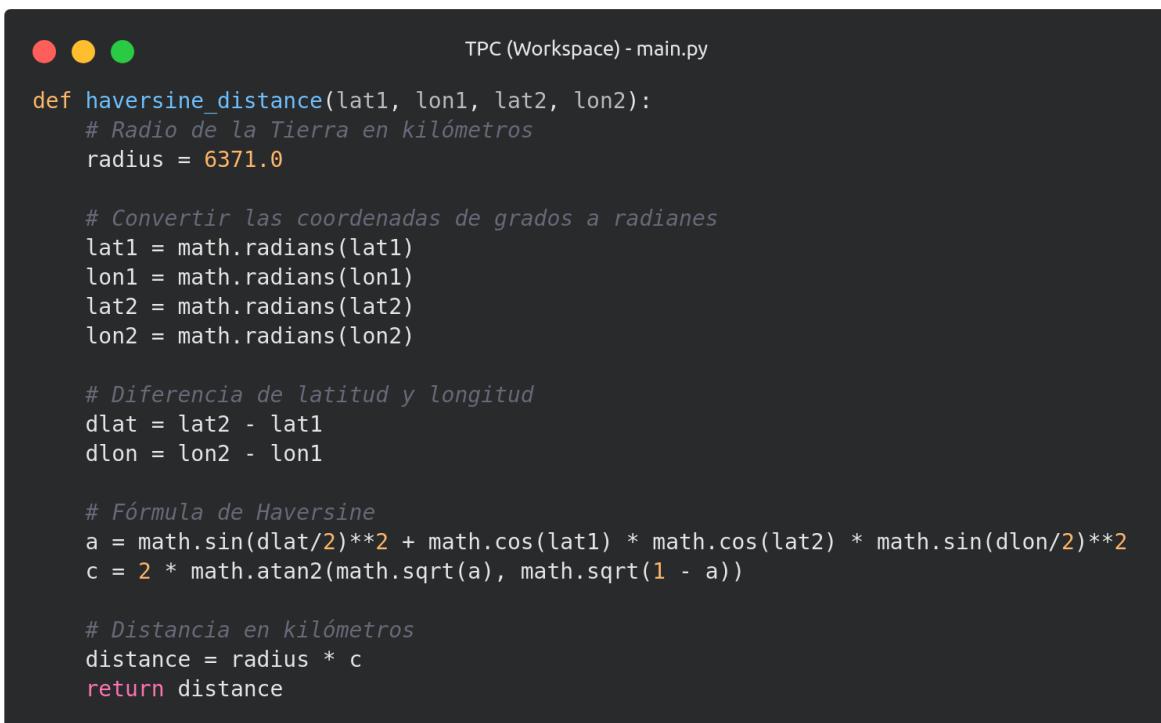
python3 main.py topologyzoo gml <Protocolo de enrutamiento> <Ruta de topología>

Para más información leer *README.md*.

Cálculo de datos necesarios

Peso de las interfaces en OSPF

Si el protocolo de OSPF es elegido, para calcular el peso de las interfaces se ha optado por aprovechar los datos de *Latitude* y *Longitude* de cada router para calcular la distancia en km que hay entre dos de ellos. Esta distancia se calcula a través del siguiente método.



```

TPC (Workspace) - main.py

def haversine_distance(lat1, lon1, lat2, lon2):
    # Radio de la Tierra en kilómetros
    radius = 6371.0

    # Convertir las coordenadas de grados a radianes
    lat1 = math.radians(lat1)
    lon1 = math.radians(lon1)
    lat2 = math.radians(lat2)
    lon2 = math.radians(lon2)

    # Diferencia de latitud y longitud
    dlat = lat2 - lat1
    dlon = lon2 - lon1

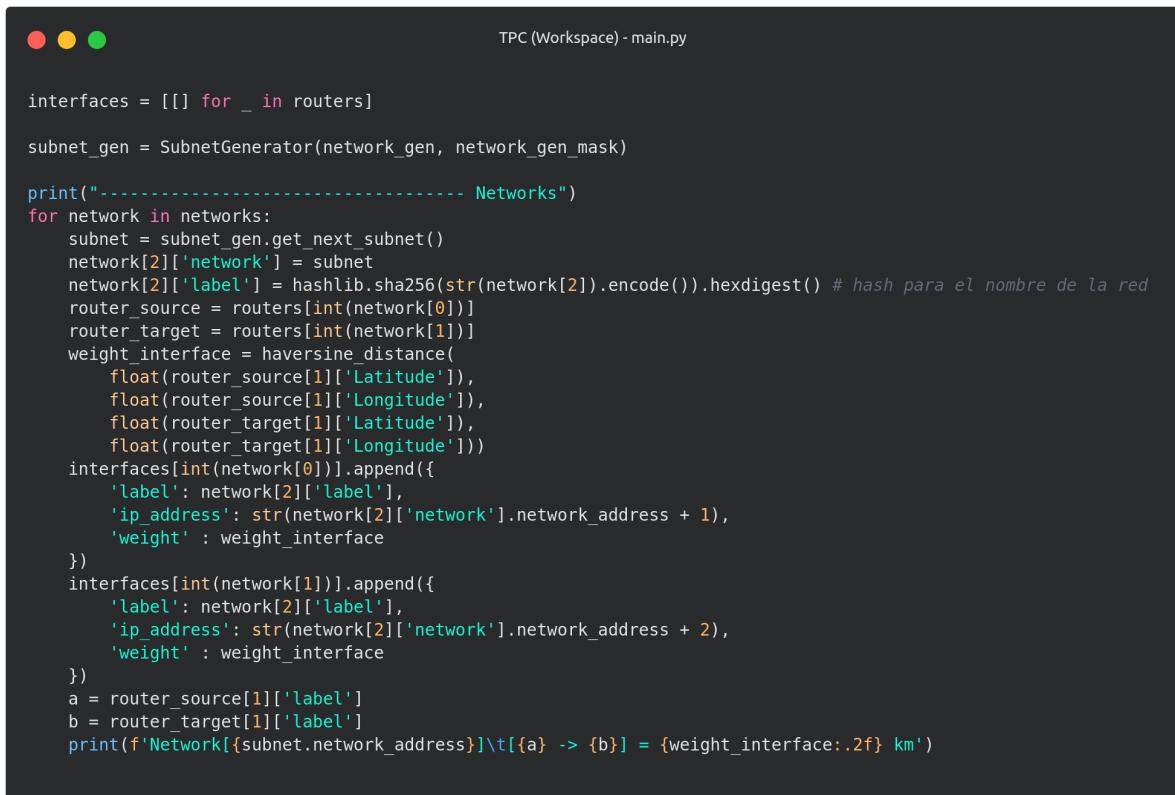
    # Fórmula de Haversine
    a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(dlon/2)**2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))

    # Distancia en kilómetros
    distance = radius * c
    return distance

```

Direcciones IP de las redes e interfaces

Para calcular las direcciones IP que va a tener cada red/interfaz se ha optado por tener una clase encargada de ello (*SubnetGenerator*) en la que cada vez que se lo pide te da la siguiente red. Así podemos recorrer la lista de *networks* e ir calculando las interfaces que va a tener cada router, que dirección IP le pertenece y qué peso tendrá en el protocolo de enrutamiento *OSPF*. Todos estos datos se guardan en una estructura auxiliar llamada *interfaces*.



```
TPC (Workspace) - main.py

interfaces = [] for _ in routers]

subnet_gen = SubnetGenerator(network_gen, network_gen_mask)

print("----- Networks")
for network in networks:
    subnet = subnet_gen.get_next_subnet()
    network[2]['network'] = subnet
    network[2]['label'] = hashlib.sha256(str(network[2]).encode()).hexdigest() # hash para el nombre de la red
    router_source = routers[int(network[0])]
    router_target = routers[int(network[1])]
    weight_interface = haversine_distance(
        float(router_source[1]['Latitude']),
        float(router_source[1]['Longitude']),
        float(router_target[1]['Latitude']),
        float(router_target[1]['Longitude']))
    interfaces[int(network[0])].append({
        'label': network[2]['label'],
        'ip_address': str(network[2]['network'].network_address + 1),
        'weight' : weight_interface
    })
    interfaces[int(network[1])].append({
        'label': network[2]['label'],
        'ip_address': str(network[2]['network'].network_address + 2),
        'weight' : weight_interface
    })
    a = router_source[1]['label']
    b = router_target[1]['label']
    print(f'Network{subnet.network_address}\t[a] -> {b} = {weight_interface:.2f} km')
```

Generación de ficheros para la aplicación

Kathara

Una vez que se tienen los datos necesarios calculados, es hora de generar los ficheros necesarios para que la topología pueda funcionar.

```

TPC (Workspace) - main.py

print("----- Routers")
caracteres_replace=[ " ", "(, )", ":" , "," , ".", "ñ", "?", "¿", "!", "¡"]
lab = open(path_lab + '/lab.conf', 'w')
for router in routers:
    machine_name = router[1]['label'].lower()
    print(machine_name)
    for caracter in caracteres_replace:
        machine_name = machine_name.replace(caracter, "_")
    router[1]['label'] = machine_name
id_machine = int(router[0])
interface_index = 0
commands_ospf = []
startup = open(path_lab + "/" + machine_name + '.startup', 'w')
for interface_router in interfaces[id_machine]:
    command = f"ifconfig eth{interface_index} {interface_router['ip_address']}/{network_gen_mask} up\n"
    startup.write(command)
    interface_label = interface_router['label']
    lab_command = f"{machine_name}[{interface_index}]{interface_label}\n"
    lab.write(lab_command)
    weight = int(interface_router['weight']) if int(interface_router['weight']) > 0 else 1
    command_ospf = f"interface eth{interface_index}\nospf cost {weight}\n"
    commands_ospf.append(command_ospf)
    interface_index += 1
    startup.write("\n/etc/init.d/quagga start") # add start quagga
    startup.close()

    route_machine = path_lab + "/" + machine_name

    if routing == "rip":
        copy_folder("files/rip", route_machine)
        with open(route_machine + "/" + file_rip, 'r+') as file:
            content = file.read()
            content_replace = content.replace(network_text, network_gen)
            file.seek(0)
            file.write(content_replace)
    if routing == "ospf":
        copy_folder("files/ospf", route_machine)
        with open(route_machine + "/" + file_ospfd, 'r+') as file:
            content = file.read()
            content_replace = content.replace(change_text, '\n'.join(commands_ospf))
            content_replace = content_replace.replace(network_text, network_gen)
            file.seek(0)
            file.write(content_replace)

create_test_machine(path_test + "/" + machine_name + ".test", interfaces, id_machine)

lab.close()

```

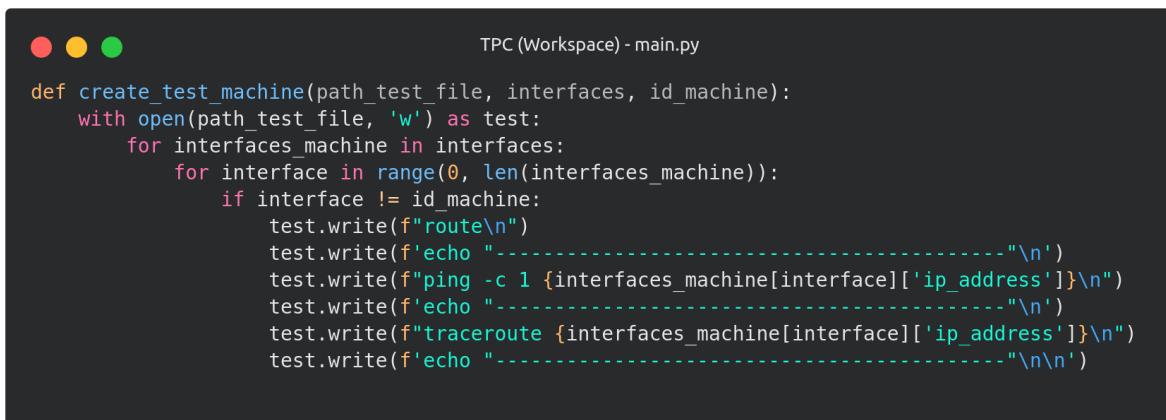
- El primer paso se establece un reemplazo de los caracteres especiales que pueda contener el nombre de los routers de la topología.
- Acto seguido, por cada interfaz, que previamente hemos guardado en *interfaces*, que le corresponda al router se va escribiendo en su archivo *.startup*. En este paso también se van a guardar los costes de las interfaces para que, si se ha elegido el protocolo de enrutamiento *OSPF*, poder escribirlo posteriormente en su fichero correspondiente.
- Al finalizar el bucle de todas las interfaces del router, se escribe también el comando para poder inicializar *quagga* cuando se inicie el router.

- En el siguiente paso, se comprueba que protocolo de enrutamiento se ha elegido para poder configurar en el router los ficheros de configuración pertinentes.
- Por último, se genera el archivo de *testing* del router en cuestión.

Pruebas realizadas

Conecividad total

La primera prueba que se ha realizado es la de conectividad total, para ello se genera, para cada router, un archivo de pruebas en el que se comprueba que cada router puede hacer *ping* con todos los demás y además se comprueba la ruta que hacen con el comando *traceroute*. Los resultados de los test se pueden comprobar en el repositorio [Kathara Github](#)



The screenshot shows a terminal window with a dark background and light-colored text. The title bar says "TPC (Workspace) - main.py". The code in the terminal is as follows:

```
def create_test_machine(path_test_file, interfaces, id_machine):
    with open(path_test_file, 'w') as test:
        for interfaces_machine in interfaces:
            for interface in range(0, len(interfaces_machine)):
                if interface != id_machine:
                    test.write(f"route\n")
                    test.write(f"echo -----"\n")
                    test.write(f"ping -c 1 {interfaces_machine[interface]['ip_address']} \n")
                    test.write(f"echo -----"\n")
                    test.write(f"traceroute {interfaces_machine[interface]['ip_address']} \n")
                    test.write(f"echo -----"\n\n")
```

Forzando a calcular otras rutas

Para el caso de estudio se ha elegido la topología *RedIris*. Se puede observar dentro de la carpeta */generate/RedIris/* en el archivo *RedIris_topology.txt* que el nodo *Extremadura* tiene dos interfaces que va a los nodos *Nacional* y *Andalucía*.

En primer lugar si se quiere ver la conectividad entre *extremadura* y *andalucía*.

1. Vemos la lista de rutas de *extremadura*
2. *Hacemos un traceroute hacia el host andalucia (dirección ip 10.0.0.90)*. Se ve que están directamente conectados al ser una ruta añadida manualmente y que el comando *traceroute* nos devuelva que solo hay un salto. (primera imagen)
3. Una vez se ha comprobado que hay conexión se ha procedido a tirar el enlace directo que une los dos host con *ifconfig eth1* en el host *extremadura*, esto para ver que el enrutamiento dinámico OSPF funciona correctamente.
4. Realizamos en comando *traceroute* hacia la misma interfaz y esta vez hay dos saltos, ahora para ir de *extremadura* a *andalucía* *hay que pasar por nacional*. (segunda imagen)

```
root@extremadura:/# route
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref  Use Iface
0.0.0.0         10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.4         10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.8         10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.12        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.16        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.20        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.24        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.28        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.32        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.36        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.40        10.0.0.90   255.255.255.252 UG        20      0    0 eth1
0.0.0.44        10.0.0.90   255.255.255.252 UG        20      0    0 eth1
0.0.0.48        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.52        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.56        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.60        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.64        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.68        10.0.0.90   255.255.255.252 UG        20      0    0 eth1
0.0.0.72        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.76        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.80        10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.84        0.0.0.0     255.255.255.252 U        0      0    0 eth0
0.0.0.88        0.0.0.0     255.255.255.252 U        0      0    0 eth1
0.0.0.92        10.0.0.90   255.255.255.252 UG        20      0    0 eth1
0.0.0.96        10.0.0.90   255.255.255.252 UG        20      0    0 eth1
0.0.0.100       10.0.0.90   255.255.255.252 UG        20      0    0 eth1
0.0.0.104       10.0.0.90   255.255.255.252 UG        20      0    0 eth1
0.0.0.108       10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.112       10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.116       10.0.0.86   255.255.255.252 UG        20      0    0 eth0
0.0.0.120       10.0.0.86   255.255.255.252 UG        20      0    0 eth0
root@extremadura:/# traceroute 10.0.0.90
traceroute to 10.0.0.90 (10.0.0.90), 30 hops max, 60 byte packets
 1  10.0.0.90 (10.0.0.90)  0.590 ms  1.499 ms  2.184 ms
root@extremadura:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.85 netmask 255.255.255.252 broadcast 10.0.0.87
    ether 02:19:ab:be:bct:15 txqueuelen 1000 (Ethernet)
      RX packets 120 bytes 13400 (13.0 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 118 bytes 11696 (11.4 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.89 netmask 255.255.255.252 broadcast 10.0.0.91
    ether fe:ec:c0:a8:0b:df txqueuelen 1000 (Ethernet)
      RX packets 177 bytes 18850 (18.4 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 165 bytes 15730 (15.3 KiB)

root@extremadura:/# --- Startup Commands Log
++ ifconfig eth0 10.0.0.46/30 up
++ ifconfig eth1 10.0.0.70/30 up
++ ifconfig eth2 10.0.0.90/30 up
++ ifconfig eth3 10.0.0.93/30 up
++ ifconfig eth4 10.0.0.97/30 up
++ ifconfig eth5 10.0.0.101/30 up
++ /etc/init.d/quagga start
Starting Quagga daemons (priorities: zebra ospfd).
Starting Quagga monitor daemon: watchquagga.
--- End Startup Commands Log
root@andalucia:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.46 netmask 255.255.255.252 broadcast 10.0.0.47
    ether ea:2f:f8:5d:cd:e4 txqueuelen 1000 (Ethernet)
      RX packets 75 bytes 10370 (10.1 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 69 bytes 8546 (8.3 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.70 netmask 255.255.255.252 broadcast 10.0.0.71
    ether 36:dc:02:59:6e:5c txqueuelen 1000 (Ethernet)
      RX packets 59 bytes 8330 (8.1 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 71 bytes 9190 (8.9 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.90 netmask 255.255.255.252 broadcast 10.0.0.91
    ether faafe:f9:f0:ff:3d txqueuelen 1000 (Ethernet)
      RX packets 57 bytes 7398 (7.2 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 73 bytes 9710 (9.4 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.93 netmask 255.255.255.252 broadcast 10.0.0.95
    ether 12:6d:7f:d5:4d:8f txqueuelen 1000 (Ethernet)
      RX packets 38 bytes 3516 (3.4 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 43 bytes 5398 (5.2 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.97 netmask 255.255.255.252 broadcast 10.0.0.99
    ether b6:fd:77:2e:92:5c txqueuelen 1000 (Ethernet)
      RX packets 43 bytes 4806 (4.6 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 50 bytes 6476 (6.3 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```

root@extremadura:/# ifconfig eth1 down
root@extremadura:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.85 netmask 255.255.255.252 broadcast 10.0.0.87
    ether 02:19:ab:be:b1:15 txqueuelen 1000 (Ethernet)
      RX packets 132 bytes 14370 (14.0 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 129 bytes 12718 (12.4 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
  inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
      RX packets 40 bytes 3280 (3.2 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 40 bytes 3280 (3.2 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@extremadura:/# traceroute 10.0.0.90
traceroute to 10.0.0.90 (10.0.0.90), 30 hops max, 60 byte packets
 1  10.0.0.86 (10.0.0.86)  1.149 ms  1.749 ms  2.382 ms
 2  10.0.0.90 (10.0.0.90)  6.074 ms  7.747 ms  9.747 ms
root@extremadura:/# []

root@nacional:/#
root@nacional:/# ifconfig
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.50 netmask 255.255.255.252 broadcast 10.0.0.51
    ether 86:2e:1d:cc:1d:2b txqueuelen 1000 (Ethernet)
      RX packets 141 bytes 15130 (14.7 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 131 bytes 12626 (12.3 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.50 netmask 255.255.255.252 broadcast 10.0.0.51
    ether 86:2e:1d:cc:1d:2b txqueuelen 1000 (Ethernet)
      RX packets 148 bytes 14672 (14.3 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 142 bytes 14996 (14.6 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.58 netmask 255.255.255.252 broadcast 10.0.0.59
    ether 42:e5:d5:e9:89:11 txqueuelen 1000 (Ethernet)
      RX packets 145 bytes 15606 (15.2 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 148 bytes 16444 (16.0 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.66 netmask 255.255.255.252 broadcast 10.0.0.67
    ether 6a:61:69:0d:d2:88 txqueuelen 1000 (Ethernet)
      RX packets 140 bytes 14656 (14.3 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 149 bytes 16298 (15.9 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.74 netmask 255.255.255.252 broadcast 10.0.0.75
    ether 36:2e:03:87:28:a8 txqueuelen 1000 (Ethernet)
      RX packets 136 bytes 13092 (12.7 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 145 bytes 15478 (15.1 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.86 netmask 255.255.255.252 broadcast 10.0.0.87
    ether 6e:5b:7c:8e:27:14 txqueuelen 1000 (Ethernet)
      RX packets 157 bytes 14982 (14.6 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 163 bytes 17282 (16.8 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.102 netmask 255.255.255.252 broadcast 10.0.0.103
    ether a6:5f:d2:d9:d7:7e txqueuelen 1000 (Ethernet)
      RX packets 160 bytes 16136 (15.7 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 162 bytes 16488 (16.1 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@nacional:/# []

root@andalucia:/#
root@andalucia:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.46 netmask 255.255.255.252 broadcast 10.0.0.47
    ether ea:2f:f9:5d:c4:44 txqueuelen 1000 (Ethernet)
      RX packets 75 bytes 10370 (10.1 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 69 bytes 8546 (8.3 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.70 netmask 255.255.255.252 broadcast 10.0.0.71
    ether 36:dc:02:58:6e:5c txqueuelen 1000 (Ethernet)
      RX packets 59 bytes 8330 (8.1 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 71 bytes 9190 (8.9 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.90 netmask 255.255.255.252 broadcast 10.0.0.91
    ether f4:ee:f9:f0:ff:3d txqueuelen 1000 (Ethernet)
      RX packets 57 bytes 7398 (7.2 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 73 bytes 9710 (9.4 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.93 netmask 255.255.255.252 broadcast 10.0.0.95
    ether 12:5d:7f:d5:4d:0f txqueuelen 1000 (Ethernet)
      RX packets 38 bytes 3516 (3.4 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 43 bytes 5398 (5.2 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.0.0.97 netmask 255.255.255.252 broadcast 10.0.0.99
    ether b6:fd:77:2e:92:5c txqueuelen 1000 (Ethernet)
      RX packets 43 bytes 4806 (4.6 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 50 bytes 6476 (6.3 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@andalucia:/# []

```