Linux IP routing

Linux Networking

Serhii Zakharchenko

TRAINING CENTER

Agenda

- Routing basics
- Linux IP routing configuration
- Linux IP routing testing

Routing basics

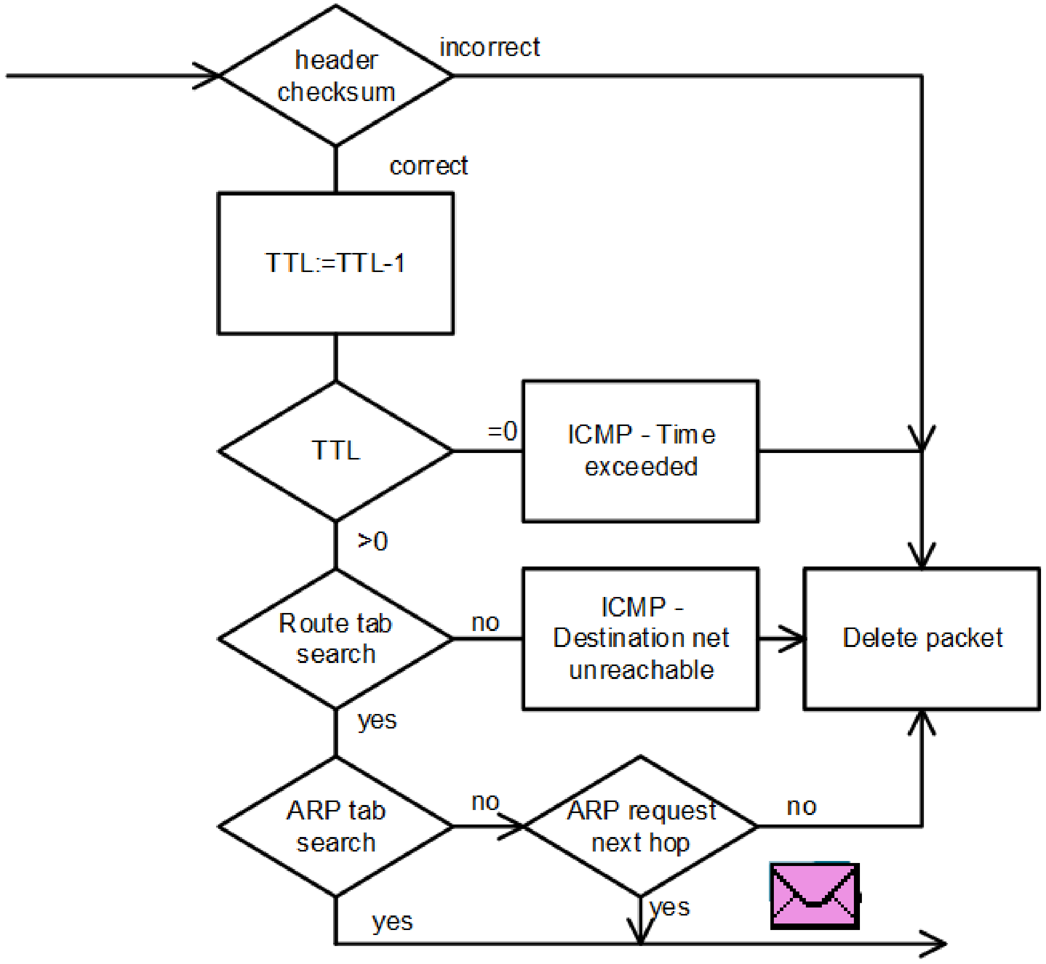
Router functions:

- Router has multiple interfaces and receives data packets through them.
- Router evaluates the network addresses of the incoming packets and decides which interface to forward the packet to.
- Router uses its local routing table for decision-making.
- Routing table can be statically configured or calculated via dynamic routing protocols such as OSPF or BGP.

Routing principles:

1. Every router makes its decision alone, based on the information it has in its own routing table.
2. The fact that one router has certain information in its routing table does not mean that other routers have the same information
3. Routing information about a path from one network to another does not provide routing information about the reverse or return path.

The stages of packet processing and analysis



Static versus Dynamic Routing

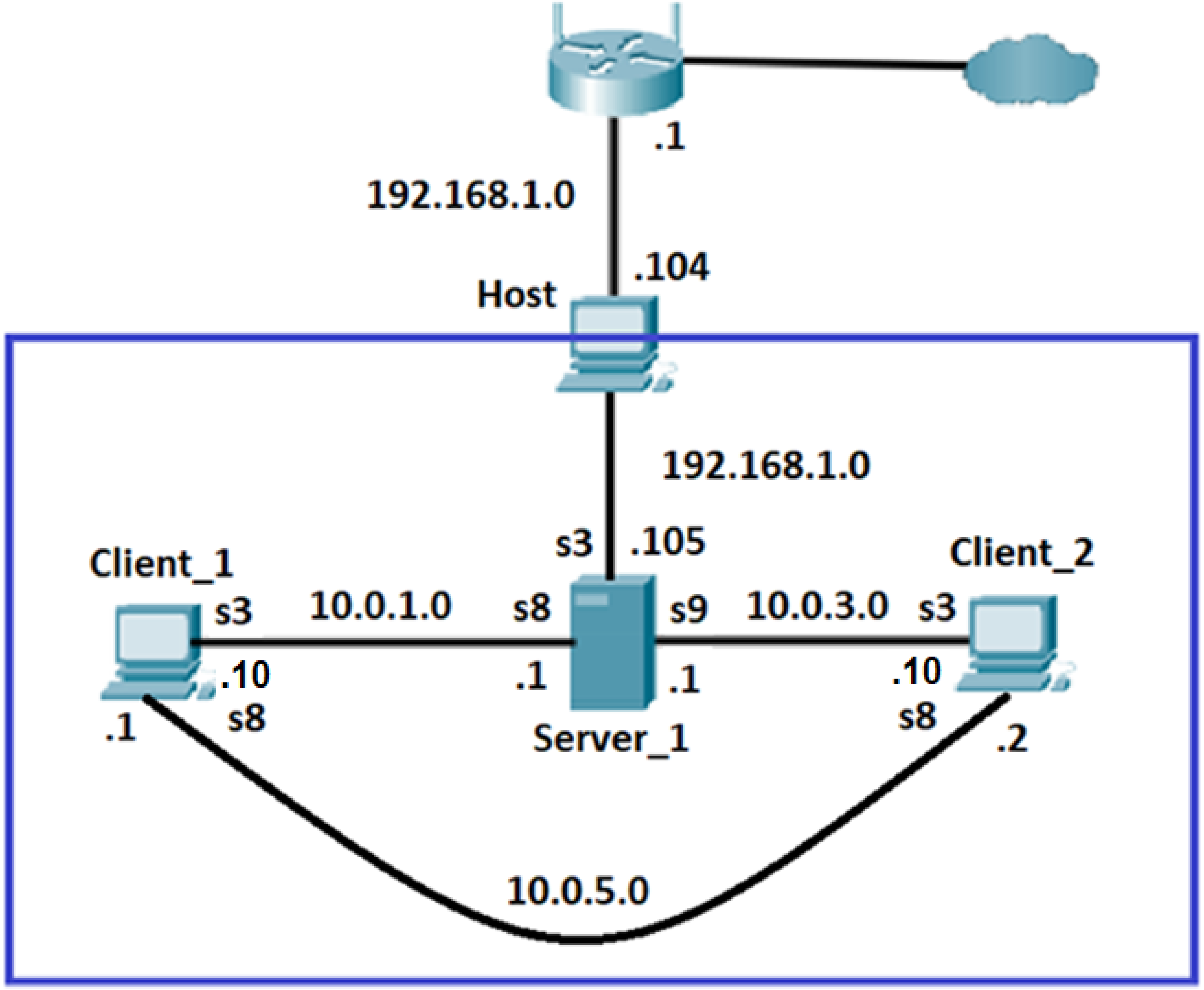
Static routing:

- Providing ease of routing table maintenance in **smaller networks** that are not expected to grow significantly.
- Routing to and from **stub networks**. A stub network is a network accessed by a single route, and the router has no other neighbors.
- Using a **single default route** to represent a path to any network that does not have a more specific match with another route in the routing table.

Dynamic routing:

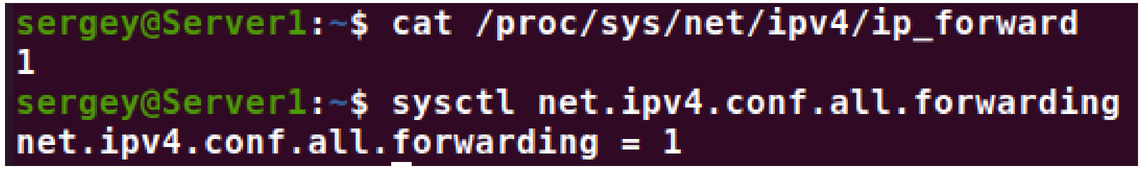
- Exchange of routing information between routers.
- Automatic update of the routing table when changing the route.
- Determining the best path to the destination.
- Allows as many routes as possible to remain valid in response to the change.

Logical Topology example

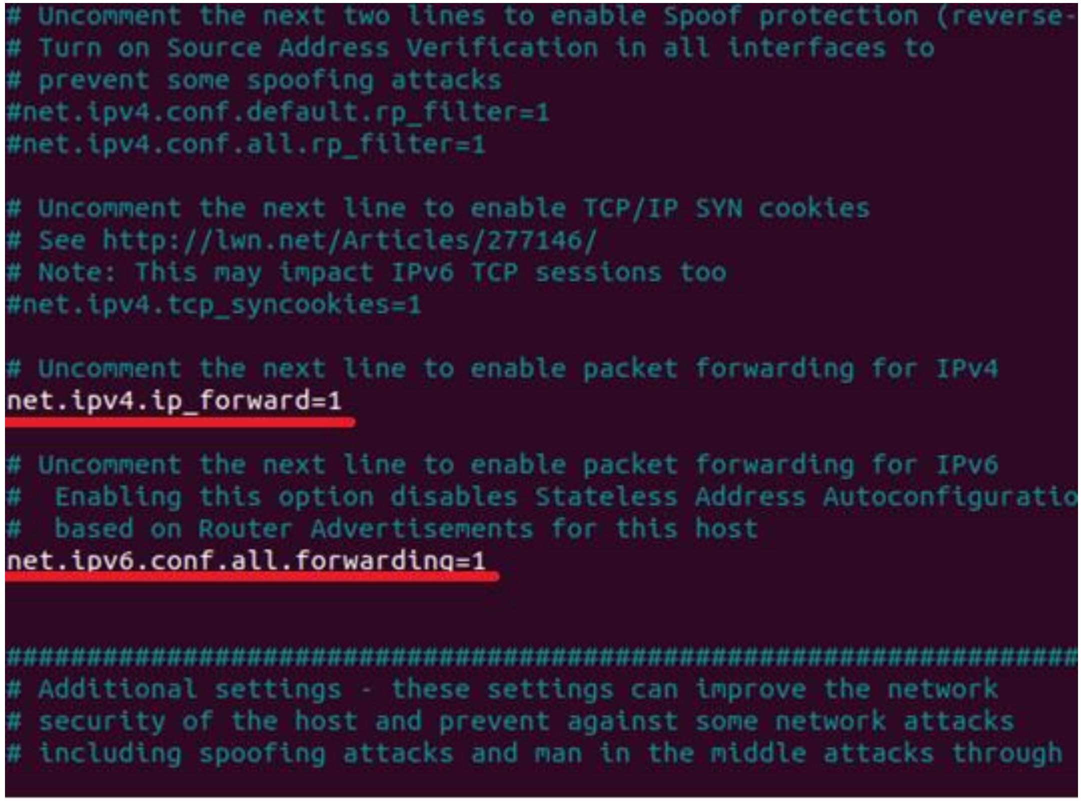


Linux Routing switch on

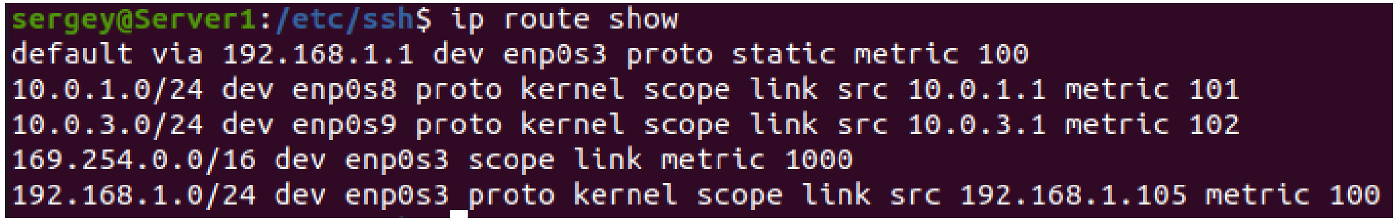
- Routing is switched on in Linux Server, but it is switched off in Linux Workstation.
- Switch on routing is needed only on **transit** devices.
- To check out routing enable use *sysctl net.ipv4. conf.all.forwarding* command
- To switch “on” or “off” routing you must edit /etc/sysctl.conf file
- To review routing table: *$ip route show*



```
sergey@Server1:~$ cat /proc/sys/net/ipv4/ip_forward
1
sergey@Server1:~$ sysctl net.ipv4.conf.all.forwarding
net.ipv4.conf.all.forwarding = 1
```

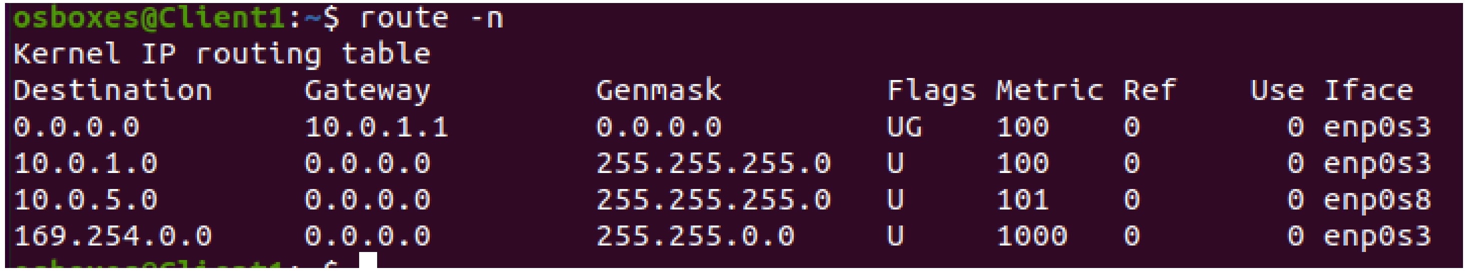


```
# Uncomment the next two lines to enable Spoof protection (reverse-
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
#  Enabling this option disables Stateless Address Autoconfiguratio
#  based on Router Advertisements for this host
net.ipv6.conf.all.forwarding=1


###################################################################
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through
```

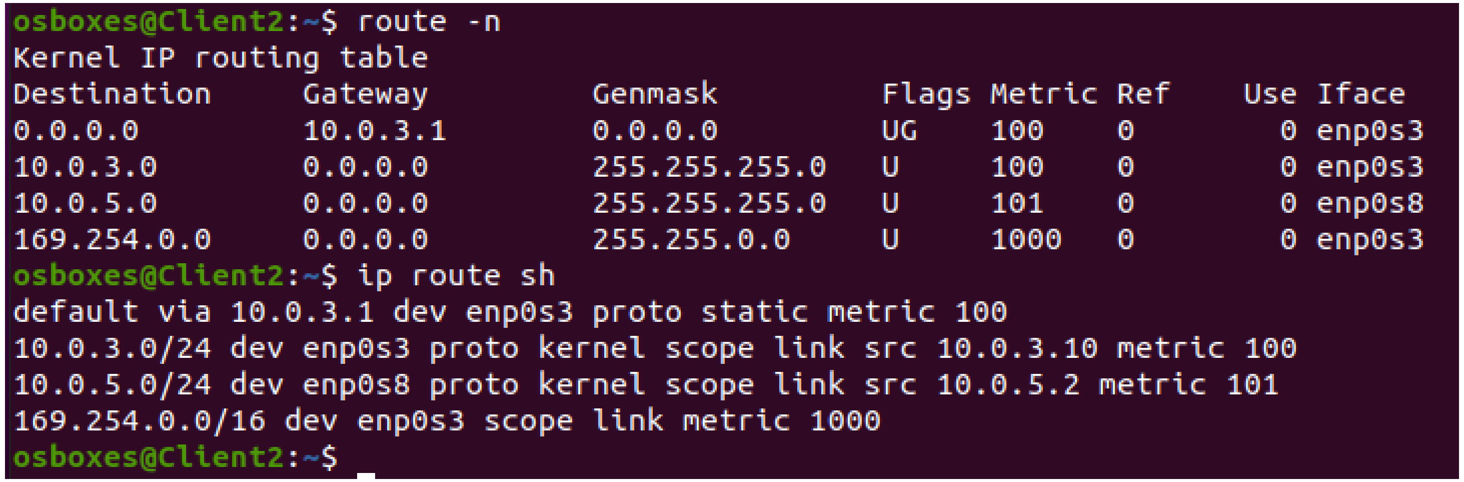


```
sergey@Server1:/etc/ssh$ ip route show
default via 192.168.1.1 dev enp0s3 proto static metric 100
10.0.1.0/24 dev enp0s8 proto kernel scope link src 10.0.1.1 metric 101
10.0.3.0/24 dev enp0s9 proto kernel scope link src 10.0.3.1 metric 102
169.254.0.0/16 dev enp0s3 scope link metric 1000
192.168.1.0/24 dev enp0s3 proto kernel scope link src 192.168.1.105 metric 100
```

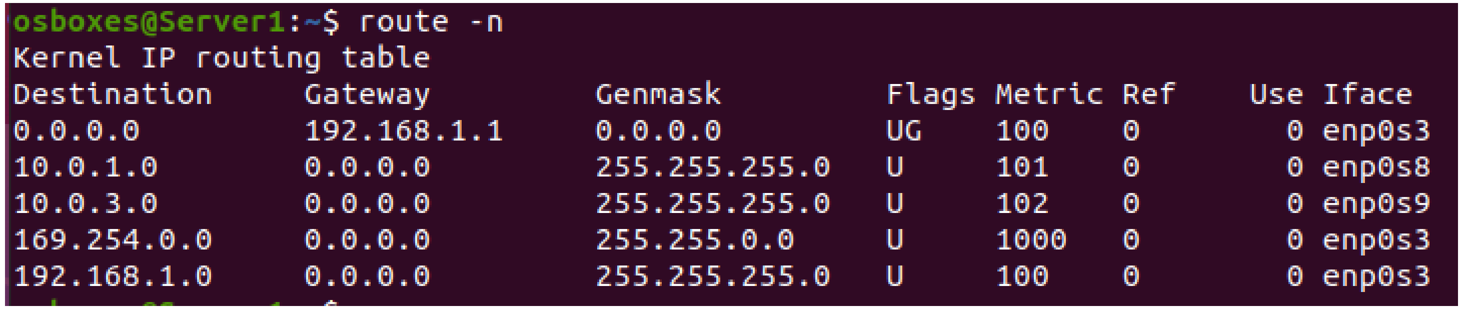
Routing Tables Review



```
osboxes@Client1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.1.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
10.0.5.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
```



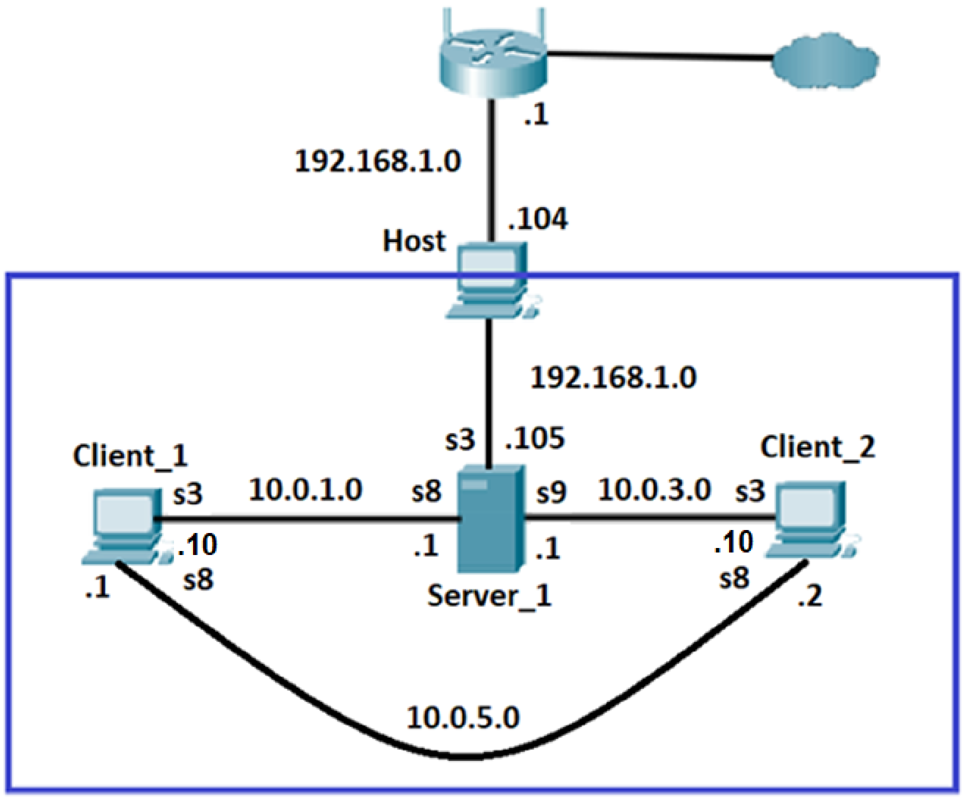
```
osboxes@Client2:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.3.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.3.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
10.0.5.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
osboxes@Client2:~$ ip route sh
default via 10.0.3.1 dev enp0s3 proto static metric 100
10.0.3.0/24 dev enp0s3 proto kernel scope link src 10.0.3.10 metric 100
10.0.5.0/24 dev enp0s8 proto kernel scope link src 10.0.5.2 metric 101
169.254.0.0/16 dev enp0s3 scope link metric 1000
osboxes@Client2:~$
```



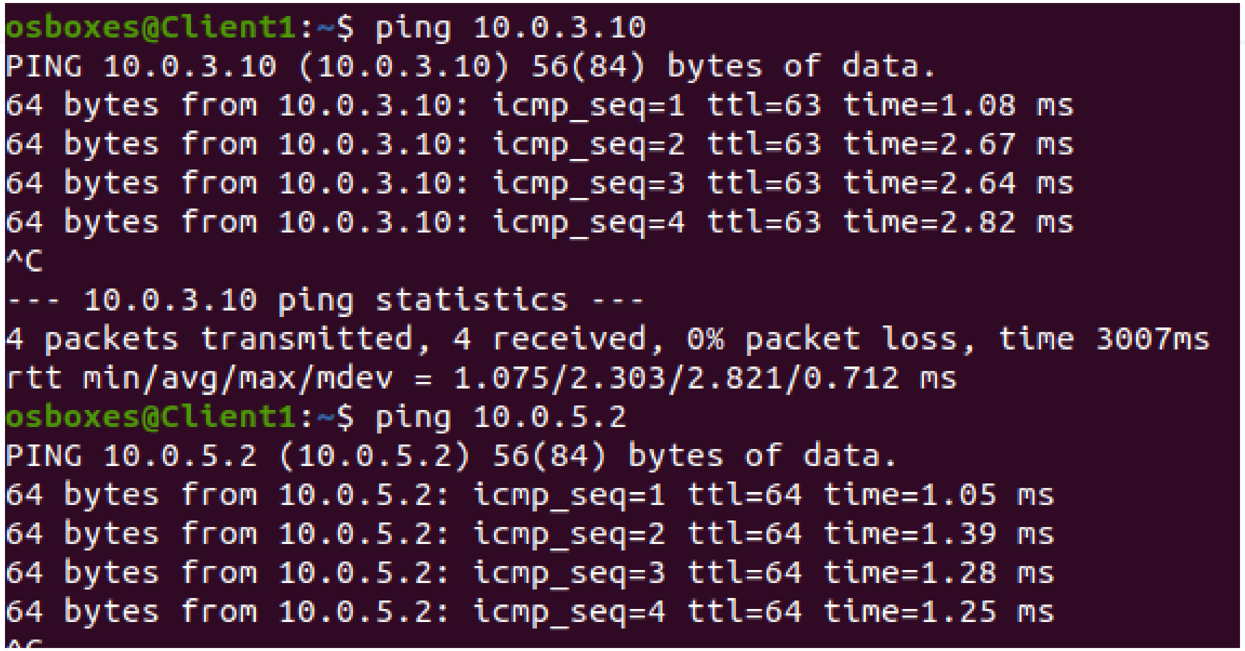
```
osboxes@Server1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
10.0.3.0        0.0.0.0         255.255.255.0   U     102    0        0 enp0s9
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
```

$route

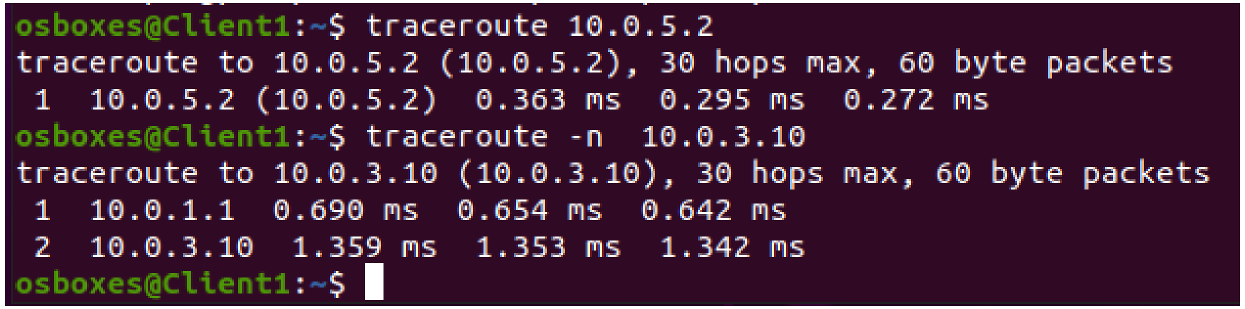
$ip route sh



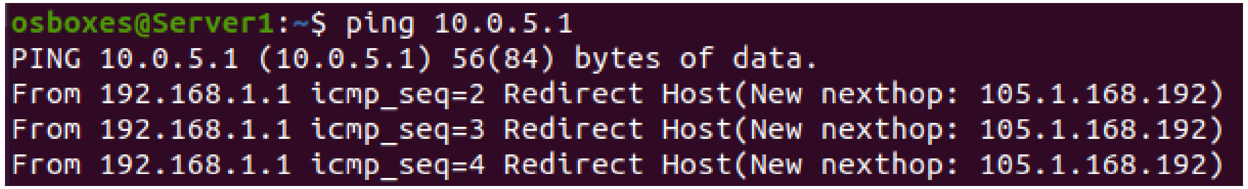
Connectivity Check



```
osboxes@Client1:~$ ping 10.0.3.10
PING 10.0.3.10 (10.0.3.10) 56(84) bytes of data.
64 bytes from 10.0.3.10: icmp_seq=1 ttl=63 time=1.08 ms
64 bytes from 10.0.3.10: icmp_seq=2 ttl=63 time=2.67 ms
64 bytes from 10.0.3.10: icmp_seq=3 ttl=63 time=2.64 ms
64 bytes from 10.0.3.10: icmp_seq=4 ttl=63 time=2.82 ms
^C
--- 10.0.3.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 1.075/2.303/2.821/0.712 ms
osboxes@Client1:~$ ping 10.0.5.2
PING 10.0.5.2 (10.0.5.2) 56(84) bytes of data.
64 bytes from 10.0.5.2: icmp_seq=1 ttl=64 time=1.05 ms
64 bytes from 10.0.5.2: icmp_seq=2 ttl=64 time=1.39 ms
64 bytes from 10.0.5.2: icmp_seq=3 ttl=64 time=1.28 ms
64 bytes from 10.0.5.2: icmp_seq=4 ttl=64 time=1.25 ms
^C
```

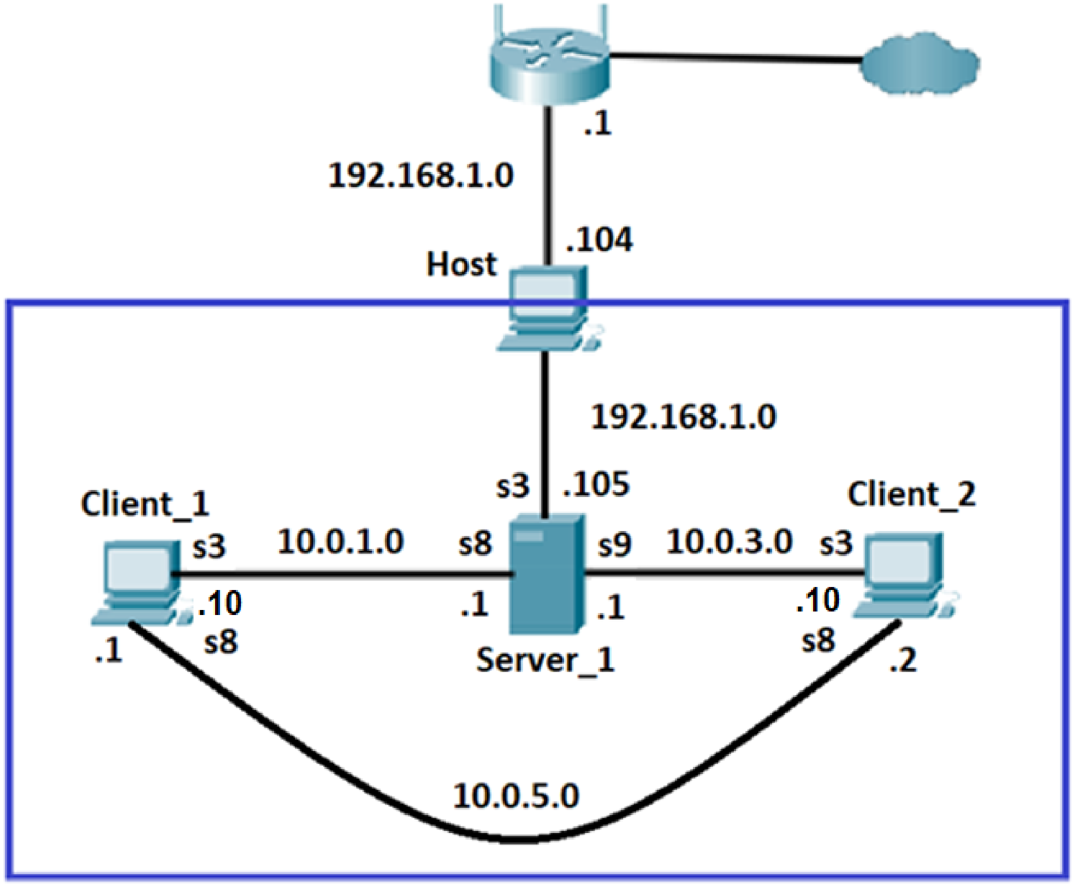


```
osboxes@Client1:~$ traceroute 10.0.5.2
traceroute to 10.0.5.2 (10.0.5.2), 30 hops max, 60 byte packets
 1  10.0.5.2 (10.0.5.2)  0.363 ms  0.295 ms  0.272 ms
osboxes@Client1:~$ traceroute -n  10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
 1  10.0.1.1  0.690 ms  0.654 ms  0.642 ms
 2  10.0.3.10  1.359 ms  1.353 ms  1.342 ms
osboxes@Client1:~$ 
```



```
osboxes@Server1:~$ ping 10.0.5.1
PING 10.0.5.1 (10.0.5.1) 56(84) bytes of data.
From 192.168.1.1 icmp_seq=2 Redirect Host(New nexthop: 105.1.168.192)
From 192.168.1.1 icmp_seq=3 Redirect Host(New nexthop: 105.1.168.192)
From 192.168.1.1 icmp_seq=4 Redirect Host(New nexthop: 105.1.168.192)
```

$ping
$traceroute



Routing Configuration

- Temporary routing configuration
  - *route*
  - *ip route*
- Permanent routing configuration
  - config files editing
  - nmcli utility using
- Dynamic routing configuration
  - quagga

ip route versus *route*

- The *ip route* suite is set to replace the net-tools suite (with *route* command) of network configuration tools. There are "synonym" commands that perform similar function in each.
- *route* is a fairly simple tool, perfect for creating static routes. It's still present in many distributions for compatibility.
- *ip route* is much more powerful, it has much more functionality, and can create more specialized rules.

route command

- **route** command in Linux is used when you want to work with the IP/kernel routing table. It is mainly used to set up static routes to specific hosts or networks via an interface. It is used for showing or update the IP/kernel routing table.
- Many Linux distributions do not have route command pre-installed. To install it:

Debian/Ubuntu

```
$sudo apt-get install net-tools
```

CentOS/RedHat

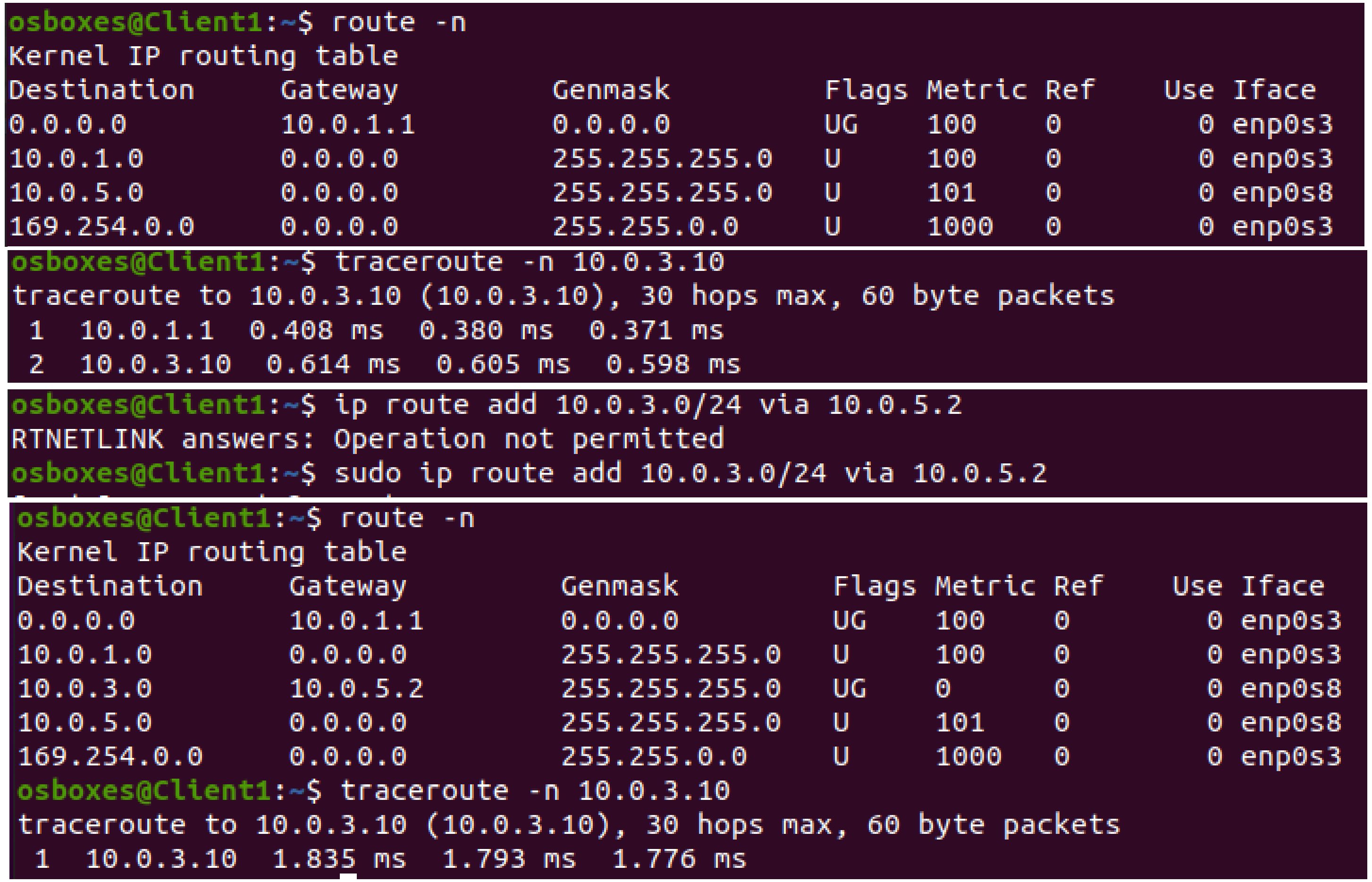
```
$sudo yum install net-tools
```

Fedora OS

```
$sudo dnf install net-tools
```

Temporary route adding

ip route add <network_ip>/<cidr> via <gateway_ip> [metric <metric>].

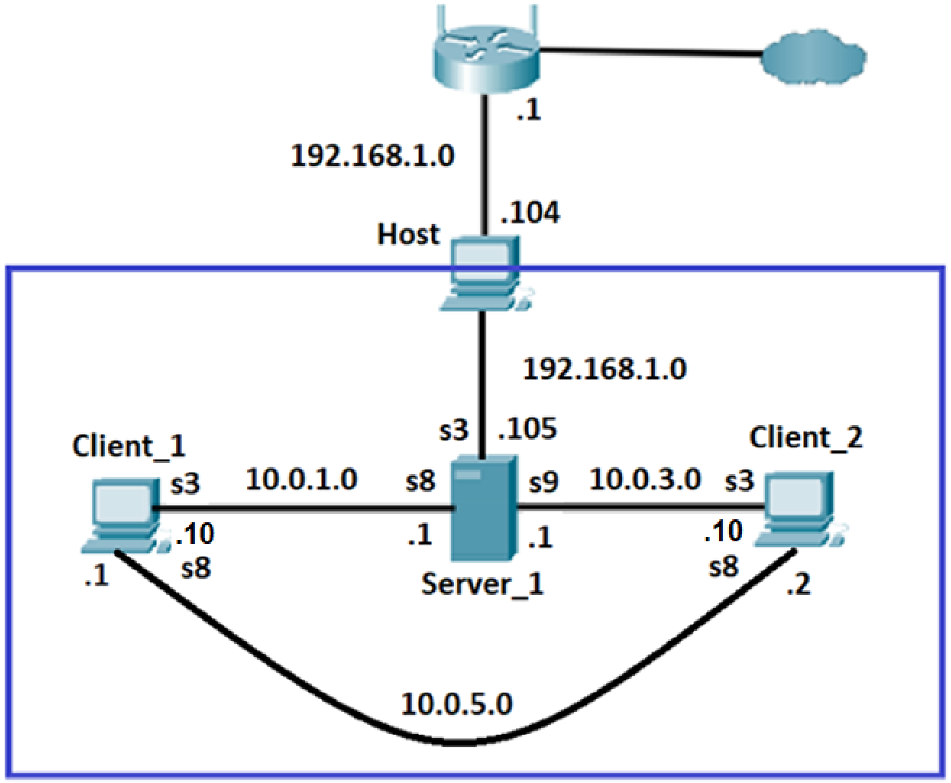


```
osboxes@Client1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.1.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
10.0.5.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
```

```
osboxes@Client1:~$ traceroute -n 10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
 1  10.0.1.1  0.408 ms  0.380 ms  0.371 ms
 2  10.0.3.10  0.614 ms  0.605 ms  0.598 ms
```

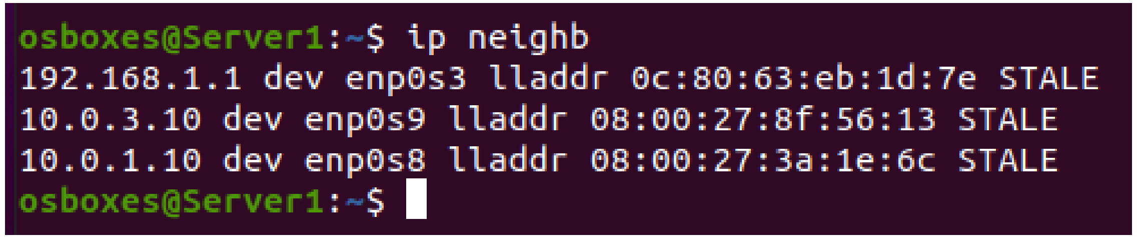
```
osboxes@Client1:~$ ip route add 10.0.3.0/24 via 10.0.5.2
RTNETLINK answers: Operation not permitted
osboxes@Client1:~$ sudo ip route add 10.0.3.0/24 via 10.0.5.2
```

```
osboxes@Client1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.1.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
10.0.3.0        10.0.5.2        255.255.255.0   UG    0      0        0 enp0s8
10.0.5.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
osboxes@Client1:~$ traceroute -n 10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
 1  10.0.3.10  1.835 ms  1.793 ms  1.776 ms
```

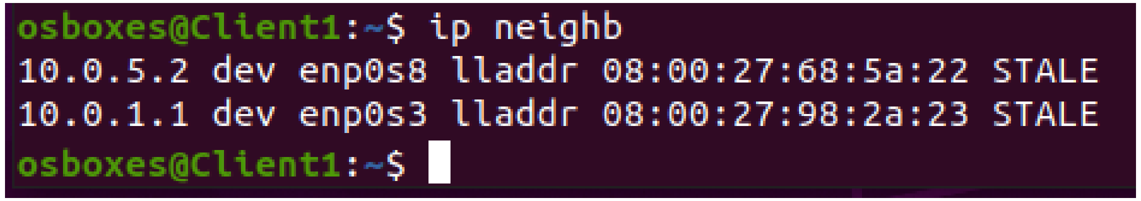


Neighbors addresses resolution

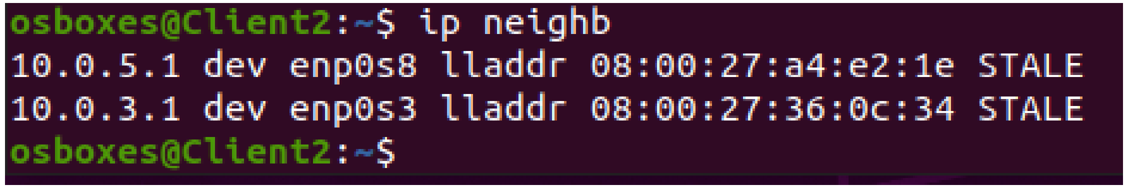
ip neighbors



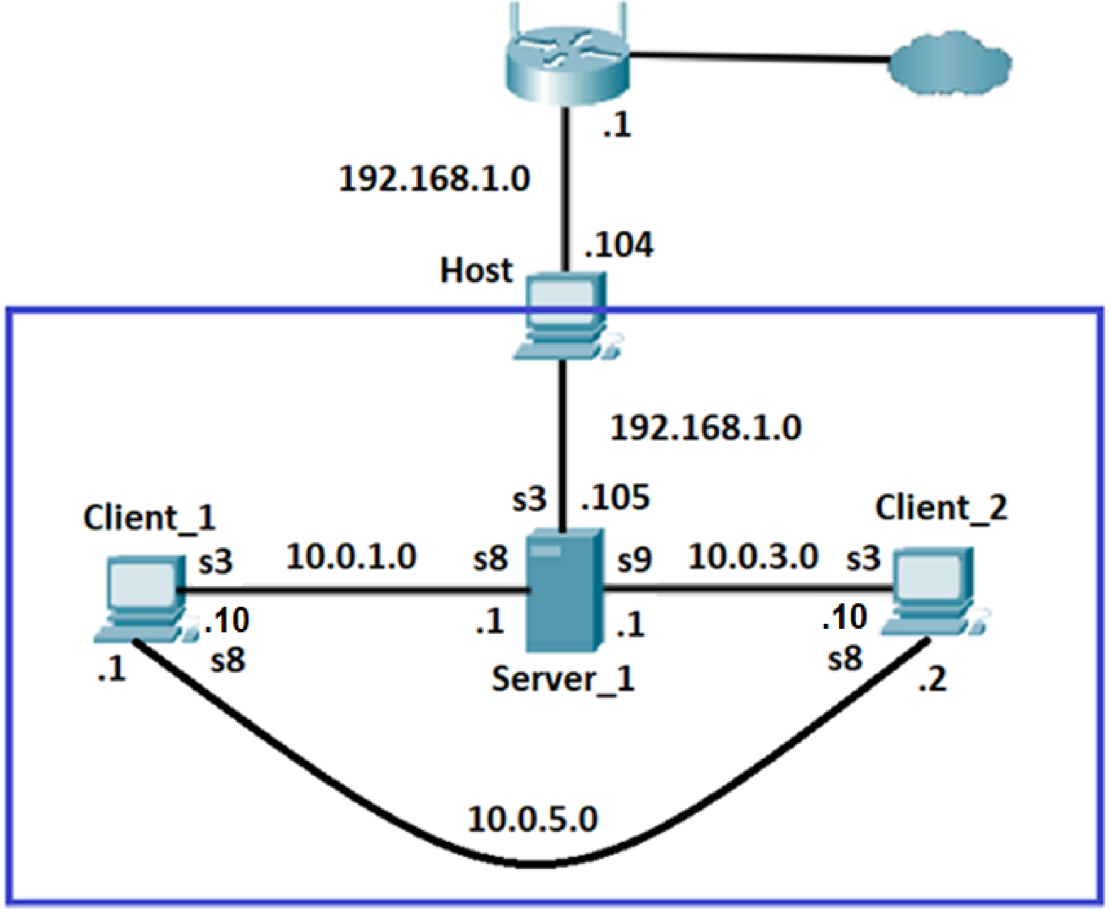
```
osboxes@Server1:~$ ip neighb
192.168.1.1 dev enp0s3 lladdr 0c:80:63:eb:1d:7e STALE
10.0.3.10 dev enp0s9 lladdr 08:00:27:8f:56:13 STALE
10.0.1.10 dev enp0s8 lladdr 08:00:27:3a:1e:6c STALE
osboxes@Server1:~$ 
```



```
osboxes@Client1:~$ ip neighb
10.0.5.2 dev enp0s8 lladdr 08:00:27:68:5a:22 STALE
10.0.1.1 dev enp0s3 lladdr 08:00:27:98:2a:23 STALE
osboxes@Client1:~$ 
```



```
osboxes@Client2:~$ ip neighb
10.0.5.1 dev enp0s8 lladdr 08:00:27:a4:e2:1e STALE
10.0.3.1 dev enp0s3 lladdr 08:00:27:36:0c:34 STALE
osboxes@Client2:~$
```



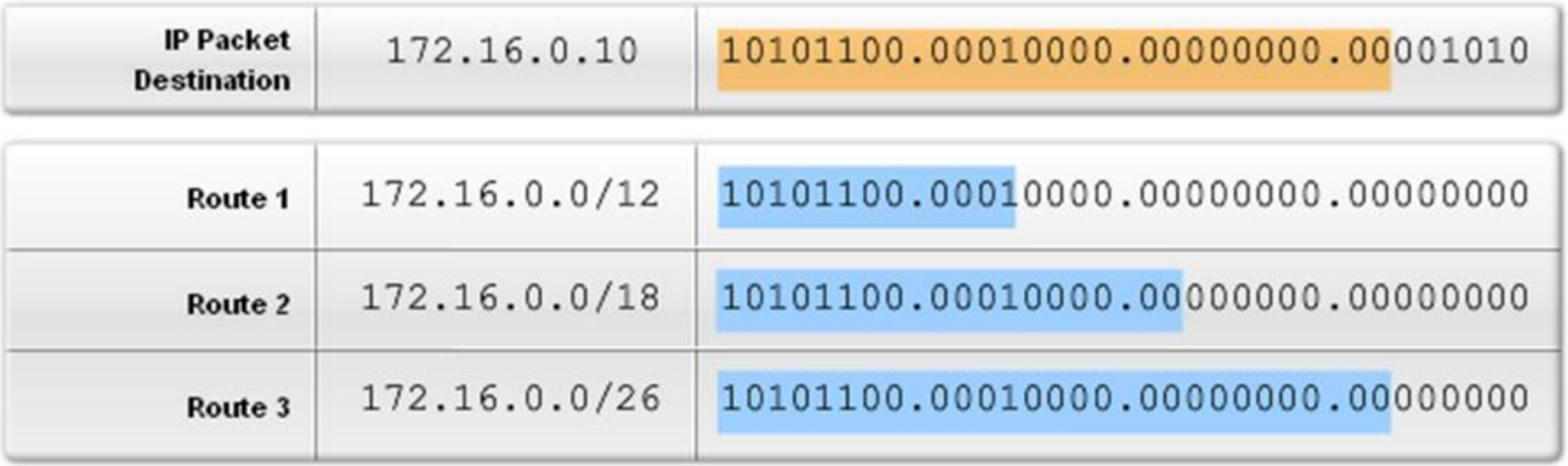
Priorities of routes

If the routing table contains two or more routes for a particular network, the following rules will be used for the final route selection:

- The “longest match” rule
- The “lowest metric” rule
- The “equal cost load balancing” rule

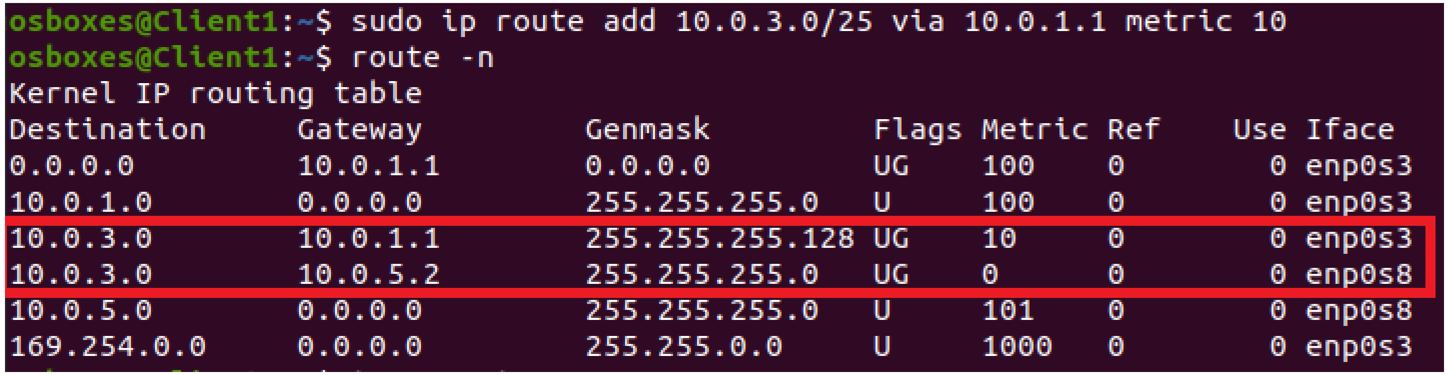
The “longest match” rule

The most priority has the route with the biggest prefix

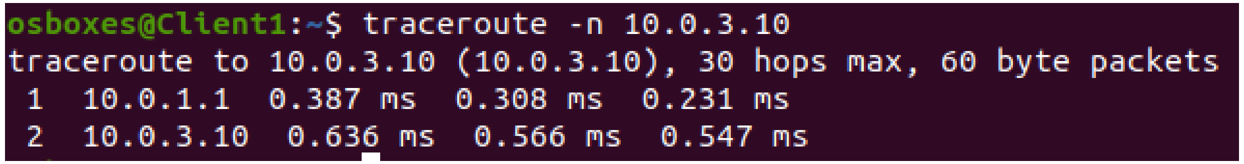


IP Packet Destination	172.16.0.10	10101100.00010000.00000000.00001010
Route 1	172.16.0.0/12	10101100.00010000.00000000.00000000
Route 2	172.16.0.0/18	10101100.00010000.00000000.00000000
Route 3	172.16.0.0/26	10101100.00010000.00000000.00000000

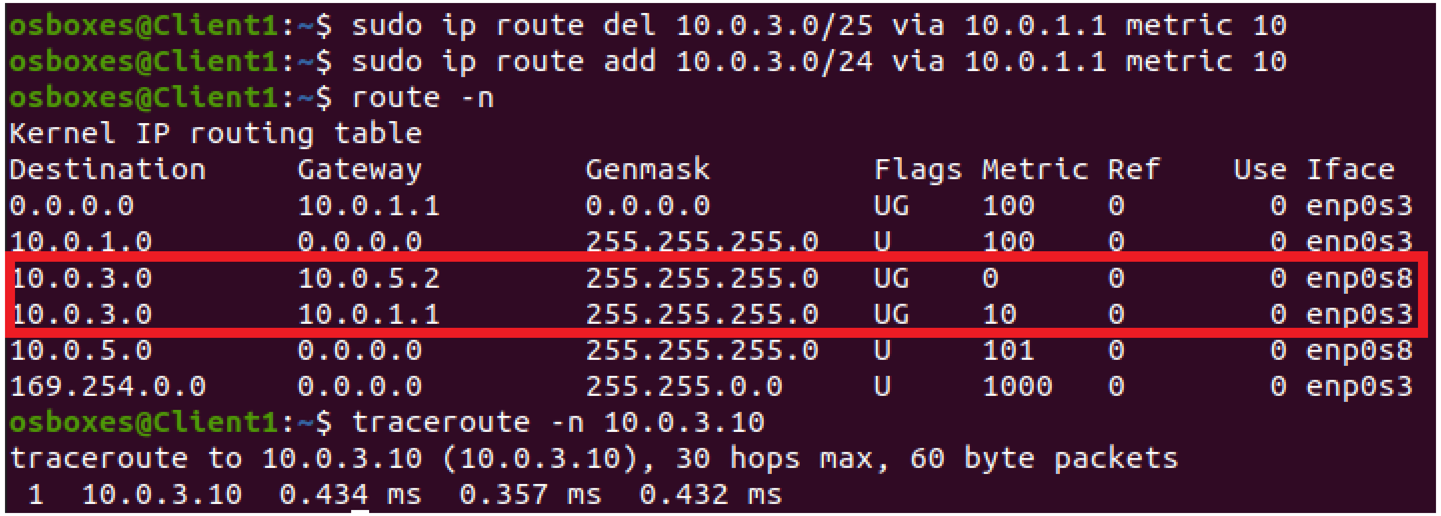
The “longest match” and “lowest metric” rules



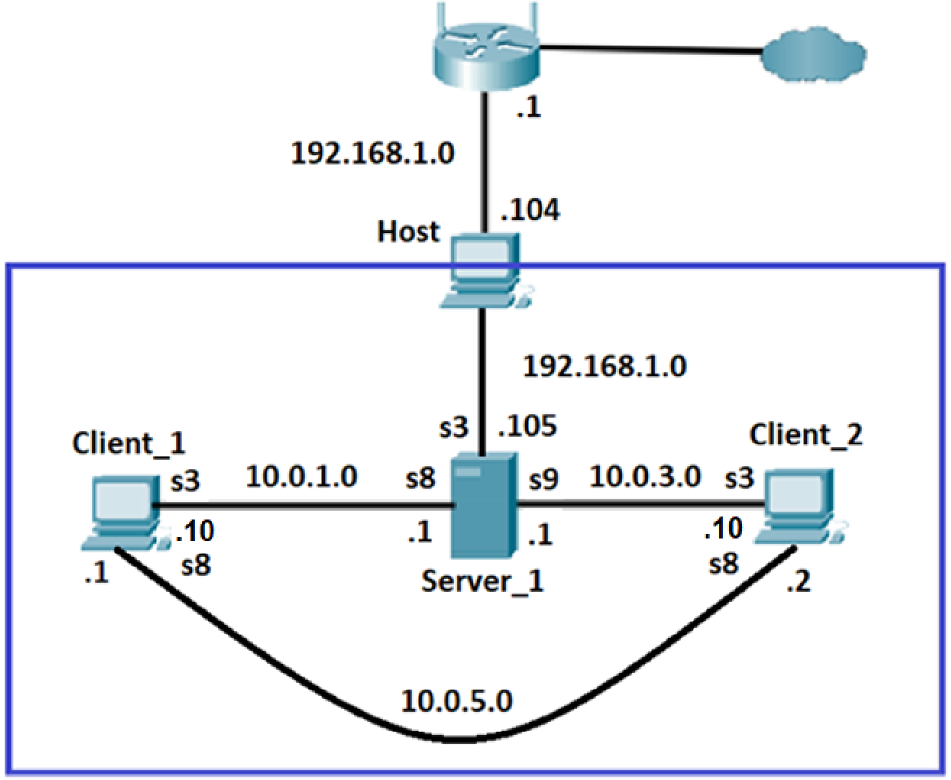
```
osboxes@Client1:~$ sudo ip route add 10.0.3.0/25 via 10.0.1.1 metric 10
osboxes@Client1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.1.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
10.0.3.0        10.0.1.1        255.255.255.128 UG    10     0        0 enp0s3
10.0.3.0        10.0.5.2        255.255.255.0   UG    0      0        0 enp0s8
10.0.5.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
```



```
osboxes@Client1:~$ traceroute -n 10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
 1  10.0.1.1  0.387 ms  0.308 ms  0.231 ms
 2  10.0.3.10  0.636 ms  0.566 ms  0.547 ms
```

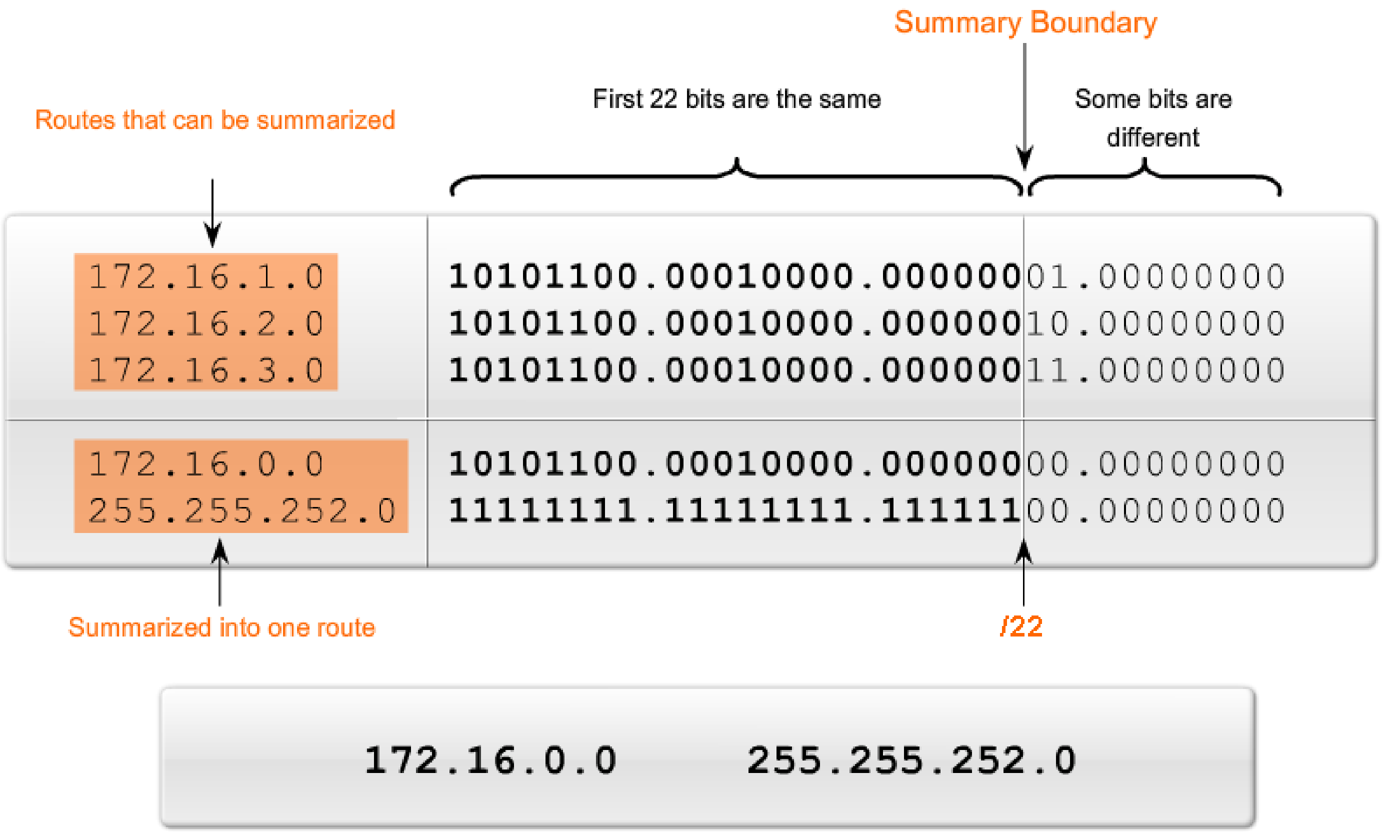


```
osboxes@Client1:~$ sudo ip route del 10.0.3.0/25 via 10.0.1.1 metric 10
osboxes@Client1:~$ sudo ip route add 10.0.3.0/24 via 10.0.1.1 metric 10
osboxes@Client1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.1.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
10.0.3.0        10.0.5.2        255.255.255.0   UG    0      0        0 enp0s8
10.0.3.0        10.0.1.1        255.255.255.0   UG    10     0        0 enp0s3
10.0.5.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
osboxes@Client1:~$ traceroute -n 10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
 1  10.0.3.10  0.434 ms  0.357 ms  0.432 ms
```



Summary Static Routes

- Summarizing routes **reduces** the size of the routing table.
- Route summarization is the process of combining a number of static routes into a single static route.
- Multiple static routes can be summarized into a single static route if:
    - The destination networks can be summarized into a single network address
    - The multiple static routes all use the same exit-interface or next-hop IP address



Summary Boundary

Routes that can be summarized

First 22 bits are the same

Some bits are different

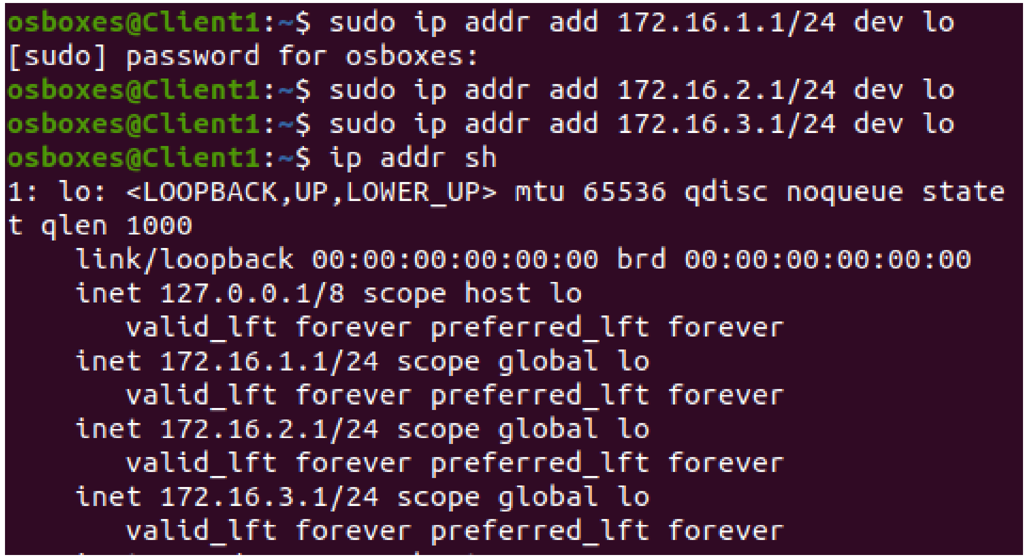
172.16.1.0	10101100.00010000.00000001.00000000
172.16.2.0	10101100.00010000.00000010.00000000
172.16.3.0	10101100.00010000.00000011.00000000
172.16.0.0	10101100.00010000.00000000.00000000
255.255.252.0	11111111.11111111.11111100.00000000

Summarized into one route

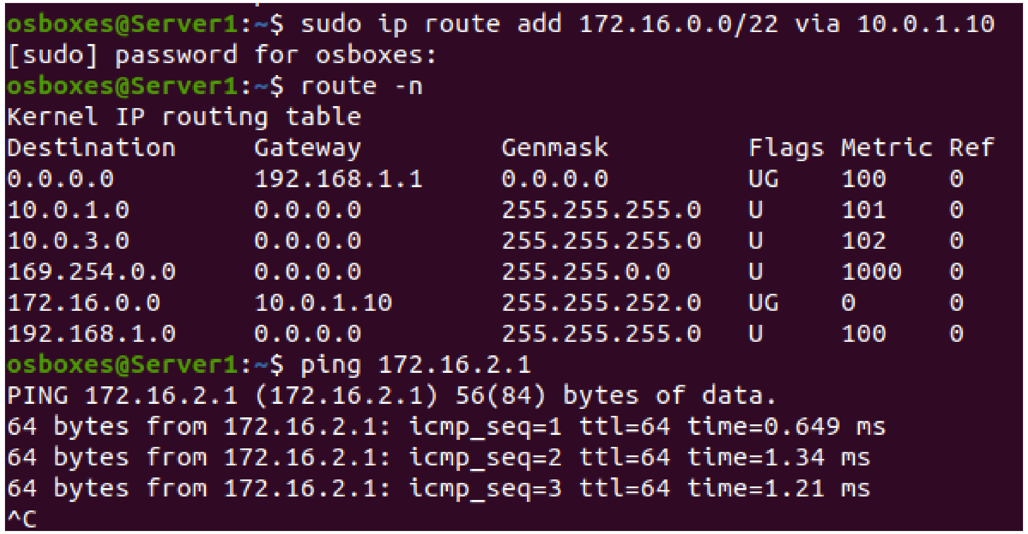
/22

172.16.0.0 255.255.252.0

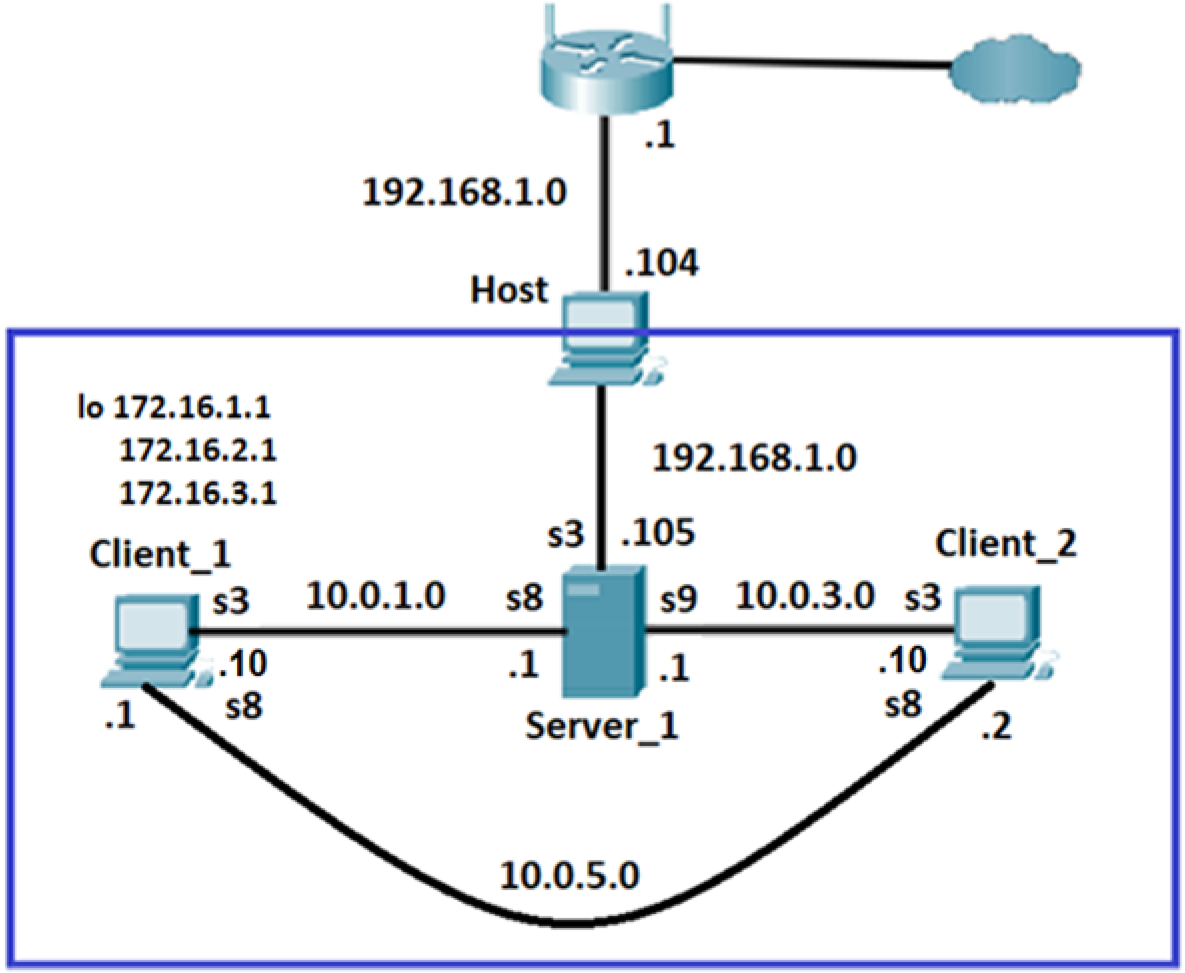
Summary Static Routes



```
osboxes@Client1:~$ sudo ip addr add 172.16.1.1/24 dev lo
[sudo] password for osboxes:
osboxes@Client1:~$ sudo ip addr add 172.16.2.1/24 dev lo
osboxes@Client1:~$ sudo ip addr add 172.16.3.1/24 dev lo
osboxes@Client1:~$ ip addr sh
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet 172.16.1.1/24 scope global lo
       valid_lft forever preferred_lft forever
    inet 172.16.2.1/24 scope global lo
       valid_lft forever preferred_lft forever
    inet 172.16.3.1/24 scope global lo
       valid_lft forever preferred_lft forever
```

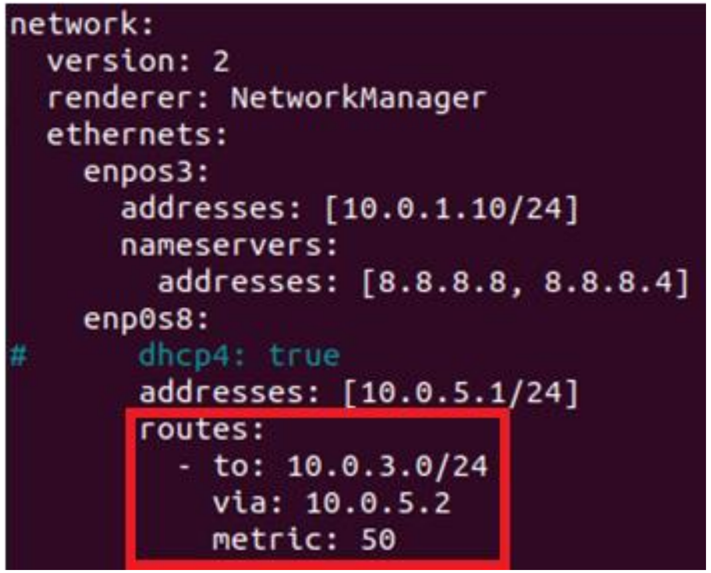


```
osboxes@Server1:~$ sudo ip route add 172.16.0.0/22 via 10.0.1.10
[sudo] password for osboxes:
osboxes@Server1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0
10.0.1.0        0.0.0.0         255.255.255.0   U     101    0
10.0.3.0        0.0.0.0         255.255.255.0   U     102    0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0
172.16.0.0      10.0.1.10       255.255.252.0   UG    0      0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0
osboxes@Server1:~$ ping 172.16.2.1
PING 172.16.2.1 (172.16.2.1) 56(84) bytes of data.
64 bytes from 172.16.2.1: icmp_seq=1 ttl=64 time=0.649 ms
64 bytes from 172.16.2.1: icmp_seq=2 ttl=64 time=1.34 ms
64 bytes from 172.16.2.1: icmp_seq=3 ttl=64 time=1.21 ms
^C
```

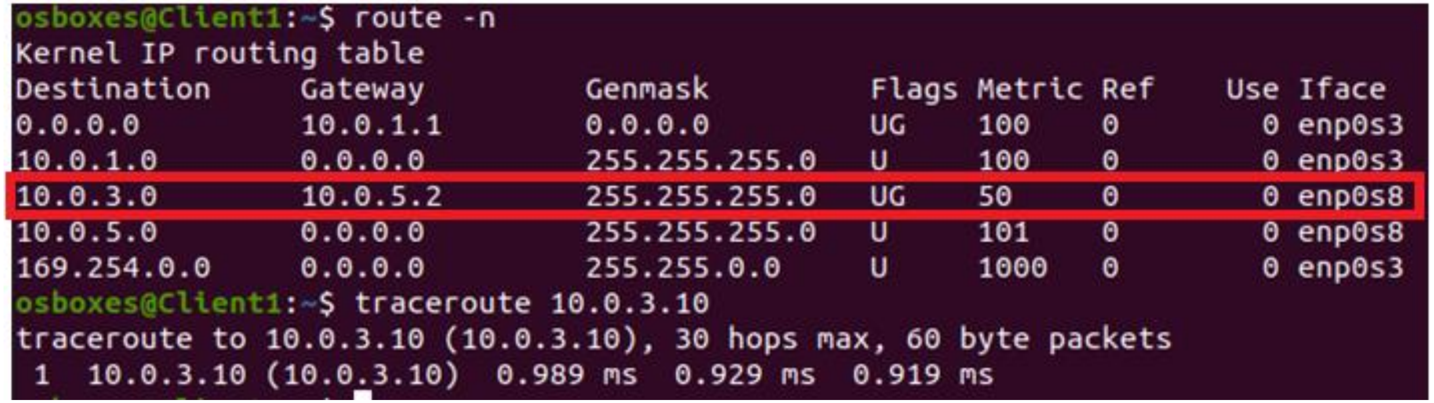


Permanent routing configuration

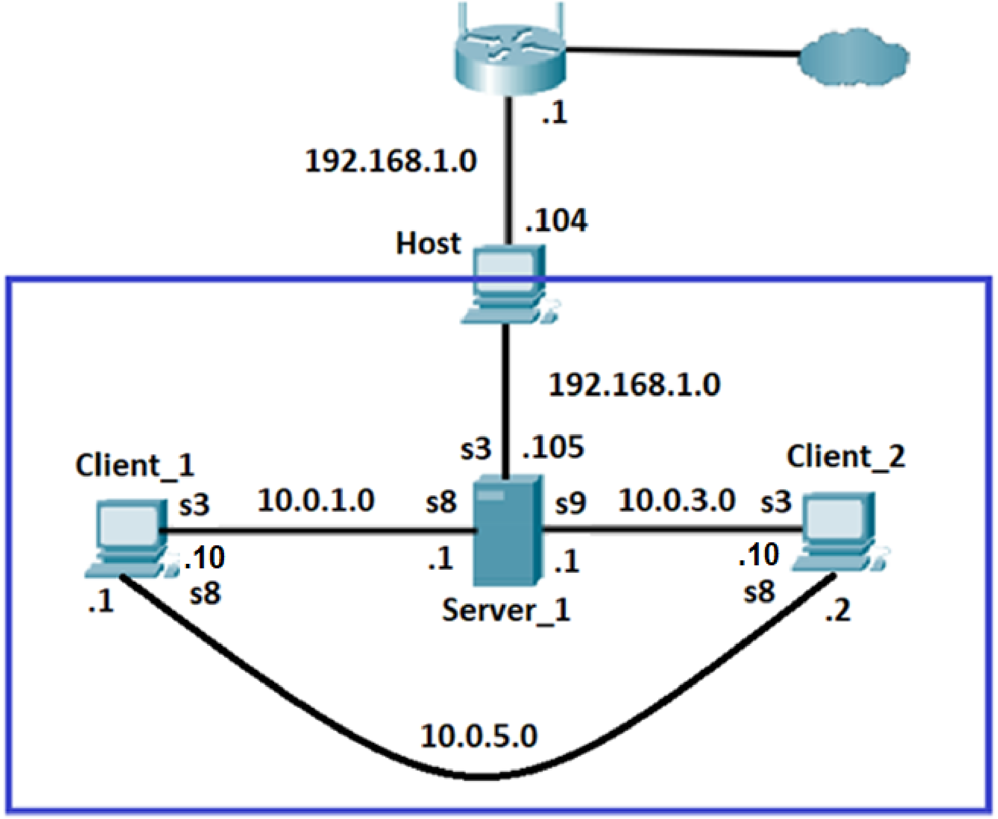
To add a permanent route in Ubuntu use Netplan.



```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enpos3:
      addresses: [10.0.1.10/24]
      nameservers:
        addresses: [8.8.8.8, 8.8.8.4]
    enp0s8:
#       dhcp4: true
       addresses: [10.0.5.1/24]
       routes:
         - to: 10.0.3.0/24
           via: 10.0.5.2
           metric: 50
```



```
osboxes@Client1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.1.1        0.0.0.0         UG    100    0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
10.0.3.0        10.0.5.2        255.255.255.0   UG    50     0        0 enp0s8
10.0.5.0        0.0.0.0         255.255.255.0   U     101    0        0 enp0s8
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
osboxes@Client1:~$ traceroute 10.0.3.10
traceroute to 10.0.3.10 (10.0.3.10), 30 hops max, 60 byte packets
 1  10.0.3.10 (10.0.3.10)  0.989 ms  0.929 ms  0.919 ms
```



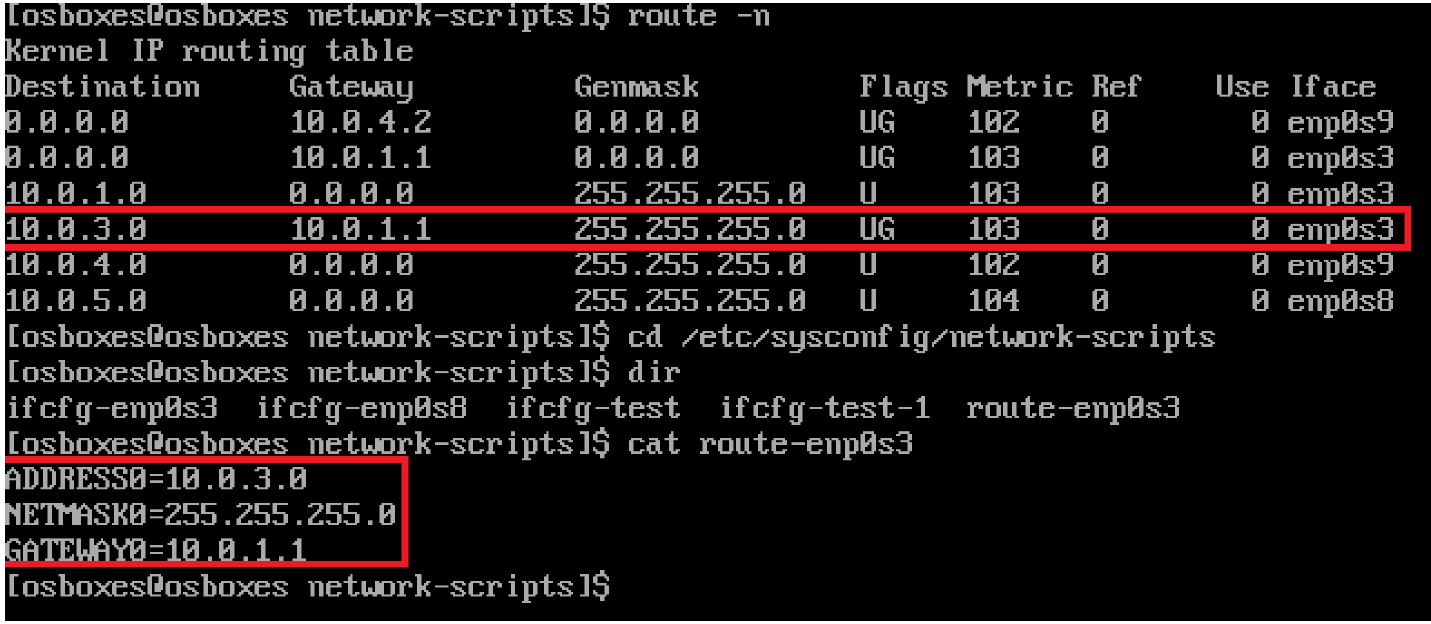
Permanent routing configuration in another Linux distributives

Ununtu legacy (before 18.04 version): **edit** the “/etc/network/interfaces”:

```
auto eth0
iface eth0 inet static
    address 10.0.2.2
    netmask 255.255.255.0
    up route add -net 10.0.3.0 netmask 255.255.255.0 gw 10.0.2.1
```

On RHEL and CentOS distributions, you need to **create** a file named “route-<device>” in the “/etc/sysconfig/network-scripts” folder:

```
$ sudo vi /etc/sysconfig/network-scripts/route-enp0s3
```



```
[osboxes@osboxes network-scripts]$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.4.2        0.0.0.0         UG    102    0        0 enp0s9
0.0.0.0         10.0.1.1        0.0.0.0         UG    103    0        0 enp0s3
10.0.1.0        0.0.0.0         255.255.255.0   U     103    0        0 enp0s3
10.0.3.0        10.0.1.1        255.255.255.0   UG    103    0        0 enp0s3
10.0.4.0        0.0.0.0         255.255.255.0   U     102    0        0 enp0s9
10.0.5.0        0.0.0.0         255.255.255.0   U     104    0        0 enp0s8
[osboxes@osboxes network-scripts]$ cd /etc/sysconfig/network-scripts
[osboxes@osboxes network-scripts]$ dir
ifcfg-enp0s3  ifcfg-enp0s8  ifcfg-test  ifcfg-test-1  route-enp0s3
[osboxes@osboxes network-scripts]$ cat route-enp0s3
ADDRESS0=10.0.3.0
NETMASK0=255.255.255.0
GATEWAY0=10.0.1.1
[osboxes@osboxes network-scripts]$
```

Permanent routing configuration via *nmcli* in CentOS

To add the new route in routing table:

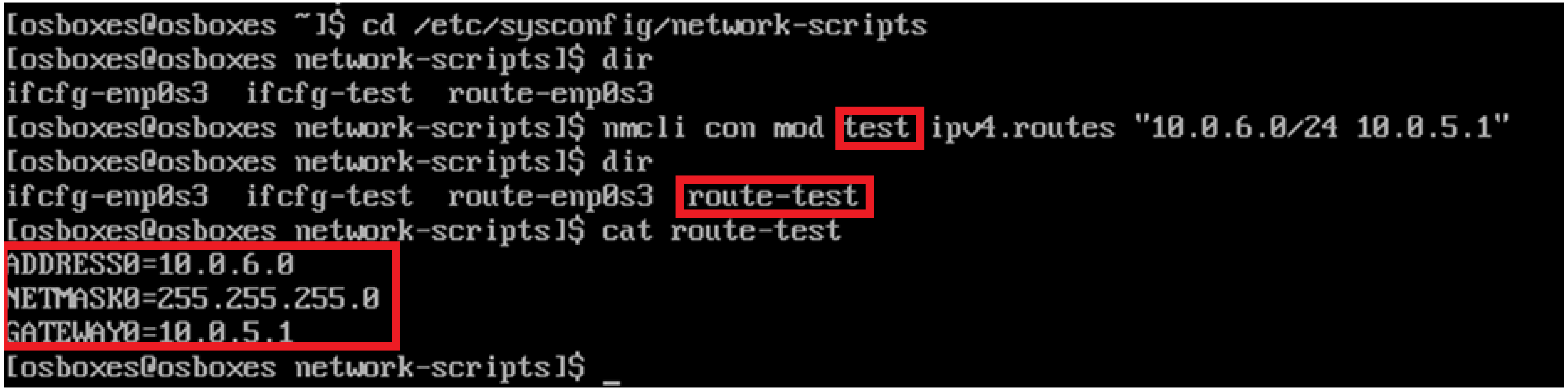
nmcli connection modify <conn-name> ipv4.routes “<network ip-addr>/<prefix> <gateway>“

systemctl restart network

Example:

nmcli con mod test ipv4.routes “10.0.6.0/24 10.0.5.1”

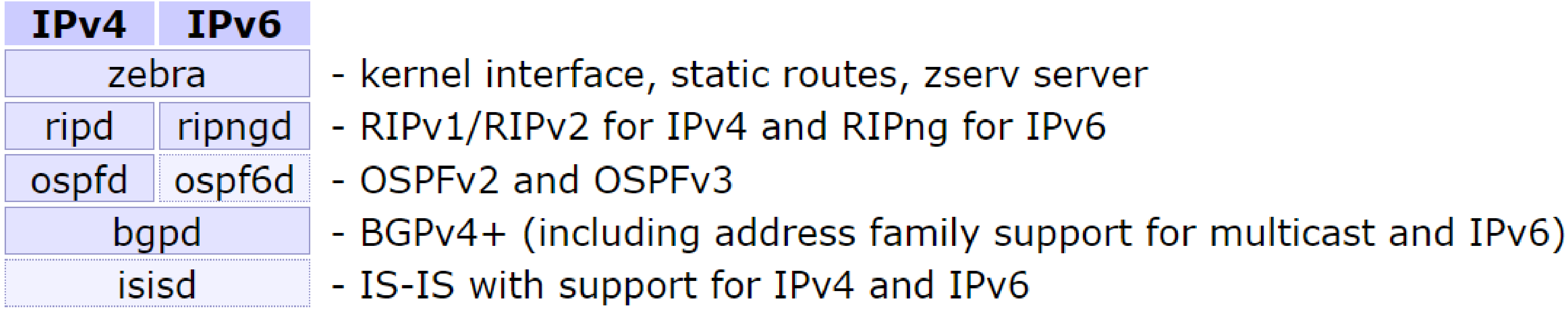
systemctl restart network



```
[osboxes@osboxes ~]$ cd /etc/sysconfig/network-scripts
[osboxes@osboxes network-scripts]$ dir
ifcfg-enp0s3  ifcfg-test  route-enp0s3
[osboxes@osboxes network-scripts]$ nmcli con mod test ipv4.routes "10.0.6.0/24 10.0.5.1"
[osboxes@osboxes network-scripts]$ dir
ifcfg-enp0s3  ifcfg-test  route-enp0s3  route-test
[osboxes@osboxes network-scripts]$ cat route-test
ADDRESS0=10.0.6.0
NETMASK0=255.255.255.0
GATEWAY0=10.0.5.1
[osboxes@osboxes network-scripts]$ _
```

Dynamic routing in Linux

- Quagga is a routing software suite, providing implementations of OSPFv2, OSPFv3, RIP v1 and v2, RIPng and BGP-4 for Unix platforms.
- The Quagga architecture consists of a core daemon, *zebra*, which acts as an abstraction layer to the underlying Unix kernel and presents the Zserv API over a Unix or TCP stream to Quagga clients.
- It is these Zserv clients which typically implement a routing protocol and communicate routing updates to the zebra daemon. Existing Zserv implementations are:



IPv4	IPv6	
zebra		- kernel interface, static routes, zserv server
ripd	ripngd	- RIPv1/RIPv2 for IPv4 and RIPng for IPv6
ospfd	ospf6d	- OSPFv2 and OSPFv3
bgpd		- BGPv4+ (including address family support for multicast and IPv6)
isisd		- IS-IS with support for IPv4 and IPv6

- Quagga daemons are each configurable via a network accessible CLI (called a 'vty'). The CLI follows a style similar to that of other routing software.

Quagga installation and configuration

- Ubuntu Quagga installation: sudo apt install quagga-core
- Configuration files should be in /etc/quagga
- Each router daemon gets its own configuration file, for example /etc/quagga/ospfd.conf:

```
hostname router1
log file /var/log/quagga/ospfd.log
router ospf
 ospf router-id 192.168.110.15
 network 192.168.110.0/0 area 0.0.0.0
 network 192.168.120.0/0 area 0.0.0.0
access-list localhost permit 127.0.0.1/32
access-list localhost deny any
line vty
  access-class localhost
```

Thank you!