Simple, test, 5-router Internetwork

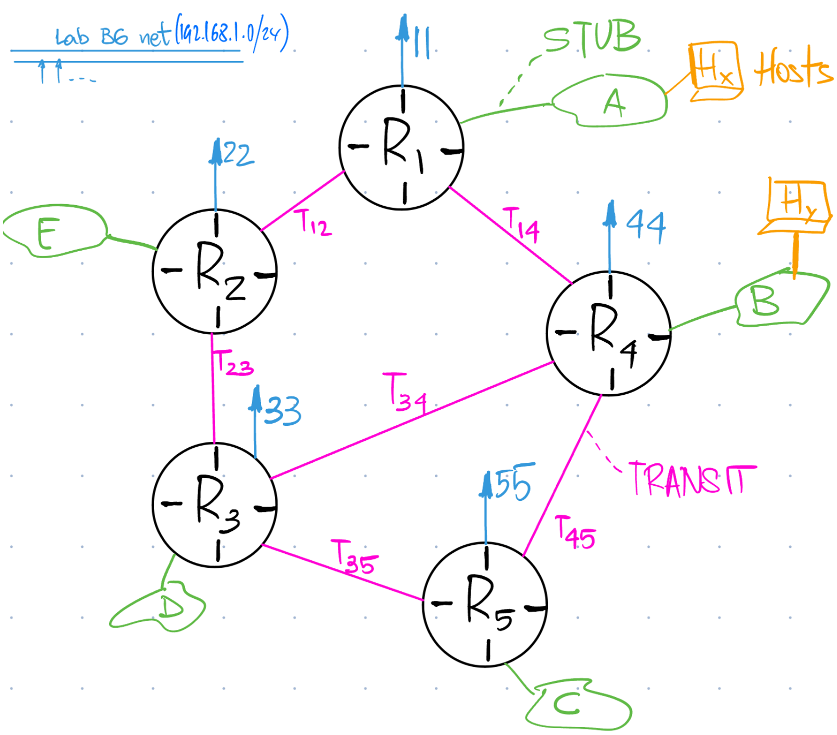


Figure 1. Prototype internetwork for practice

- Debian Linux
- TRANSIT or STUB links
- STUB links connect to LANs where servers and users are connected
    - o Size is in all cases 64 IP's, or CIDR /32-log64 = /32-6 = /26 which netmask is 255.255.255.192
    - o Next week (Week #22) we'll connect one host per stub network and we'll check full bidirectional connectivity between each pair of hosts
    - o Every router should have a route to each stub network

- TRANSIT links are point-to-point links which goal is only to permit the bidirectional communication between two directly connected routers
    - o Size is in all cases 8 IP's, CIDR = /29, Netmask 255.255.255.248
    - o Routing tables at each of Ri are calculated by applying Dijkstra by hand
    - o No hosts should be connected to any of these networks

- All routers keep a physical connection with Lab B6 network (192.168.1.0/24), so remote access and Internet access is guaranteed. The default router is, consequently 192.168.1.1. We refer to the helper network as *scaffold network.*

- LAN switches implement VLAN (IEEE 802.1Q) for delimiting (Separating) the broadcast domains (LANs). Linux also implements

VLAN at each physical interface which turns out into each host availing as many *logical interfaces* as necessary. The notation that is used in /etc/network/interfaces for referring to VLAN interfaces is, for example, eno1.1202 if we want to refer to the VLAN 1202 as implemented upon the physical interface eno1.

IP address allocation

- Use pool 192.168.2.0/23. Next week we'll carry out the whole allocation process by applying an algorithm that guarantees the correct alignment of each IP block.
    - Stub nets start at 192.168.2.0
        - Stub size is 64, or /26 = netmask 255.255.255.192
    - Transit nets start at 192.168.3.128
        - Transit net size is 8, or /29 = netmask 255.255.255.248
    - Both spaces (Stub nets and Transit nets) are correctly joined into the pool mentioned above 192.168.2.0/23. This will be proved **correct in next week**'s B1/B3 session.

STUB NETWORKS :: SIZE 64:: /26 :: 255.255.255.192

STUB LABEL	Start	End	VLAN
A	192.168.2.0	192.168.2.63	101
B	192.168.2.64	192.168.2.127	202
C	192.168.2.128	192.168.2.191	303
D	192.168.2.192	192.168.2.255	404
E	192.168.3.0	192.168.3.63	505

TRANSIT NETWORKS :: SIZE 8 :: /29 :: 255.255.255.248

STUB LABEL	Start	End	VLAN
T_{12}	192.168.3.128	192.168.3.135	707
T_{23}	192.168.3.136	192.168.3.143	1000
T_{35}	192.168.2.144	192.168.2.151	1101
T_{45}	192.168.2.152	192.168.2.159	1202
T_{14}	192.168.3.160	192.168.3.167	808
T_{34}	192.168.3.168	192.168.3.175	1303

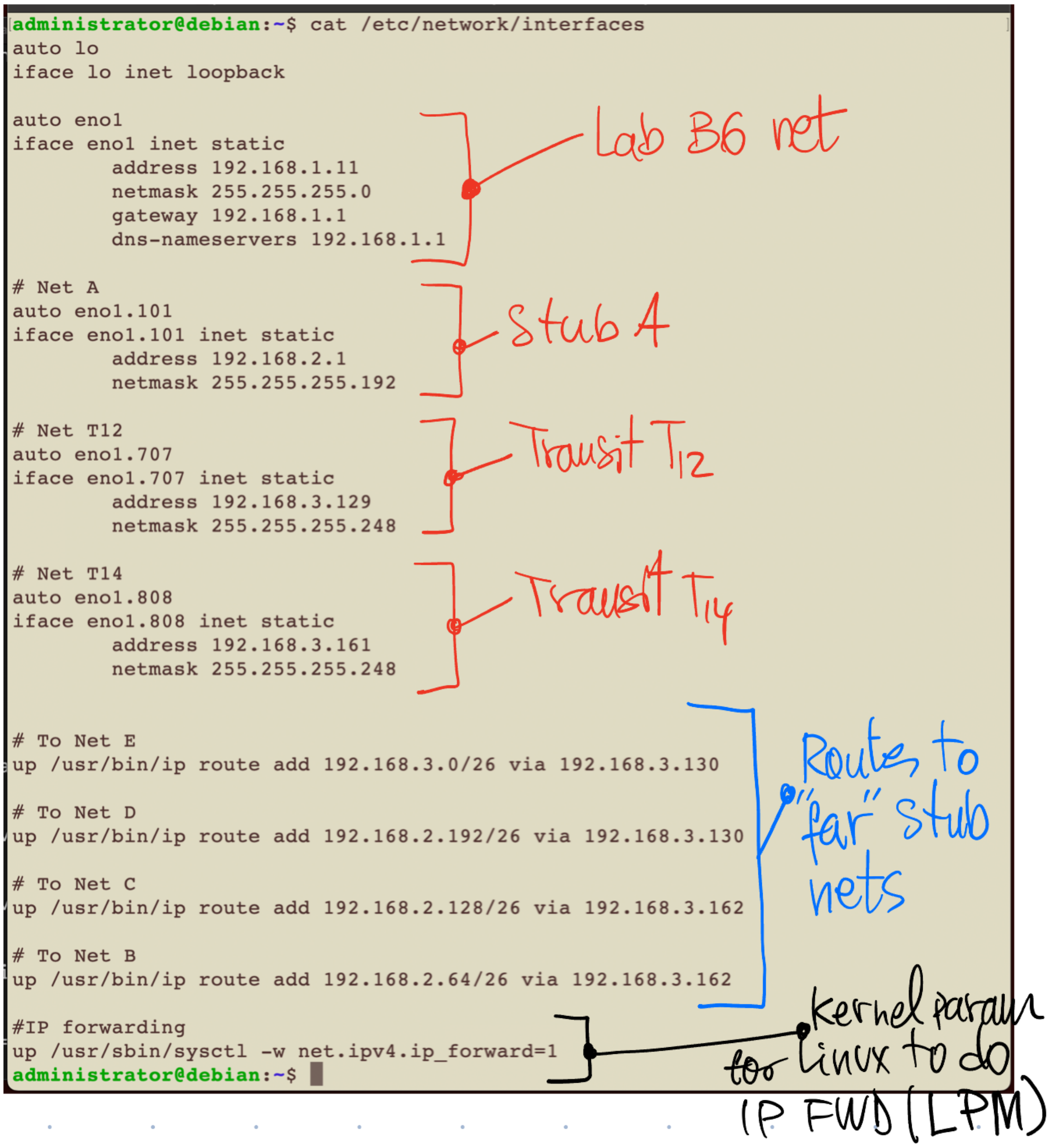
Exercise 1: Skimming /etc/network/interfaces at R_1

Connect with router R_1 by using the following command (The IP address to be used is 192.168.1.11 since the router's suffix is 1, R_1). This is the convention that we'll use along the whole practical:

```
$ ssh administrator@192.168.1.11
```

(Password is XXXcb1x2q%)

Check the contents of the /etc/network/interfaces file in R_1; use the cat command (Avoid editing the file altogether, please). Observe that R_1 has direct-link (Ethernets) configurations for all of its networks:



```
administrator@debian:~$ cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eno1
iface eno1 inet static
        address 192.168.1.11
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 192.168.1.1

# Net A
auto eno1.101
iface eno1.101 inet static
        address 192.168.2.1
        netmask 255.255.255.192

# Net T12
auto eno1.707
iface eno1.707 inet static
        address 192.168.3.129
        netmask 255.255.255.248

# Net T14
auto eno1.808
iface eno1.808 inet static
        address 192.168.3.161
        netmask 255.255.255.248

# To Net E
up /usr/bin/ip route add 192.168.3.0/26 via 192.168.3.130

# To Net D
up /usr/bin/ip route add 192.168.2.192/26 via 192.168.3.130

# To Net C
up /usr/bin/ip route add 192.168.2.128/26 via 192.168.3.162

# To Net B
up /usr/bin/ip route add 192.168.2.64/26 via 192.168.3.162

#IP forwarding
up /usr/sbin/sysctl -w net.ipv4.ip_forward=1
administrator@debian:~$
```

Lab B6 net
Stub A
Transit T_{12}
Transit T_{14}
Routes to "far" stub nets
kernel param for Linux to do IP FWD (LPM)

Figure 2. Checking the contents of file /etc/network/interfaces R_1

All of the remaining routers have equivalent network configurations that permit them to implement the connectivity captured in the network diagram inf Fig. 1. These files can be downloaded from paloalto just in case that is necessary:

```
$ wget paloalto.unileon.es/cn/labs/interfaces.11
```

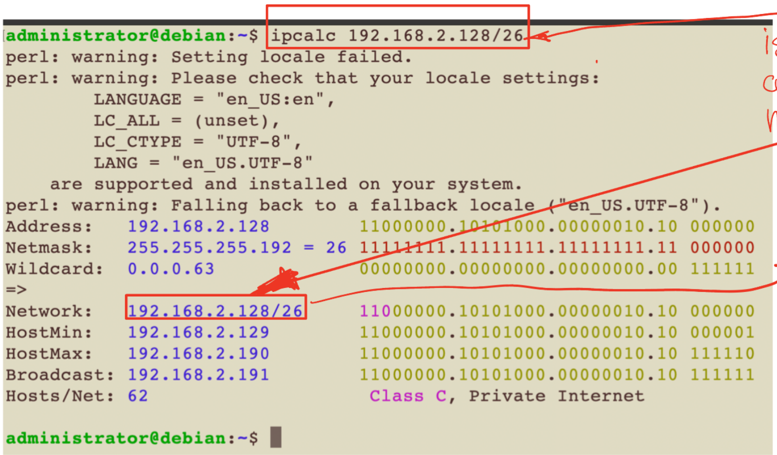
```
$ wget paloalto.unileon.es/cn/labs/interfaces.22 etc…
```

(In Lab B6, use IP 192.168.1.89 for accessing the server)

Exercise 2: Check one stub network's IP block for correctness

Use the **ipcalc** utility for checking that the IP block allocated to network C is correct (This exercise can be done at your local host, no need for remote access to any other host):

$ ipcalc 192.168.2.128/26



```
administrator@debian:~$ ipcalc 192.168.2.128/26
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en_US:en",
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to a fallback locale ("en_US.UTF-8").
Address:   192.168.2.128         11000000.10101000.00000010.10 000000
Netmask:   255.255.255.192 = 26  11111111.11111111.11111111.11 000000
Wildcard:  0.0.0.63              00000000.00000000.00000000.00 111111
=>
Network:   192.168.2.128/26      11000000.10101000.00000010.10 000000
HostMin:   192.168.2.129         11000000.10101000.00000010.10 000001
HostMax:   192.168.2.190         11000000.10101000.00000010.10 111110
Broadcast: 192.168.2.191         11000000.10101000.00000010.10 111111
Hosts/Net: 62                     Class C, Private Internet

administrator@debian:~$ 
```

Given prefix is the same as the Network number

therefore, the given prefix is a properly aligned prefix

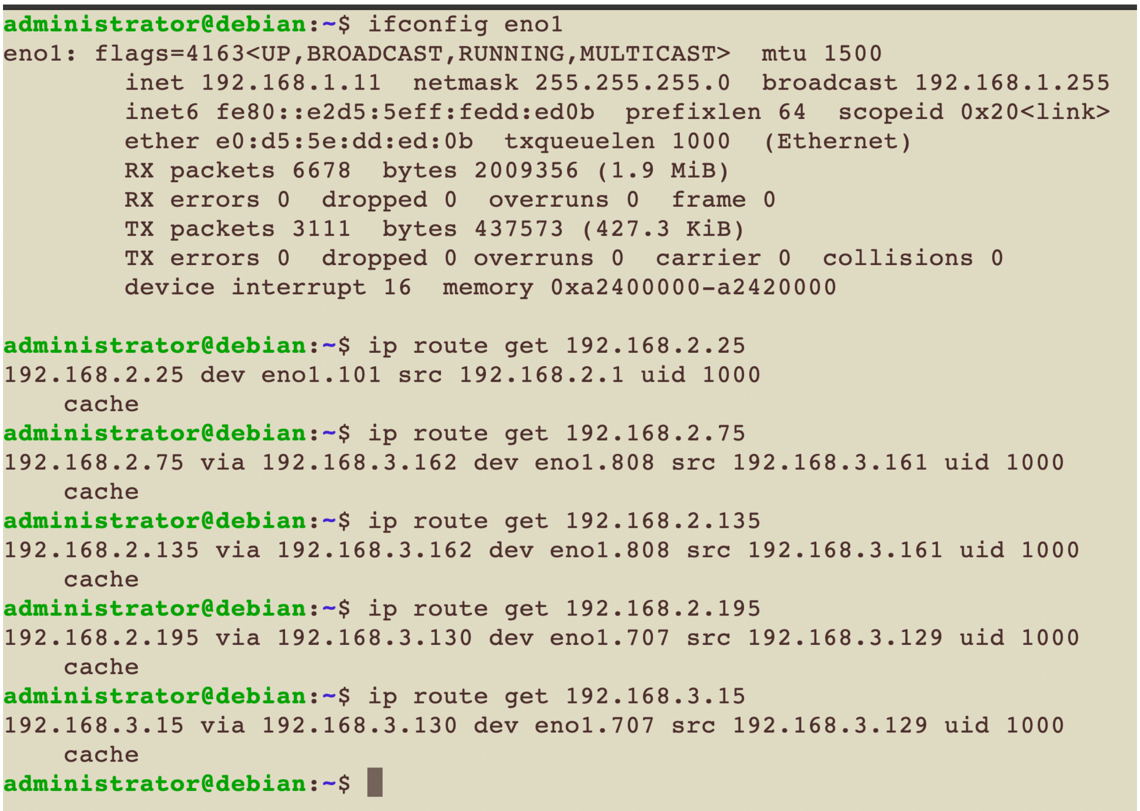
Figure 3. Checking that a prefix is correctly aligned by using ipcalc

Exercise 3: Check bidirectional connectivity to router's directly connected routers

Remotely (ssh) to one router (R_x) of your choice. Use ping for checking each of R_x direct physical connections to other routers. Notice, you'll have to fetch the relevant IP addresses by looking them up in the tables above. Take into account the numbering conventions as they affect the least significant digit in a direct link (The least-prefix Router always must receive the least value for the least significant digit of the IP address). Include all of the results that you have obtained, below:

Exercise 4: Checking the routes to all of the Stub networks

In the present week, we'll not connect any hosts to our internetwork. That we'll do next week. Nevrtheless, we wish to verify the integrity of the routing tables resulting from the network configuration entered into /etc/network/interfaces. To that purpose, we'll use an option of the Linux ip command that allows us to simulate the execution of the IP FWD algorithm (Longest Prefix Matching, LPM).



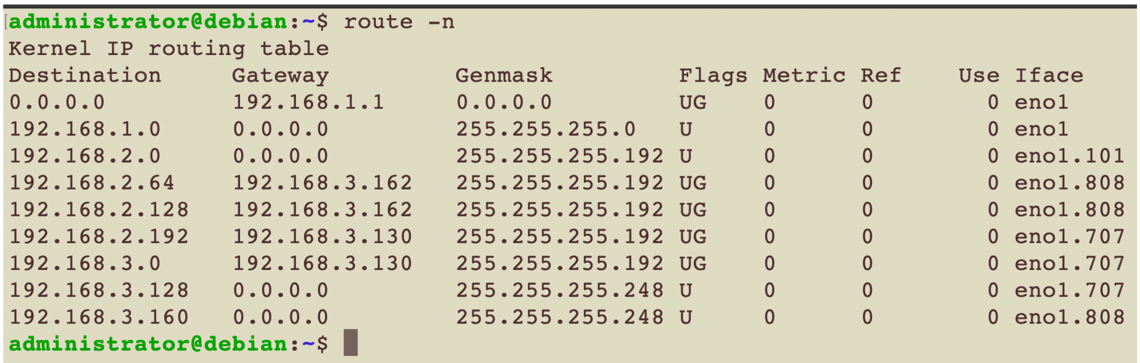
```
administrator@debian:~$ ifconfig eno1
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.11  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e2d5:5eff:fedd:ed0b  prefixlen 64  scopeid 0x20<link>
        ether e0:d5:5e:dd:ed:0b  txqueuelen 1000  (Ethernet)
        RX packets 6678  bytes 2009356 (1.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3111  bytes 437573 (427.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xa2400000-a2420000

administrator@debian:~$ ip route get 192.168.2.25
192.168.2.25 dev eno1.101 src 192.168.2.1 uid 1000
    cache
administrator@debian:~$ ip route get 192.168.2.75
192.168.2.75 via 192.168.3.162 dev eno1.808 src 192.168.3.161 uid 1000
    cache
administrator@debian:~$ ip route get 192.168.2.135
192.168.2.135 via 192.168.3.162 dev eno1.808 src 192.168.3.161 uid 1000
    cache
administrator@debian:~$ ip route get 192.168.2.195
192.168.2.195 via 192.168.3.130 dev eno1.707 src 192.168.3.129 uid 1000
    cache
administrator@debian:~$ ip route get 192.168.3.15
192.168.3.15 via 192.168.3.130 dev eno1.707 src 192.168.3.129 uid 1000
    cache
administrator@debian:~$
```

Figure 4. Checking routes computed by LPM at router R_1

a) Interpret the results printed out by the execution of the command $ ip route get 192.168.3.15 (See fig. 4).

b) *[Next week]* Apply the LPM algorithm to router R_1 forwarding tables assuming that an IP packet ingresses in that router which destination IP is 192.168.3.15 (Forwarding tables are included below)



```
administrator@debian:~$ 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    0      0        0 eno1
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eno1
192.168.2.0     0.0.0.0         255.255.255.192 U     0      0        0 eno1.101
192.168.2.64    192.168.3.162   255.255.255.192 UG    0      0        0 eno1.808
192.168.2.128   192.168.3.162   255.255.255.192 UG    0      0        0 eno1.808
192.168.2.192   192.168.3.130   255.255.255.192 UG    0      0        0 eno1.707
192.168.3.0     192.168.3.130   255.255.255.192 UG    0      0        0 eno1.707
192.168.3.128   0.0.0.0         255.255.255.248 U     0      0        0 eno1.707
192.168.3.160   0.0.0.0         255.255.255.248 U     0      0        0 eno1.808
administrator@debian:~$
```

Figure 5. FDB_1 (Forwarding DataBase of R_1) or Forwarding Table of R_1