Túneles de IP sobre IP (v4)

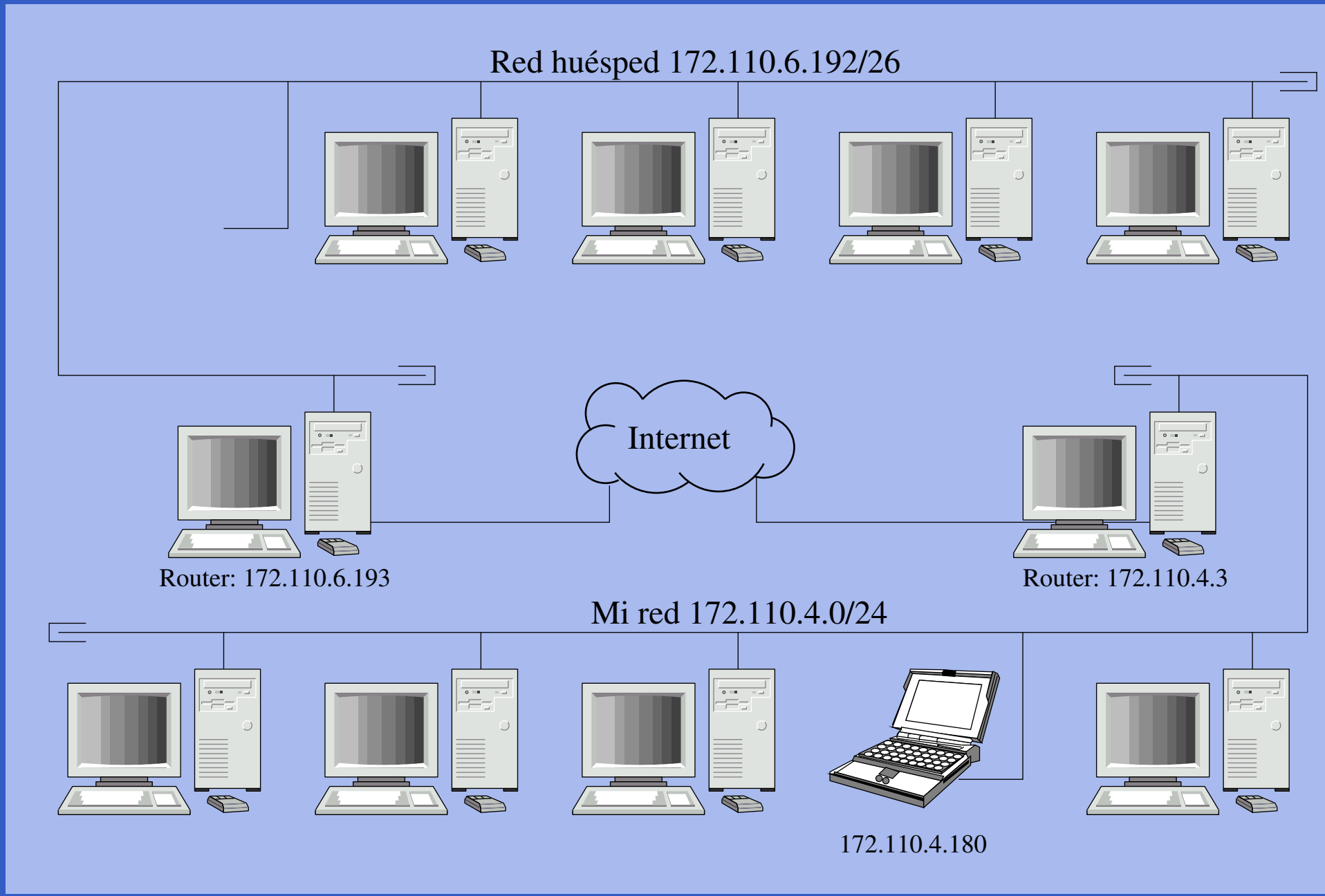
Movilidad sobre IP

Federico Bareilles

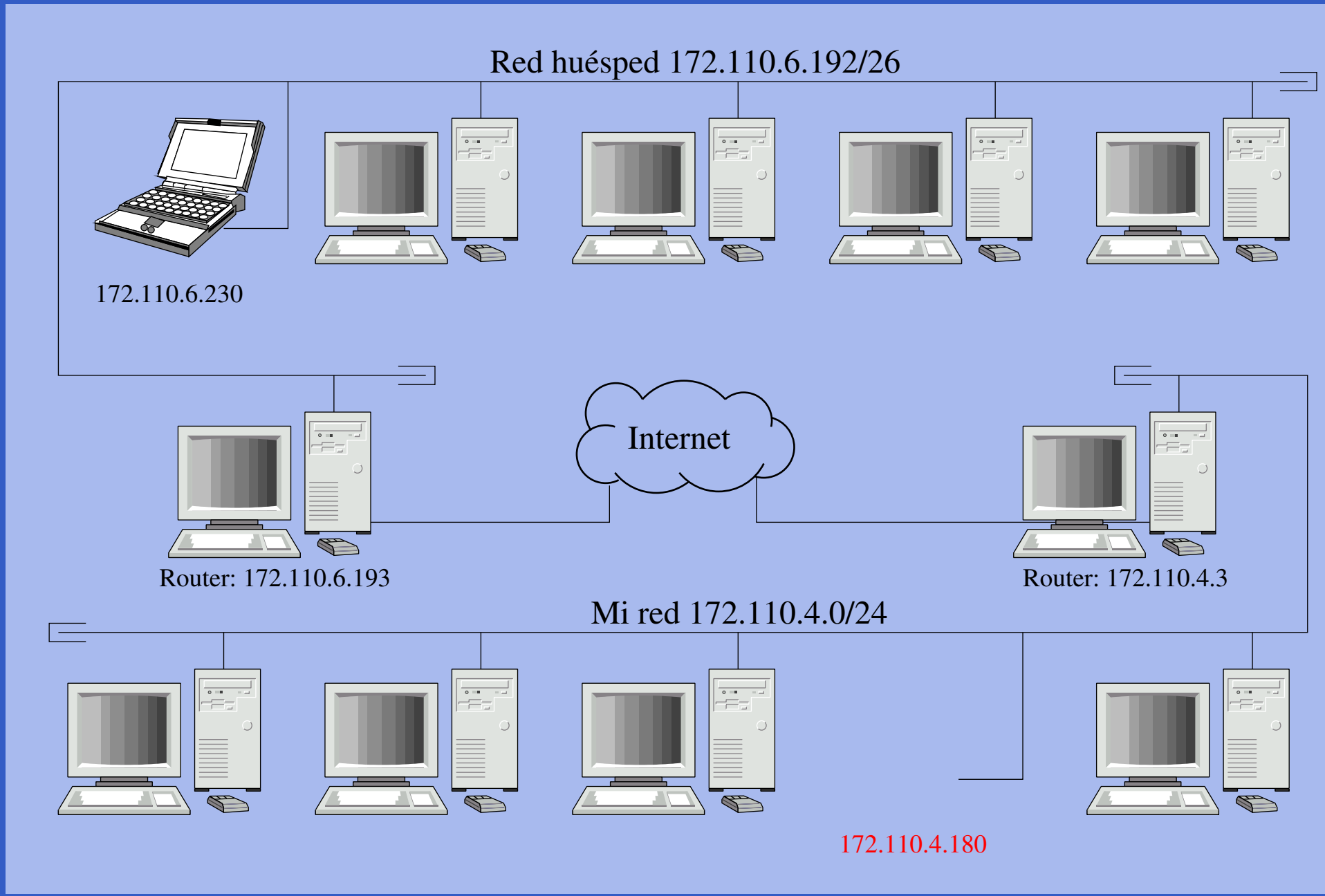
BALUG

Balug Argentina Linux User Group

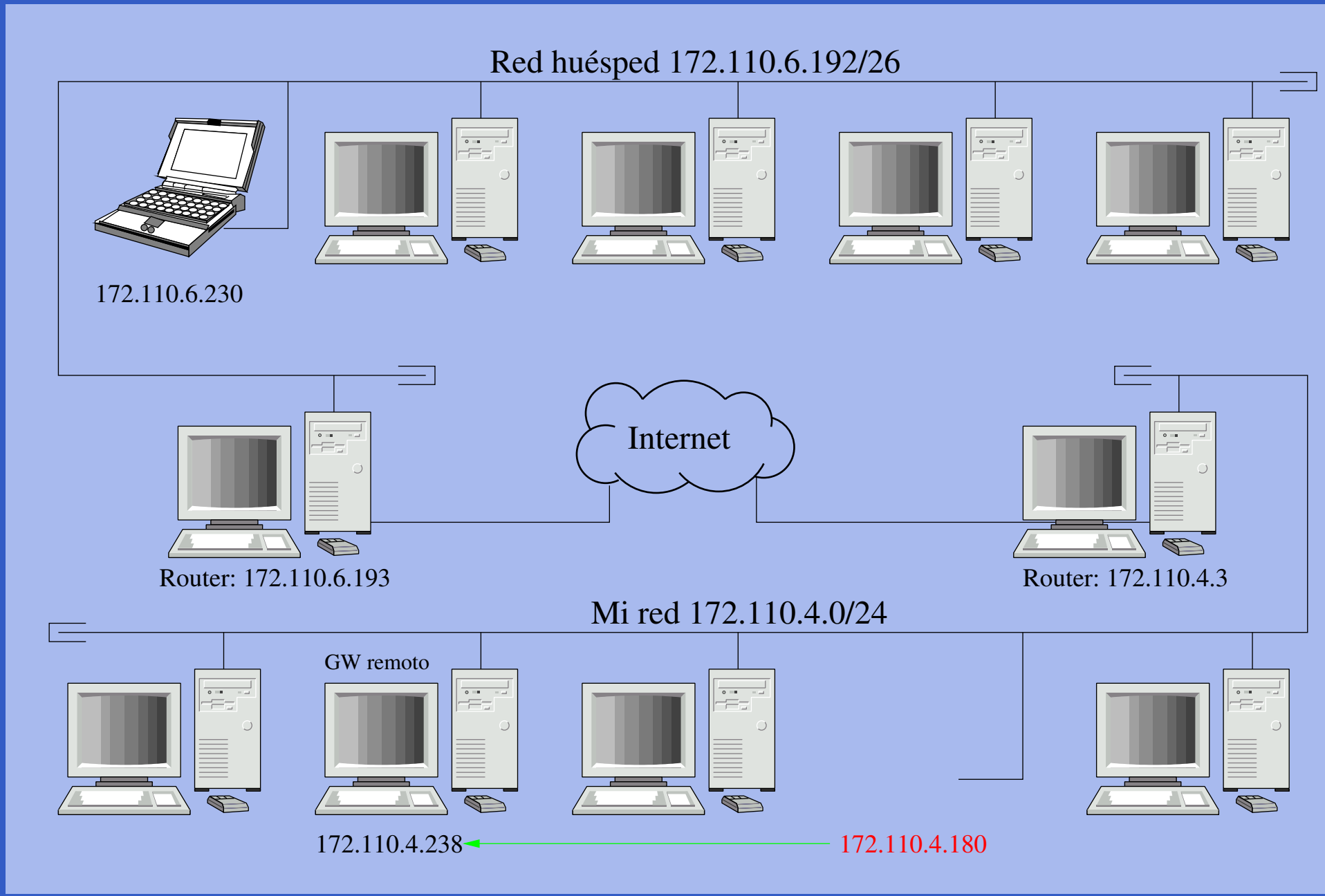
¿Qué se quiere resolver?



¿Qué se quiere resolver?



¿Qué se quiere resolver?



Programejos

Lo único que necesitamos:

- ip - Linux IPv4 protocol implementation.
  Es el reemplazo de *route*, *ifconfig*, *arp* y algún otro.
- ipip en el kernel ($\geq$ 2.2.x):

```
# Networking options
CONFIG_INET=y
CONFIG_NETLINK_DEV=y
CONFIG_NETFILTER=y
CONFIG_IP_ADVANCED_ROUTER=y
CONFIG_NET_IPIP=m
CONFIG_NET_IPGRE=m
```

- iptables (opcional con kernel ($\geq$ 2.4.x)

GW Remoto (el lado fácil)

Creamos el túnel:

```
# ip tunnel add mode ipip tunl1 local 172.110.4.238 remote 172.110.6.230
# ip addr add dev tunl1 local 192.168.0.10 peer 192.168.0.20
# ip link set tunl1 up
```

GW Remoto (el lado fácil)

Creamos el túnel:

```
# ip tunnel add mode ipip tunl1 local 172.110.4.238 remote 172.110.6.230
# ip addr add dev tunl1 local 192.168.0.10 peer 192.168.0.20
# ip link set tunl1 up
```

¿Qué *ruteamos* por ahí?

```
# ip route add 172.110.4.180 dev tunl1
# ip route flush cache
```

GW Remoto (el lado fácil)

Creamos el túnel:

ip tunnel add mode ipip tunl1 local 172.110.4.238 remote 172.110.6.230

ip addr add dev tunl1 local 192.168.0.10 peer 192.168.0.20

ip link set tunl1 up

¿Qué *ruteamos* por ahí?

ip route add 172.110.4.180 dev tunl1

ip route flush cache

La máquina remota necesita contestar los *arp* del IP que va a *rutear*:

ip neigh add proxy 172.110.4.180 dev eth0

GW Remoto (el lado fácil)

Creamos el túnel:

ip tunnel add mode ipip tunl1 local 172.110.4.238 remote 172.110.6.230

ip addr add dev tunl1 local 192.168.0.10 peer 192.168.0.20

ip link set tunl1 up

¿Qué *ruteamos* por ahí?

ip route add 172.110.4.180 dev tunl1

ip route flush cache

La máquina remota necesita contestar los *arp* del IP que va a *rutear*:

ip neigh add proxy 172.110.4.180 dev eth0

Finalmente *Forwardear*:

echo "1"> /proc/sys/net/ipv4/ip_forward

¿Qué muestra *ifconfig*?

```
# ifconfig
eth0    Link encap:Ethernet  HWaddr 00:00:E2:2A:05:A0
        inet addr:172.110.4.238  Bcast:172.110.4.255  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:408879 errors:0 dropped:0 overruns:0 frame:2
        TX packets:341976 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100
        Interrupt:20

tunl1   Link encap:IPIP Tunnel  HWaddr
        inet addr:192.168.0.10  P-t-P:192.168.0.20  Mask:255.255.255.255
        UP POINTOPOINT RUNNING NOARP  MTU:1480  Metric:1
        RX packets:906 errors:0 dropped:0 overruns:0 frame:0
        TX packets:843 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
```

Pero *ifconfig* ya está obsoleto, mejor veamos que muestra ip...

¿Qué muestra *ip*?

```
# ip add
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:00:e2:2a:05:a0 brd ff:ff:ff:ff:ff:ff
    inet 172.110.4.238/24 brd 172.110.4.255 scope global eth0
5: tunl1@NONE: <POINTOPOINT,NOARP,UP> mtu 1480 qdisc noqueue
    link/ipip 172.110.4.238 peer 172.110.6.230
    inet 192.168.0.10 peer 192.168.0.20/32 scope global tunl1

# ip tunnel show tunl1
tunl1: ip/ip  remote 172.110.6.230  local 172.110.4.238  ttl inherit

# ip route
192.168.0.20 dev tunl1  proto kernel  scope link  src 192.168.0.10
172.110.4.180 dev tunl1  scope link
172.110.4.0/24 dev eth0  proto kernel  scope link  src 172.110.4.238
127.0.0.0/8 dev lo  scope link
default via 172.110.4.3 dev eth0
```

Lado local (el móvil)

Creamos el túnel:

```
# ip tunnel add mode ipip tunl1 local 172.110.6.230 remote 172.110.4.238
# ip addr add dev tunl1 local 192.168.0.20 peer 192.168.0.10
# ip link set tunl1 up
```

Lado local (el móvil)

Creamos el túnel:

ip tunnel add mode ipip tunl1 local 172.110.6.230 remote 172.110.4.238

ip addr add dev tunl1 local 192.168.0.20 peer 192.168.0.10

ip link set tunl1 up

Agregamos el número IP (exportado) a la interfase tunl1:

ip addr add 172.110.4.180 dev tunl1

Lado local (el móvil)

Creamos el túnel:

ip tunnel add mode ipip tunl1 local 172.110.6.230 remote 172.110.4.238

ip addr add dev tunl1 local 192.168.0.20 peer 192.168.0.10

ip link set tunl1 up

Agregamos el número IP (exportado) a la interfase tunl1:

ip addr add 172.110.4.180 dev tunl1

Si queremos *rutear* todo por el túnel, es fácil; mejor *rutiemos* por el túnel lo que viene por el túnel:

echo 200 mi_tunel >> /etc/iproute2/rt_tables (por única vez)

ip rule add from 172.110.4.180 table mi_tunel

ip route add default via 192.168.0.10 dev tunl1 table mi_tunel

ip route flush cache

Lado local (el móvil)

```
# ip rule
0:          from all lookup local
32765:      from 172.110.4.180 lookup mi_tunel
32766:      from all lookup main
32767:      from all lookup 253
```

Lado local (el móvil)

```
# ip rule
0:          from all lookup local
32765:      from 172.110.4.180 lookup mi_tunel
32766:      from all lookup main
32767:      from all lookup 253

# ip route (la tabla clásica)
192.168.0.10 dev tunl1 proto kernel scope link src 192.168.0.20
172.110.6.192/26 dev eth0 proto kernel scope link src 172.110.6.230
127.0.0.0/8 dev lo scope link
default via 172.110.6.193 dev eth0
```

Lado local (el móvil)

```
# ip rule
0:          from all lookup local
32765:      from 172.110.4.180 lookup mi_tunel
32766:      from all lookup main
32767:      from all lookup 253

# ip route (la tabla clásica)
192.168.0.10 dev tunl1 proto kernel scope link src 192.168.0.20
172.110.6.192/26 dev eth0 proto kernel scope link src 172.110.6.230
127.0.0.0/8 dev lo scope link
default via 172.110.6.193 dev eth0

# ip route list table mi_tunel
default via 192.168.0.20 dev tunl1
```

Un *ruteo* más interesante (marcando paquetes)

Podemos pretender que todo el correo saliente se *rutee* por el túnel:

- Marcamos los paquetes que tienen como destino SMTP (25)
  # iptables −t mangle −A PREROUTING −p tcp −−dport 25 −j MARK −−set−mark 1

Un *ruteo* más interesante (marcando paquetes)

Podemos pretender que todo el correo saliente se *rutee* por el túnel:

- Marcamos los paquetes que tienen como destino SMTP (25)
  # iptables −t mangle −A PREROUTING −p tcp −−dport 25 −j MARK −−set−mark 1
- Agregamos una regla para dirigir los paquetes marcados con “1” por la tabla mi_tunel
  # ip rule add fwmark 1 table mi_tunel

Un *ruteo* más interesante (marcando paquetes)

Podemos pretender que todo el correo saliente se *rutee* por el túnel:

- Marcamos los paquetes que tienen como destino SMTP (25)
  # iptables −t mangle −A PREROUTING −p tcp −−dport 25 −j MARK −−set−mark 1
- Agregamos una regla para dirigir los paquetes marcados con “1” por la tabla mi_tunel
  # ip rule add fwmark 1 table mi_tunel
- La ruta ya la agregamos :-)

Un *ruteo* más interesante

Agreguemos el *ruteo* a *mi red* (172.110.4.0/24) por el túnel:

- Fijamos la ruta al GW remoto:
  # ip route add 172.110.4.238 via 172.110.6.193

Un *ruteo* más interesante

Agreguemos el *ruteo* a *mi red* (172.110.4.0/24) por el túnel:

- Fijamos la ruta al GW remoto:
  # ip route add 172.110.4.238 via 172.110.6.193
- Luego, toda *mi red* por el túnel. Pero ¿ Cómo?...

Un *ruteo* más interesante

Agreguemos el *ruteo* a *mi red* (172.110.4.0/24) por el túnel:

- Fijamos la ruta al GW remoto:
  # ip route add 172.110.4.238 via 172.110.6.193
- Luego, toda *mi red* por el túnel. Pero ¿ Cómo?...
  - ¿Agregando la ruta?
    # ip route add 172.110.4.0/24 dev tunl1

Un *ruteo* más interesante

Agreguemos el *ruteo* a *mi red* (172.110.4.0/24) por el túnel:

- Fijamos la ruta al GW remoto:
  # ip route add 172.110.4.238 via 172.110.6.193
- Luego, toda *mi red* por el túnel. Pero ¿ Cómo?...
  - ¿Agregando la ruta?
    # ip route add 172.110.4.0/24 dev tunl1
    ¡¡¡NO!!!

Un *ruteo* más interesante

Agreguemos el *ruteo* a *mi red* (172.110.4.0/24) por el túnel:

- Fijamos la ruta al GW remoto:
  # ip route add 172.110.4.238 via 172.110.6.193
- Luego, toda *mi red* por el túnel. Pero ¿ Cómo?...
  - ¿Agregando la ruta?
    # ip route add 172.110.4.0/24 dev tunl1
    ¡¡¡NO!!!
  - Modificando el paso donde agregábamos el IP al túnel:
    # ip addr del 172.110.4.180 dev tunl1
    y agregando el IP con toda la red
    # ip addr add 172.110.4.180/24 dev tunl1
  - ó simplemente:
    # ip route add 172.110.4.0/24 dev tunl1 src 172.110.4.180
    con esto agregamos la ruta y el IP que se publica.

¿Qué muestra *ip*?

Esto es con una conección telefónica:

```
# ip addr
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
3: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qdisc pfifo_fa
    link/ppp
    inet 200.32.17.74 peer 200.32.1.234/32 scope global ppp0
4: tunl0@NONE: <NOARP> mtu 1480 qdisc noop
    link/ipip 0.0.0.0 brd 0.0.0.0
5: tunl1@NONE: <POINTOPOINT,NOARP,UP> mtu 1480 qdisc noqueue
    link/ipip 200.32.17.74 peer 172.110.4.238
    inet 192.168.15.180 peer 192.168.15.238/32 scope global tunl
    inet 172.110.4.180/24 scope global tunl1
```

¿Qué muestra *ip*?

Esto es con una conección telefónica:

```
# ip tun
tunl0: ip/ip  remote any  local any  ttl inherit  nopmtudisc
tunl1: ip/ip  remote 172.110.4.238  local 200.32.17.74  ttl inhe

# ip route
200.32.1.234 dev ppp0  proto kernel  scope link  src 200.32.17.7
192.168.15.238 dev tunl1  proto kernel  scope link  src 192.168.
172.110.4.238 via 200.32.1.234 dev ppp0
172.110.4.0/24 dev tunl1  proto kernel  scope link  src 172.110.
127.0.0.0/8 dev lo  scope link
default via 200.32.1.234 dev ppp0

# ip route show table mi_tunel
default via 192.168.15.238 dev tunl1
```

Mail con movilidad:

.procmailrc en el server (mail.some.domain.ar):

```
:0
* !^From.*fede
! `ping -nc 1 -w 2 172.110.4.180|awk '/^64/{ALIVE=1}\
        END{ if(ALIVE) print " fede@libertad.some.domain.ar";\
             else print " fede@mail.some.domain.ar";}'`
```

.fetchmailrc (local):

```
defaults
        user fede is fede
        no keep
        fetchall
        no rewrite

poll localhost with protocol pop3 and port 2266:
        preconnect "ssh -C -f fede@172.110.4.152 \
                    -L 2266:172.110.4.152:110 sleep 10";
```

Referencias:

Docs:

- Redundant Internet Connections Using Linux (Sys Admin): http://www.samag.com/print/documentID=20295
- IP Command Reference: http://defiant.coinet.com/iproute2/ip-cref/
- iptables Tutorial: http://people.unix-fu.org/andreasson/iptables-tutorial/
- Adv-Routing-HOWTO.pdf: http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/

Soft:

- iproute: ftp://ftp.inr.ac.ru/ip-routing/
- netfilter: http://www.netfilter.org/
- iptables: http://www.iptables.org/
- Las macros de LaTeX que usé en esto: http://prosper.sourceforge.net/

ping...

```
[fede@libertad fede]$ ping -n mail
PING mail.some.domain.ar (172.110.4.152) from 192.168.0.20 :
56(84) bytes of data.

--- mail.some.domain.ar ping statistics ---
65 packets transmitted, 0 packets received, 100% packet loss
[fede@libertad fede]$
```

ping...

```
[fede@libertad fede]$ ping -n mail
PING mail.some.domain.ar (172.110.4.152) from 172.110.4.180 :
56(84) bytes of data.
64 bytes from 172.110.4.152: icmp_seq=1 ttl=244 time=1.719 sec
64 bytes from 172.110.4.152: icmp_seq=2 ttl=244 time=1.189 sec
64 bytes from 172.110.4.152: icmp_seq=3 ttl=244 time=969.580 mse
64 bytes from 172.110.4.152: icmp_seq=4 ttl=244 time=199.596 mse

--- mail.some.domain.ar ping statistics ---
5 packets transmitted, 4 packets received, 20% packet loss
round-trip min/avg/max/mdev = 199.596/1019.568/1719.560/546.290 m
```