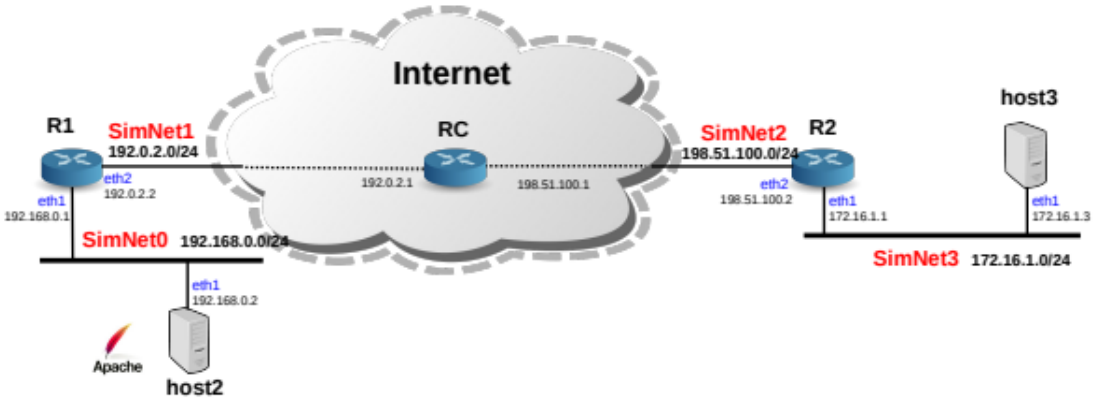
P9: IP Tunnels



0.3 starting the scenario using simctl

//from terminal
simctl iptunnel sh
start

```
Virtual machines from iptunnel:
num     vms         enabled tty's Id
  1     RC          0
  2     R2          0
  3     host3       0
  4     R1          0
  5     host2       0
```

0.4 Configuring IPIP tunnels

//from R2
ip tunnel add tunnel0 mode ipip local 198.51.100.2 remote 192.0.2.2 ttl 0 nopmtudisc dev eth2 #configuring the tunnel
ifconfig tunnel0 1.2.3.4 #activando la nueva interfaz virtual
route add -net 192.168.0.0/24 dev tunnel0 #actualizar tabla de ruta

```
R2:~# ip tunnel add tunnel0 mode ipip local 198.51.100.2 remote 192.0.2.2 ttl 0
nopmtudisc dev eth2
R2:~# ifconfig tunnel0 1.2.3.4
R2:~# route add -net 192.168.0.0/24 dev tunnel0
```

//from R1
ip tunnel add tunnel0 mode ipip local 192.0.2.2 remote 198.51.100.2 ttl 0 nopmtudisc dev eth2
ifconfig tunnel0 1.2.3.4
route add -net 172.16.1.0/24 dev tunnel0

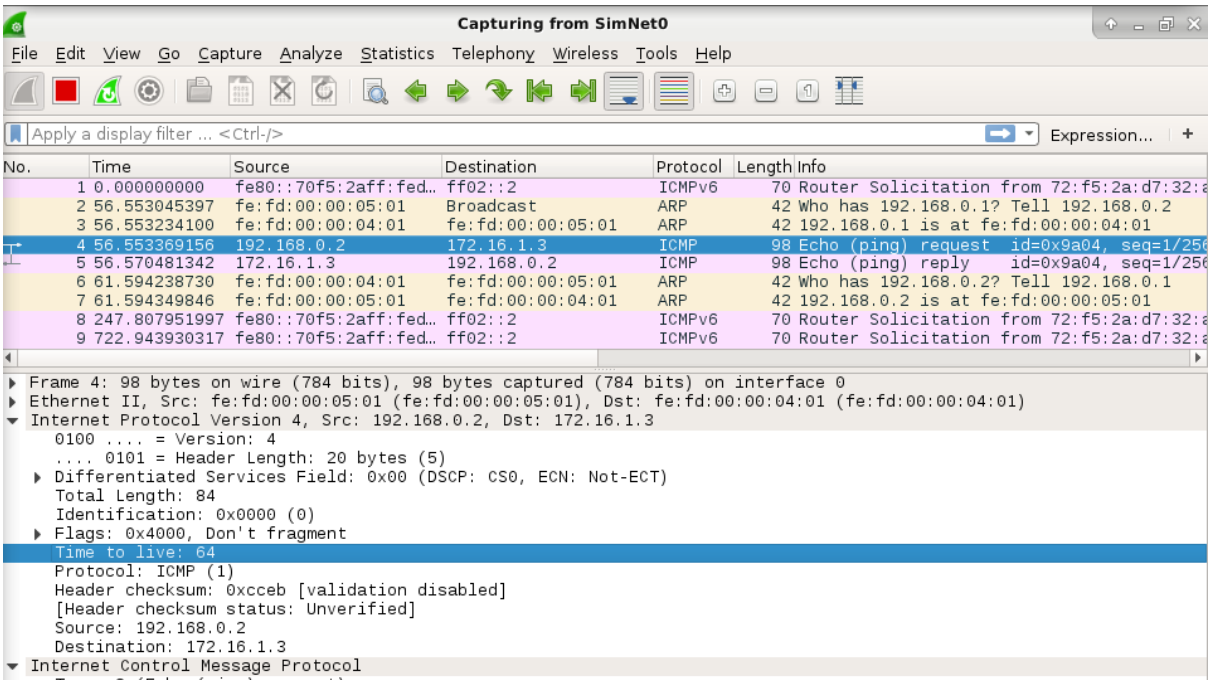
0.5 Testing the tunnels: protocol and TTL fields

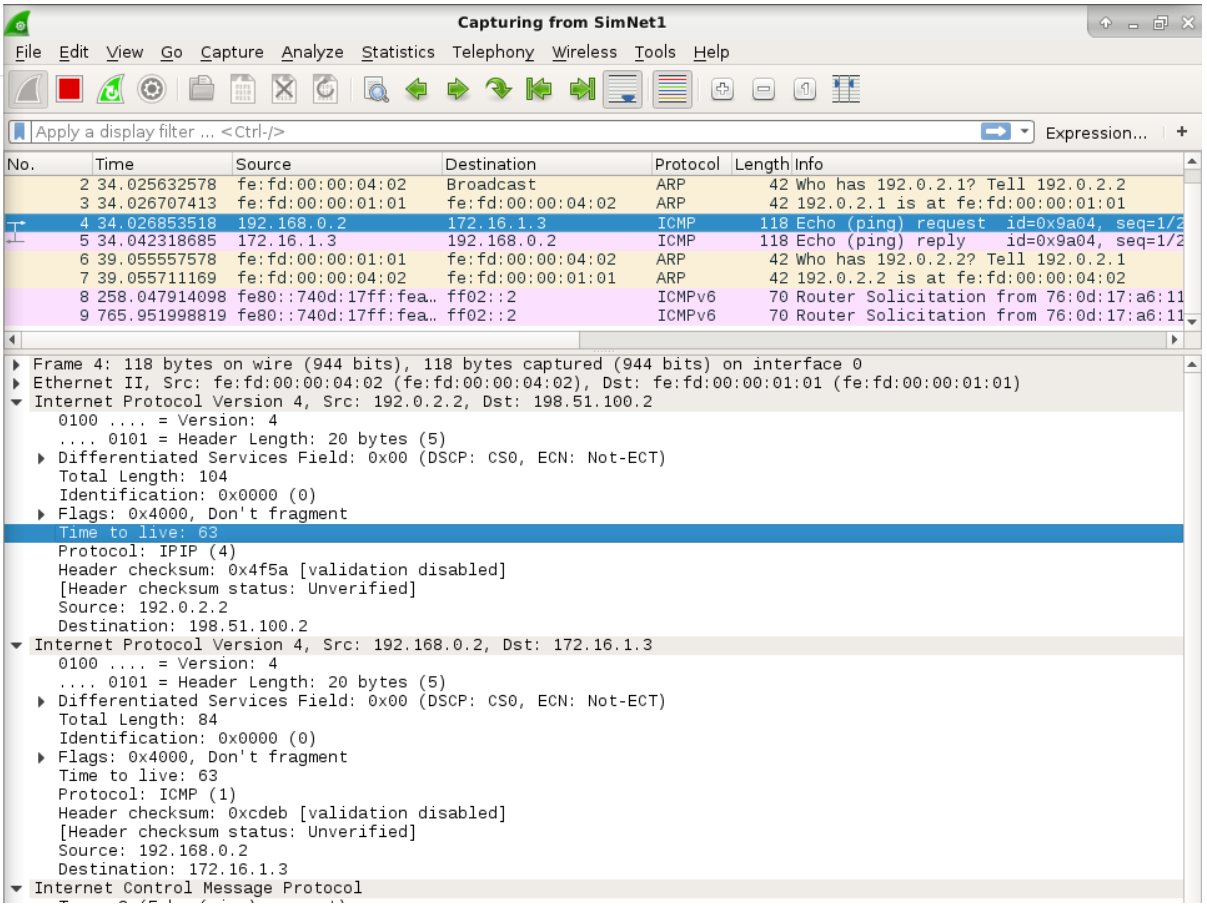
1. Open FOUR wireshark network analyzers in the PHYSICAL HOST and capture traffic in each of these networks: SimNet0, SimNet1, SimNet2 and SimNet3.
To properly see fragmented messages, DISABLE the following option in all these wiresharks:Edit-Preferences-Protocols-IPv4-Reassenble fragmented IPv4 datagrams.

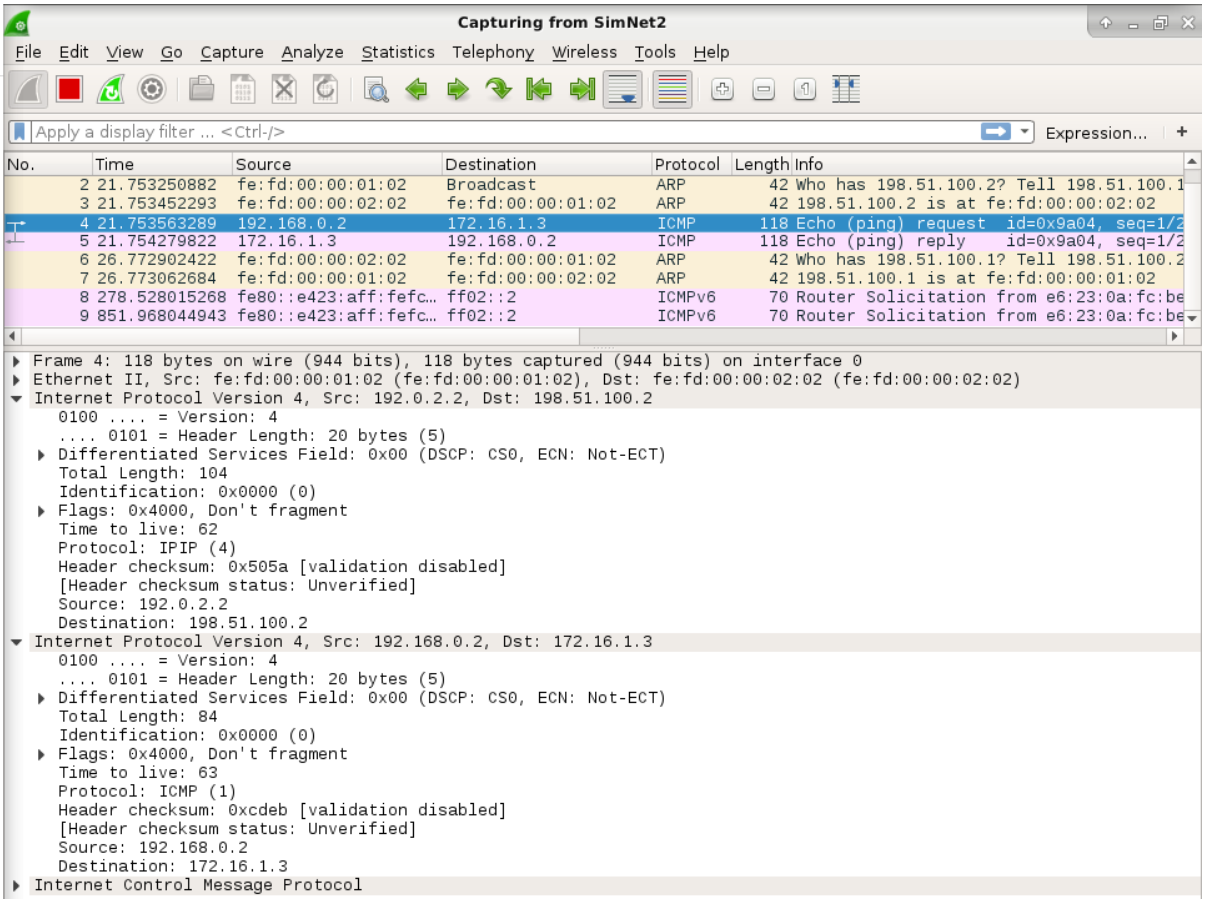
2. Send the following ping from host2 to host3:

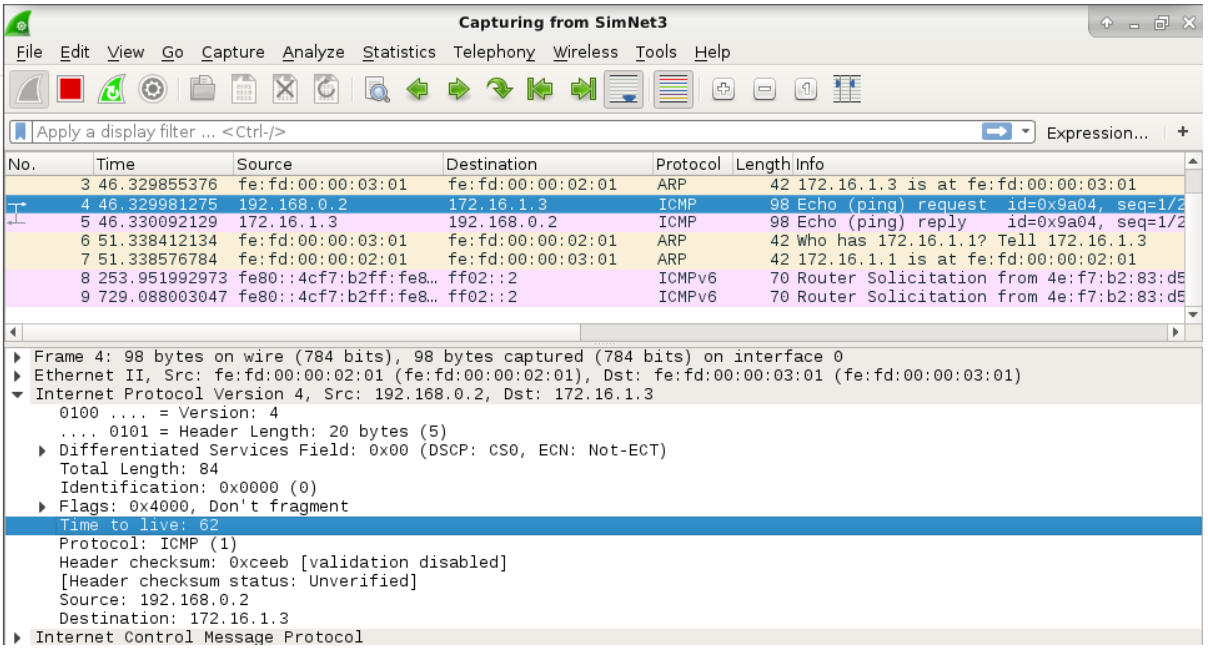
//from host2
ping -c1 172.16.1.3

```
host2:~# ping -c1 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 56(84) bytes of data.
64 bytes from 172.16.1.3: icmp_seq=1 ttl=62 time=38.1 ms

--- 172.16.1.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 38.183/38.183/38.183/0.000 ms
```




Capturing from SimNet1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
2 34.025632578 fe:fd:00:00:04:02 Broadcast ARP 42 Who has 192.0.2.1? Tell 192.0.2.2
3 34.026707413 fe:fd:00:00:01:01 fe:fd:00:00:04:02 ARP 42 192.0.2.1 is at fe:fd:00:00:01:01
4 34.026853518 192.168.0.2 172.16.1.3 ICMP 118 Echo (ping) request id=0x9a04, seq=1/2
5 34.042318685 172.16.1.3 192.168.0.2 ICMP 118 Echo (ping) reply id=0x9a04, seq=1/2
6 39.055557578 fe:fd:00:00:01:01 fe:fd:00:00:04:02 ARP 42 Who has 192.0.2.2? Tell 192.0.2.1
7 39.055711169 fe:fd:00:00:04:02 fe:fd:00:00:01:01 ARP 42 192.0.2.2 is at fe:fd:00:00:04:02
8 258.047914098 fe80::740d:17ff:fea... ff02::2 ICMPv6 70 Router Solicitation from 76:0d:17:a6:11
9 765.951998819 fe80::740d:17ff:fea... ff02::2 ICMPv6 70 Router Solicitation from 76:0d:17:a6:11
Frame 4: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
Ethernet II, Src: fe:fd:00:00:04:02 (fe:fd:00:00:04:02), Dst: fe:fd:00:00:01:01 (fe:fd:00:00:01:01)
Internet Protocol Version 4, Src: 192.0.2.2, Dst: 198.51.100.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 104
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
Time to live: 63
Protocol: IPIP (4)
Header checksum: 0x4f5a [validation disabled]
[Header checksum status: Unverified]
Source: 192.0.2.2
Destination: 198.51.100.2
Internet Protocol Version 4, Src: 192.168.0.2, Dst: 172.16.1.3
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 84
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
Time to live: 63
Protocol: ICMP (1)
Header checksum: 0xcdeb [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.0.2
Destination: 172.16.1.3
Internet Control Message Protocol


Capturing from SimNet2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
2 21.753250882 fe:fd:00:00:01:02 Broadcast ARP 42 Who has 198.51.100.2? Tell 198.51.100.1
3 21.753452293 fe:fd:00:00:02:02 fe:fd:00:00:01:02 ARP 42 198.51.100.2 is at fe:fd:00:00:02:02
4 21.753563289 192.168.0.2 172.16.1.3 ICMP 118 Echo (ping) request id=0x9a04, seq=1/2
5 21.754279822 172.16.1.3 192.168.0.2 ICMP 118 Echo (ping) reply id=0x9a04, seq=1/2
6 26.772902422 fe:fd:00:00:02:02 fe:fd:00:00:01:02 ARP 42 Who has 198.51.100.1? Tell 198.51.100.2
7 26.773062684 fe:fd:00:00:01:02 fe:fd:00:00:02:02 ARP 42 198.51.100.1 is at fe:fd:00:00:01:02
8 278.528015268 fe80::e423:aff:fefc... ff02::2 ICMPv6 70 Router Solicitation from e6:23:0a:fc:be
9 851.968044943 fe80::e423:aff:fefc... ff02::2 ICMPv6 70 Router Solicitation from e6:23:0a:fc:be
Frame 4: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
Ethernet II, Src: fe:fd:00:00:01:02 (fe:fd:00:00:01:02), Dst: fe:fd:00:00:02:02 (fe:fd:00:00:02:02)
Internet Protocol Version 4, Src: 192.0.2.2, Dst: 198.51.100.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 104
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
Time to live: 62
Protocol: IPIP (4)
Header checksum: 0x505a [validation disabled]
[Header checksum status: Unverified]
Source: 192.0.2.2
Destination: 198.51.100.2
Internet Protocol Version 4, Src: 192.168.0.2, Dst: 172.16.1.3
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 84
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
Time to live: 63
Protocol: ICMP (1)
Header checksum: 0xcdeb [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.0.2
Destination: 172.16.1.3
Internet Control Message Protocol

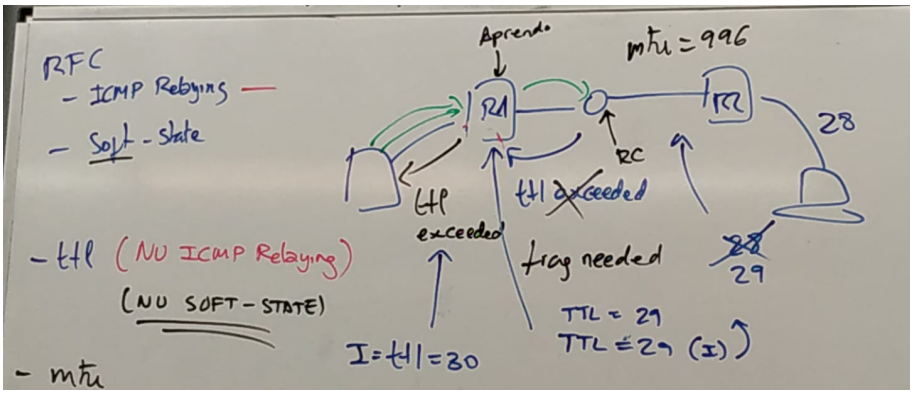


(a) What is the meaning of the -c option? (consult the man page of ping if necessary)
para enviar pings finitos. -c 2 envias 2 pings.
(b) Is the ping working? How many packets can you see?
si, 1 pq hemos hecho ping de 1 paquete.

3. Assuming that the ping is working, it is necessary to emphasize certain aspects related to tunneling issues:
(a) Verify that the IP packet in SimNet1 and SimNet2 is encapsulated using IPIP. How many IP headers can you see in these packets? Which is the value of the “Protocol” field in the Outer Header?
Dentro del outer header de los paquetes de capturados en SimNet1 y 2 podemos ver un campo Protocol: IPIP. Dentro de estos paquetes hay dos headers IP, el outer y el inner header (exterior e interior). En las inner header el protocolo obviamente es ICMP ya que estamos encapsulando un echo-request/reply.
EL valor de protocolo es IPIP(4)

(b) Determine the difference in size from the original IP packet (in SimNet0) and the encapsulted one (in SimNet1).
El paquete IP original tiene un tamaño de 84 bytes mientras que el encapsulado es de 104 bytes, 20 bytes más correspondientes a la outer header.

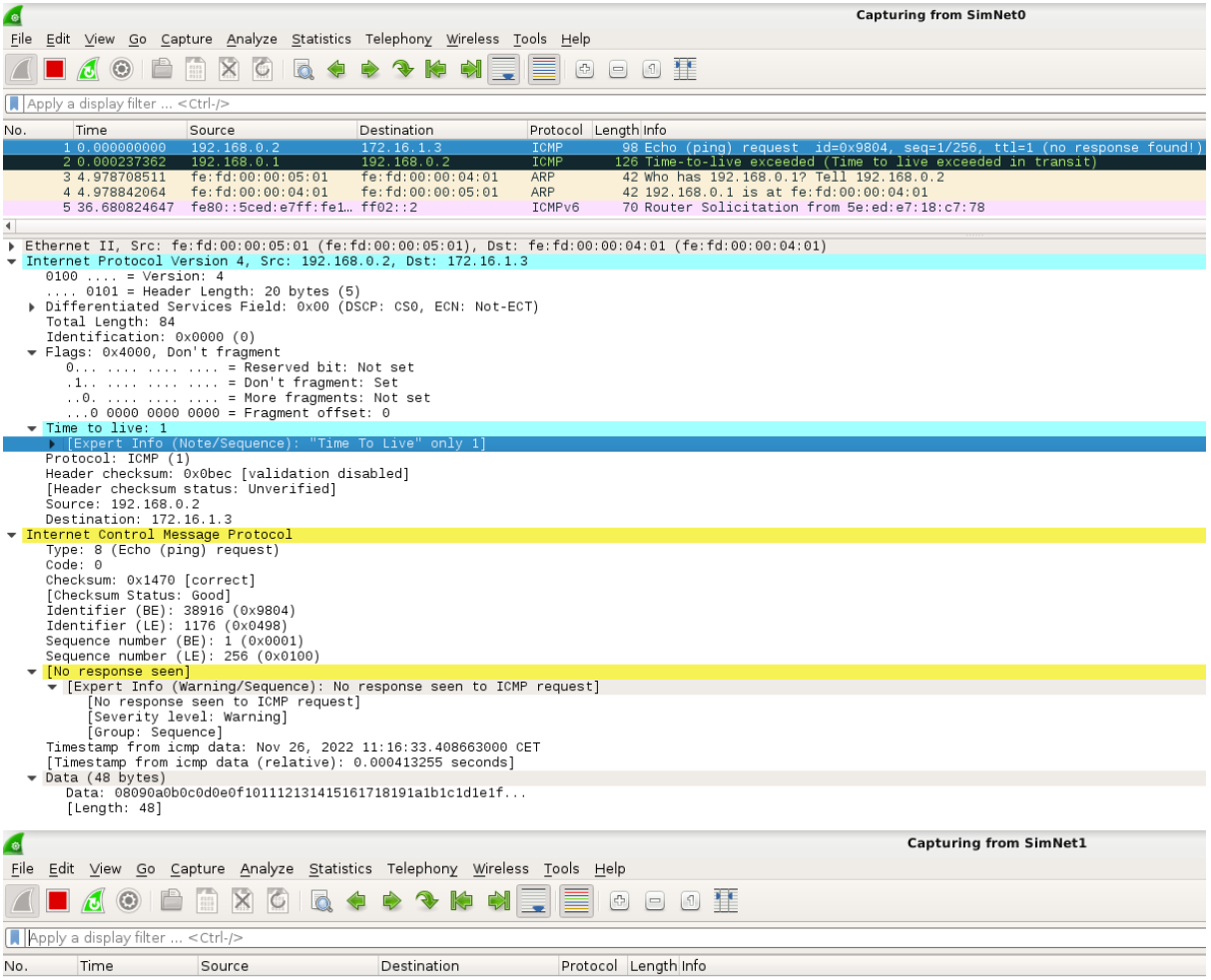
(c) Identify the value of the “TTL” field in the original IP packet and how this value evolve during the transmission, both in the inner and in the outer headers. Remind that we used the option ttl inherit during the creation of the tunnels.



Siguiendo este esquema(mirar lo azul), en nuestro caso, al inicio tenemos ttl 64, cuando pasa al R1 tenemos ttl= 63. Este decremento es debido a que el router ha de consultar su tabla de enrutamiento, entonces se decrementa 1 ttl, pq solo hace 1 salto. Y como ahí se empieza a hacer la encapsulacion, el protocolo IPIPI copia el ttl con ese valor. Al llegar al R2, router destino, como ha de consultar tambien su tabla de enrutamiento de decrementa 1 , ttl=62. Pero solo se decrementa el de outer header, el inner header se queda igual, ttl=63, porque en los tunneles, el inner header no se modifica. y Despues como ha de ir de R2 al destino hace un salto mas, entonces se decrementa 1 ttl de inner head.
Al final, vemos que tenemos 2 saltos, empezamos con 64 y acabamos con 62.

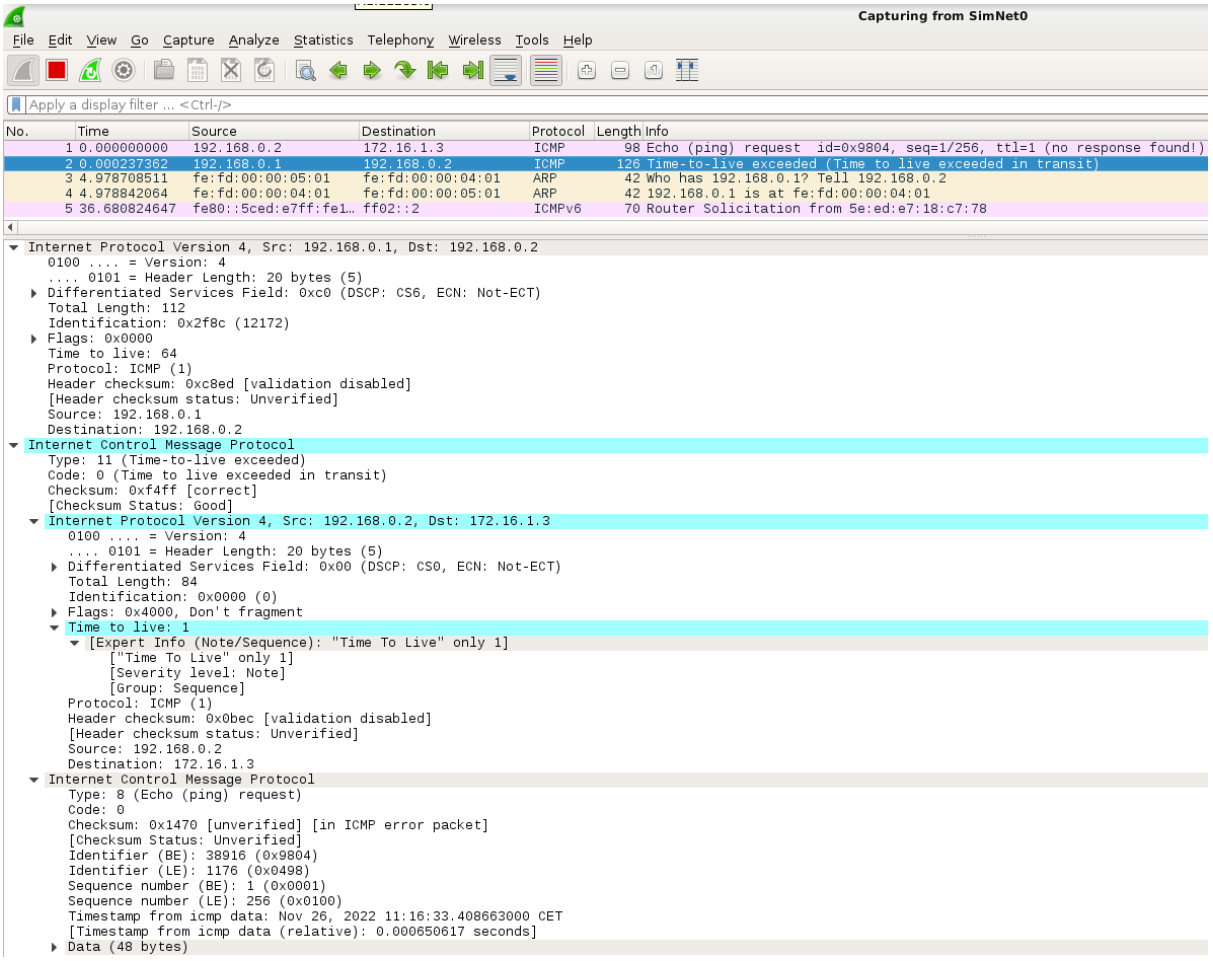
TTL	SimNet0	SimNet1	SimNet2	SimNet3
Outer header		63	62	
Inner header	64	63	63	62

4. To evaluate the behavior of the tunnel when the TTL expires, we will send IP packets from host2 (192.168.0.2) to host3 (172.16.1.3) increasing one by one the TTL. First, clear and start again the capture in the four wiresharks to perfectly monitor step by step the transmission of this packets in all the involved networks (SimNet0, SimNet1,SimNet2 and SimNet3). Next, send again one ICMP echo-request messages using the ping program and TTL=1.
(a) Can you see packets in all the networks? Did the packet reach the destination host? In which host the packet was lost?
// from host2
ping -c1 -t1 172.16.1.3



El ping solo llega al R1 y se pierde, Como tiene solo 1 ttl, hace solo 1 salto. Cuando llega al R1 su ttl es 0, entonces no hace mas saltos y se descarta.

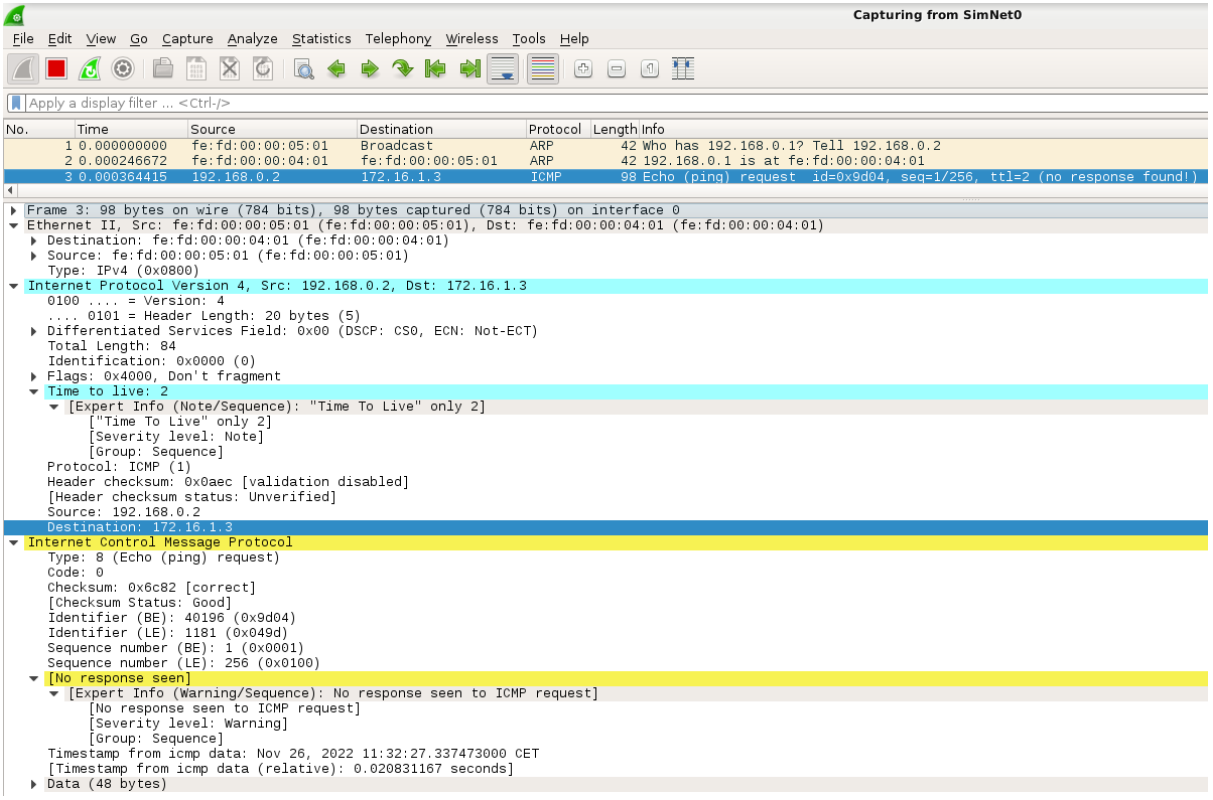
(b) Can you see any ICMP error message? Which one? In which network?
Y el R1 le envia al host2 el error ICMP: TTL exceeded in transit) in SimNet0.

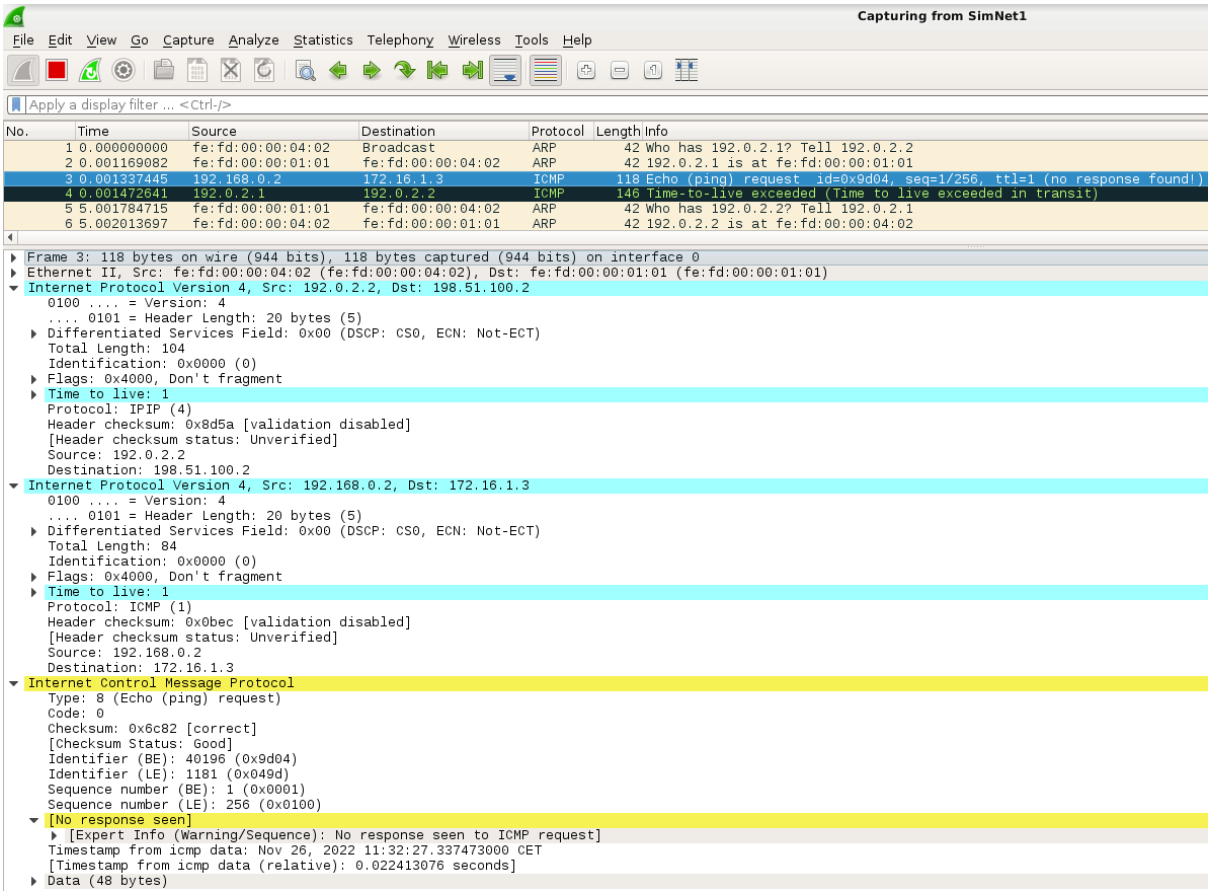


(c) Which are the origin and destination IP addresses of this error message?
ip.src = 192.168.0.1(R1 eth1) ip.dst = 192.168.0.2 (host2)

Now clear all the wiresharks and send again another ping, but with TTL=2.
(a) Did the packet reach the destination host? In which machine the packet was lost?
//from host2
ping -c1 -t2 172.16.1.3

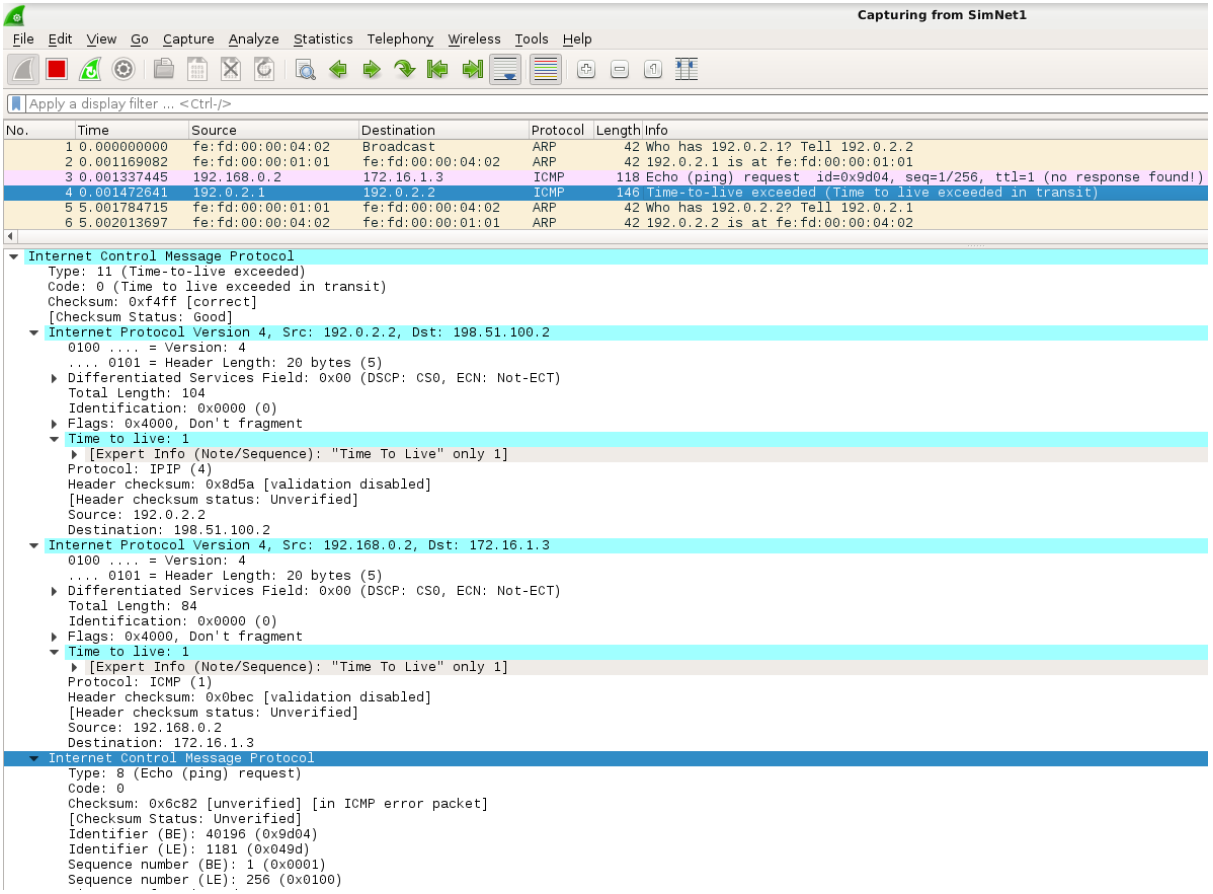
```
host2:~# ping -c1 -t2 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 56(84) bytes of data.

--- 172.16.1.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```





El paquete se vuelve a perder, Pero ahora el RC.

(b) Can you see any ICMP error message? Which one? In which network?
Si, el mismo que antes, TTL exceeded, en la SimNet0.



(c) Which is the origin and destination IP addresses of this error message?
ip.src = 192.0.2.1 (RC) ip.dst = 192.0.0.2 (eht2 R1)

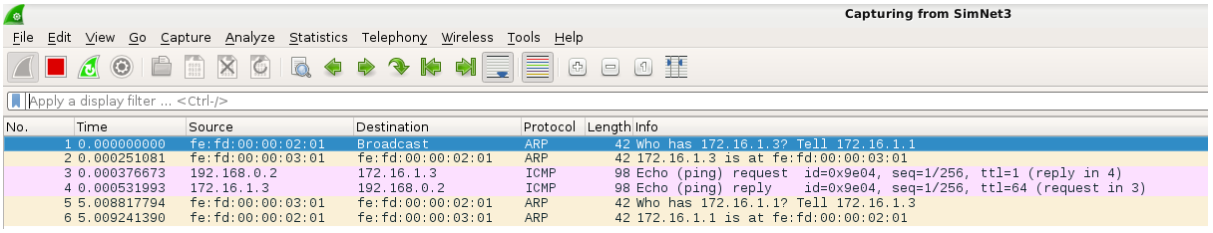
(d) Can you see any ICMP relaying?
No vemos ICMP relaying porque no el mensaje de error no va al host sino al R1.

(e) Send again the same ping with TTL=2. Is there any soft state?
No, ya que no se envia ningun mensaje desde R1 a host2.

(f) Is this behavior compliant with RFC 2003 (see section 4.4)?
Si, porque el host2 no ha recibido ningún mensaje.
(mirar el 4.4 de la documentacion)

Now use TTL=3:
//from host2
ping -c1 -t3 172.16.1.3

```
host2:~# ping -c1 -t3 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 56(84) bytes of data.
64 bytes from 172.16.1.3: icmp_seq=1 ttl=62 time=34.8 ms

--- 172.16.1.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 34.857/34.857/34.857/0.000 ms
```



(a) Did the packet reach the destination host?
si, se envia correctamente

(b) Can you see any ICMP reply message?
Suuu

0.6 Testing the tunnels: nopmtudisc

0.6.1 Limiting the MTU
1. Use ifconfig to see which is the value of the MTU in the tunnel inferfaces both in R1 and R2.
–recordatorio:

- The maximum length of an IP datagram is 64KBytes
- The value of the MTU depends on the type of transmission link, for example, ethernet links have a MTU of 1500 bytes.
- The "Path MTU" is the smallest MTU of all the hops of the path between source and destination.

//from R2 & R1
ifconfig

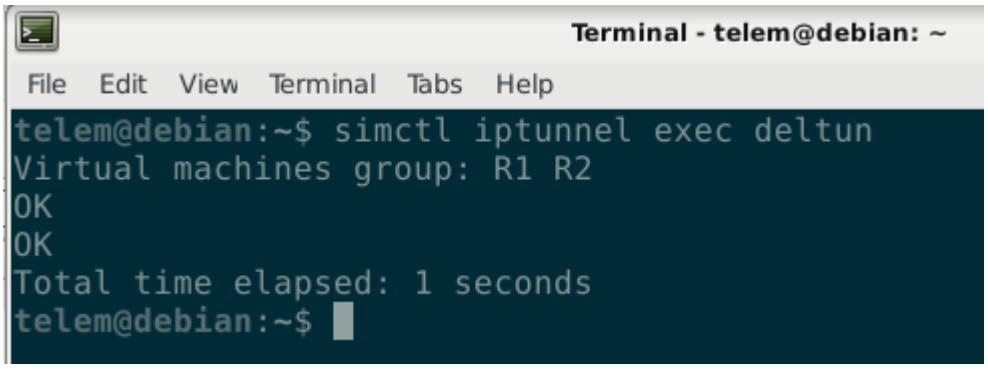
```
tunnel0   Link encap:IPIP Tunnel  HWaddr
          inet addr:1.2.3.4  P-t-P:1.2.3.4  Mask:
          UP POINTOPOINT RUNNING NOARP  MTU:1480
          RX packets:2 errors:0 dropped:0 overrun
          TX packets:2 errors:0 dropped:0 overrun
          collisions:0 txqueuelen:0
          RX bytes:168 (168.0 B)  TX bytes:168 (1

R2:~#
```

```
tunnel0   Link encap:IPIP Tunnel  HWaddr
          inet addr:1.2.3.4  P-t-P:1.2.3.4  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP  MTU:1480  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:84 (84.0 B)  TX bytes:84 (84.0 B)

R1:~#
```

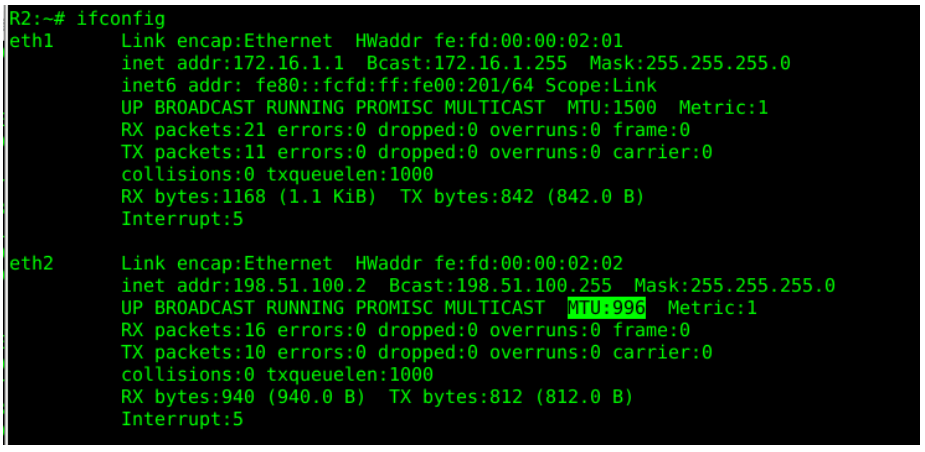
(a) Which is this value?
La MTU de tunnel es 1480 Bytes.

(b) Is there any relationship between the MTU assigned to the tunnel, and the MTU assigned to the physical interface?
La MTU de la interfaz fisica es 1500 Bytes (la estandar). La relación entre estas 2 MTU es de 20 Bytes.
Basicamente, es para que el el de tunnel pueda añadir la cabecera de outer header.

2. Delete the previously configured tunnel in R2. Instead of doing it manually, you can use this script to do so. Go to the console of your HOST machine, and execute:
//from terminal
simctl iptunnel exec deltun



3. Change the MTU of SimNet2 to the value 996 bytes. So, get the consoles of R2 and RC and use the ifconfig
command to do so. For instance, in R2 execute:
//from R2
ifconfig eth2 mtu 996
//from RC
ifconfig eth2 mtu 996



```
RC:~# ifconfig
eth1      Link encap:Ethernet  HWaddr fe:fd:00:00:01:01
          inet addr:192.0.2.1  Bcast:192.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::fcfd:ff:fe00:101/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:37 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3096 (3.0 KiB)  TX bytes:1202 (1.1 KiB)
          Interrupt:5

eth2      Link encap:Ethernet  HWaddr fe:fd:00:00:01:02
          inet addr:198.51.100.1  Bcast:198.51.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:996  Metric:1
          RX packets:32 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2852 (2.7 KiB)  TX bytes:742 (742.0 B)
          Interrupt:5
```

4. Reestablish again the tunnel using the following script:

//form terminal

simctl iptunnel exec addtun_nopmtu

```
telem@debian:~$ simctl iptunnel exec addtun_nopmtu
Virtual machines group: R1 R2
OK
OK
Total time elapsed: 1 seconds
```

(a) Which is now the MTU in both sides of the tunnel?

//from R1 //from R2

ifconfig

```
tunnel0   Link encap:IPIP Tunnel  HWaddr
          UP POINTOPOINT RUNNING NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

R1:~#
```

MTU tunnel R1= 1480

```
tunnel0   Link encap:IPIP Tunnel  HWaddr
          UP POINTOPOINT RUNNING NOARP  MTU:976  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

R2:~#
```

MTU tunnel R2= 976

(b) Why this change in the MTU value?

Para poder meter la cabecera (outer header) cuando se encapsulen los paquetes.

0.6.2 Fragmentation fields and options

1. Put all wireshark's listening traffic in all SimNet interfaces.
2. Send the following ping from host2 to host3:

//from host2

ping -c1 -s 500 -M want 172.16.1.3

-s option is used to set the number of bytes in the “data”.

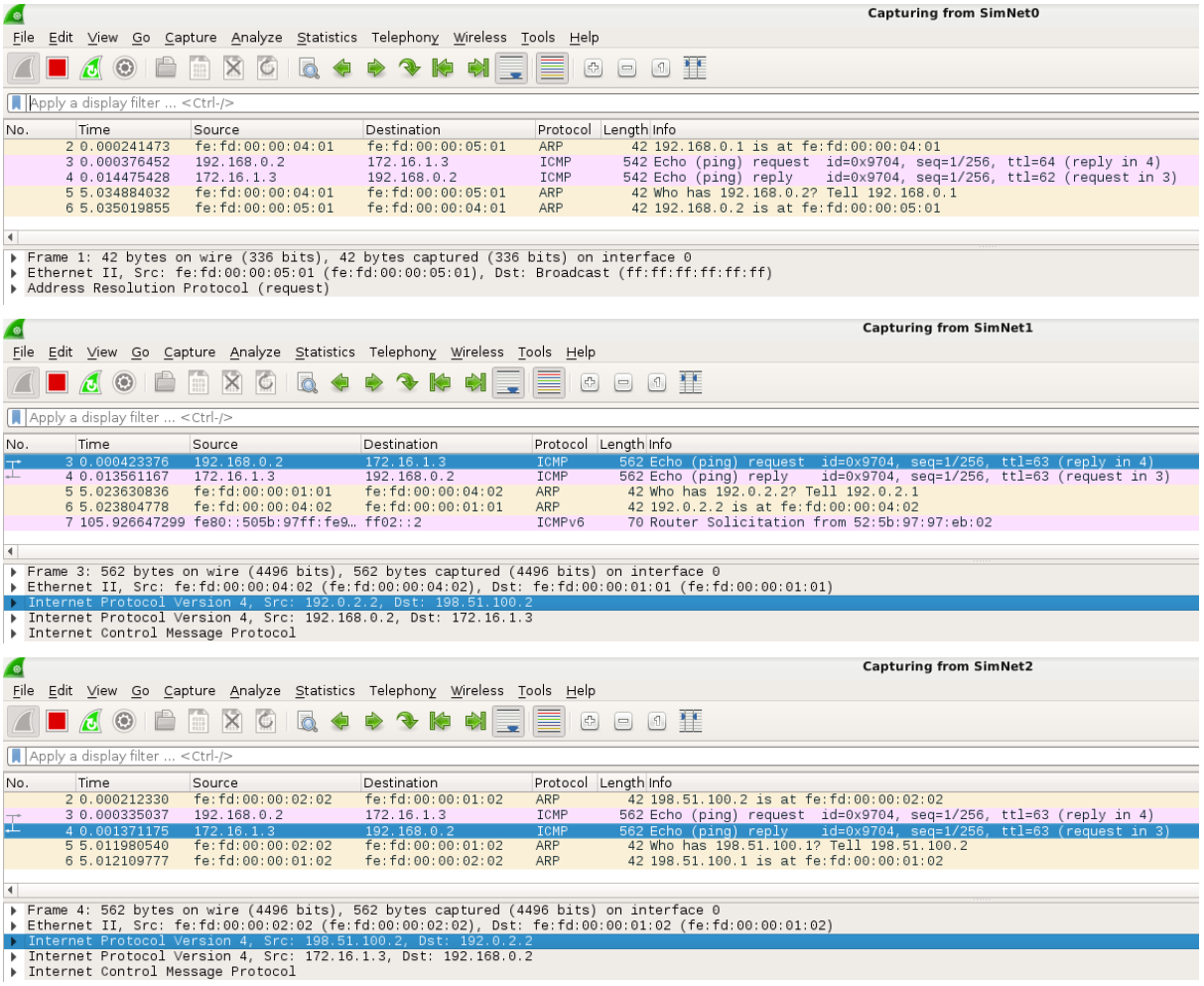
-M option is used to manage fragmentation in the origin host.

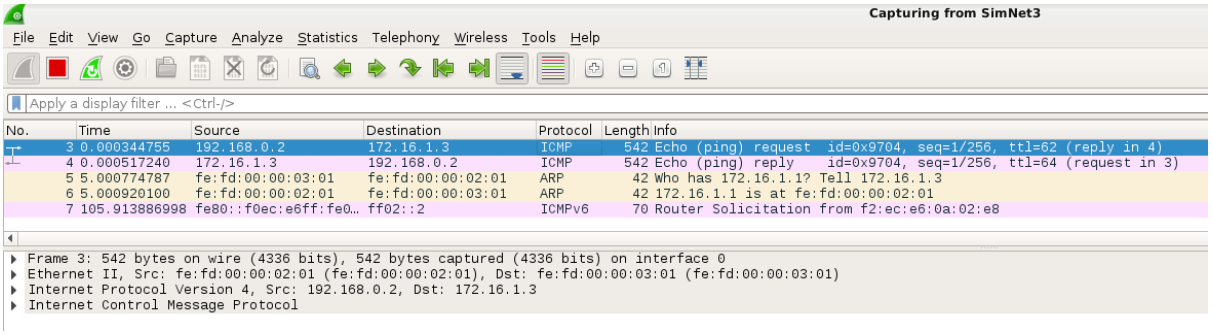
do (prohibit fragmentation, even local one).

want (do PMTU discovery, fragment locally when packet size is large).

dont (do not set DF flag).

```
host2:~# ping -c1 -s 500 -M want 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 500(528) bytes of data.
508 bytes from 172.16.1.3: icmp_seq=1 ttl=62 time=35.1 ms

--- 172.16.1.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 35.138/35.138/35.138/0.000 ms
host2:~#
```



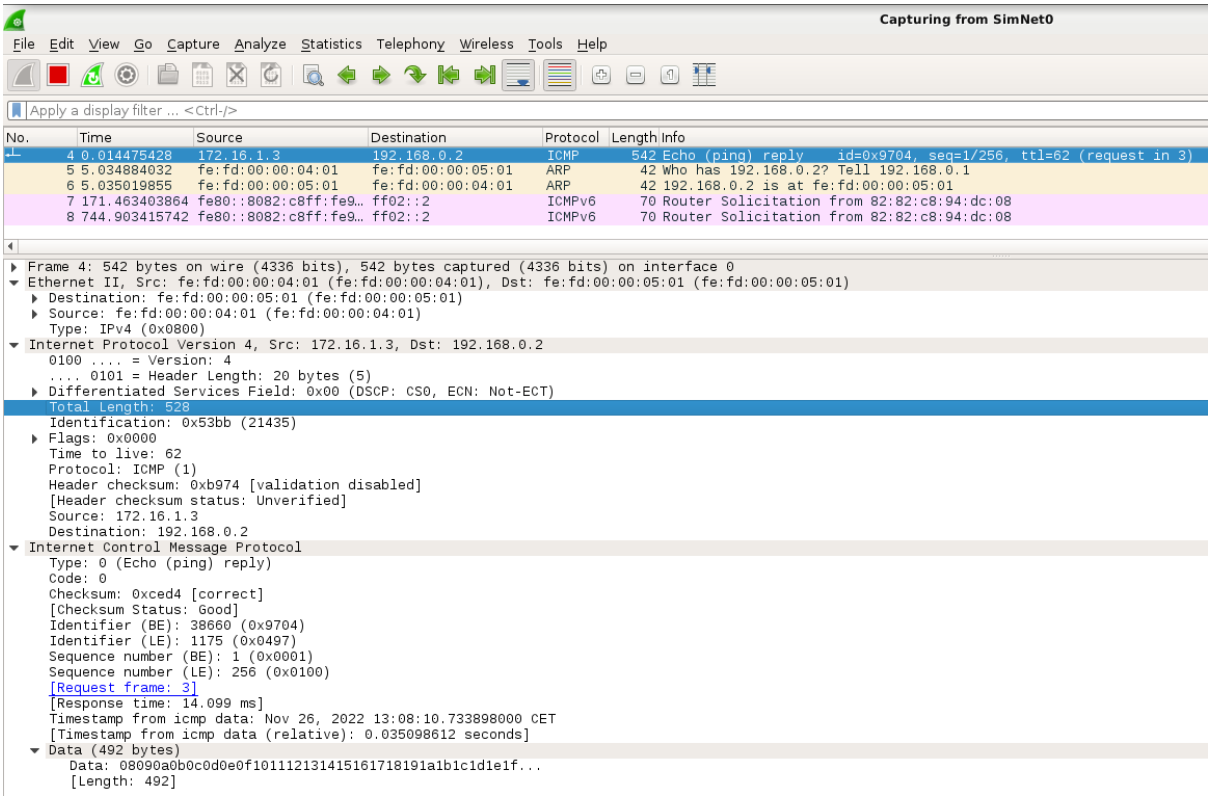


(a) How many ICMP packets can you see? What kind of packets?

En cada SImNet tenemos un echo-request y un echo-reply. En SimNet0 y SimNet3 tenemos paquetes IP y en SimNet1 y SimNet2 (tunnel) tenemos paquetes IP encapsulados en paquetes IPIP.

(b) Which is the size of these packets? Describe the headers of these packets.

En SimNet0 y SImNet3 tenemos paquete=542 Bytes. En SimNet0 y SImNet3 tenemos paquete=562 Bytes (+20 de IPIP).



TAMAÑO CABECERAS (8 icmp+20 ip +20 IPIP), pero hay mas. 14 ehternet

(c) Have these packets been fragmented? Why?

Los paquetes no han sido fragmentados porque no se supera la MTU mínima que hay entre host2 y host3 (que son 976 bytes teniendo en cuenta el túnel).

(d) Which is the value of the “don’t fragment” (DF) flag in the ICMP packets in SimNet0?

Echo repl—> ip.flags.df == 0

Echo request—> ip.flags.df == 1

(e) And in SimNet1, which is the value of DF in the Outer and Inner headers?
Echo repl—> ip.flags.df == 0
Echo request—> ip.flags.df == 1

En los 2 casos, inner header y outer header. Oute header hereda a info de inner como pasaba con los ttls.

3. Now execute
//from host2
ping -c1 -s 500 -M dont 172.16.1.3

```
host2:~# ping -c1 -s 500 -M dont 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 500(528) bytes of data.
508 bytes from 172.16.1.3: icmp_seq=1 ttl=62 time=38.9 ms

--- 172.16.1.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 38.953/38.953/38.953/0.000 ms
```

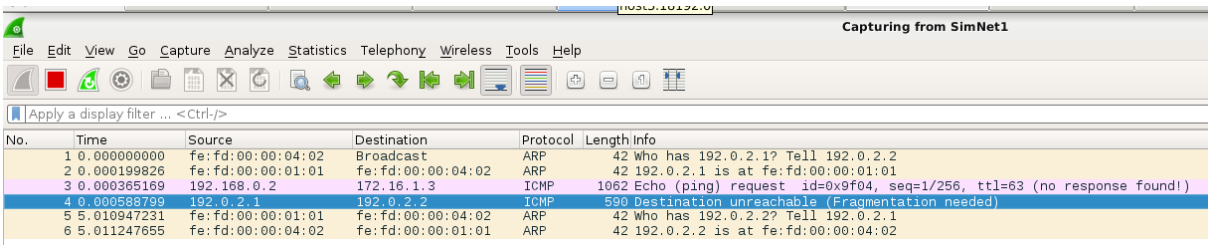
(a) What is the difference compared to the previous test?
Ahora todos los paquetes, tanto echo request como echo reply tienen DF= 0. (dont permite fragmentacion).

4. Now execute:
//from host2
ping -c1 -s 500 -M do 172.16.1.3

(a) Can you see any difference in wireshark compared to the first case? Is there any difference at all?
No hay diferencia con -M want, eso es porque no hemos necesitado fragmentar en ninguno de los casos.
La MTU mas pequeña possible es la de 996 que viene condicionada por la R2, pero en el tunnel es de 976 visto anteriormente, entonces el tamaño maximo de datos es 976-20-8= 948 Bytes.(restando cabecera ip y icmp a la mtu del tunnel)

Aqui no restamos cabecera ethernet, porque va aparte de la MTU.
Si enviamos mas de 948 Bytes de datos, el paquete se pierde.

```
host2:~# ping -c1 -s 1000 -M do  172.16.1.3
PING 172.16.1.3 (172.16.1.3) 1000(1028) bytes of data.

--- 172.16.1.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```



En este caso el router RC envia a R1 q necesita fragmentar.

The -M want option

-M want option is the default option used by ping, we only deactivate DF when needed. messages ICMP Datagram Too Big (Type 3, Code 4, also known as Fragmentation Needed), used in the PMTU Discovery mechanism, are able to modify the status of the dynamic routing table (kernel routing cache).

So, you are going to increment the size of packets to cause fragmentation:

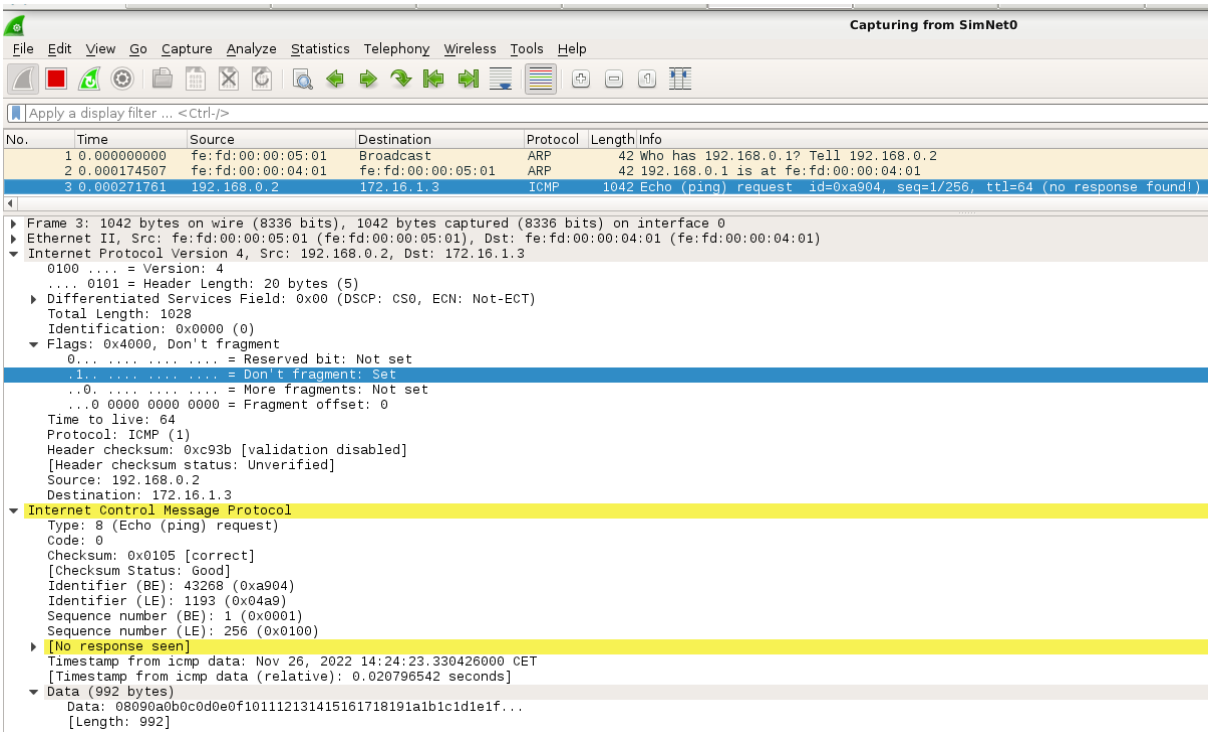
1. Delete the kernel routing cache executing the label flushcache.

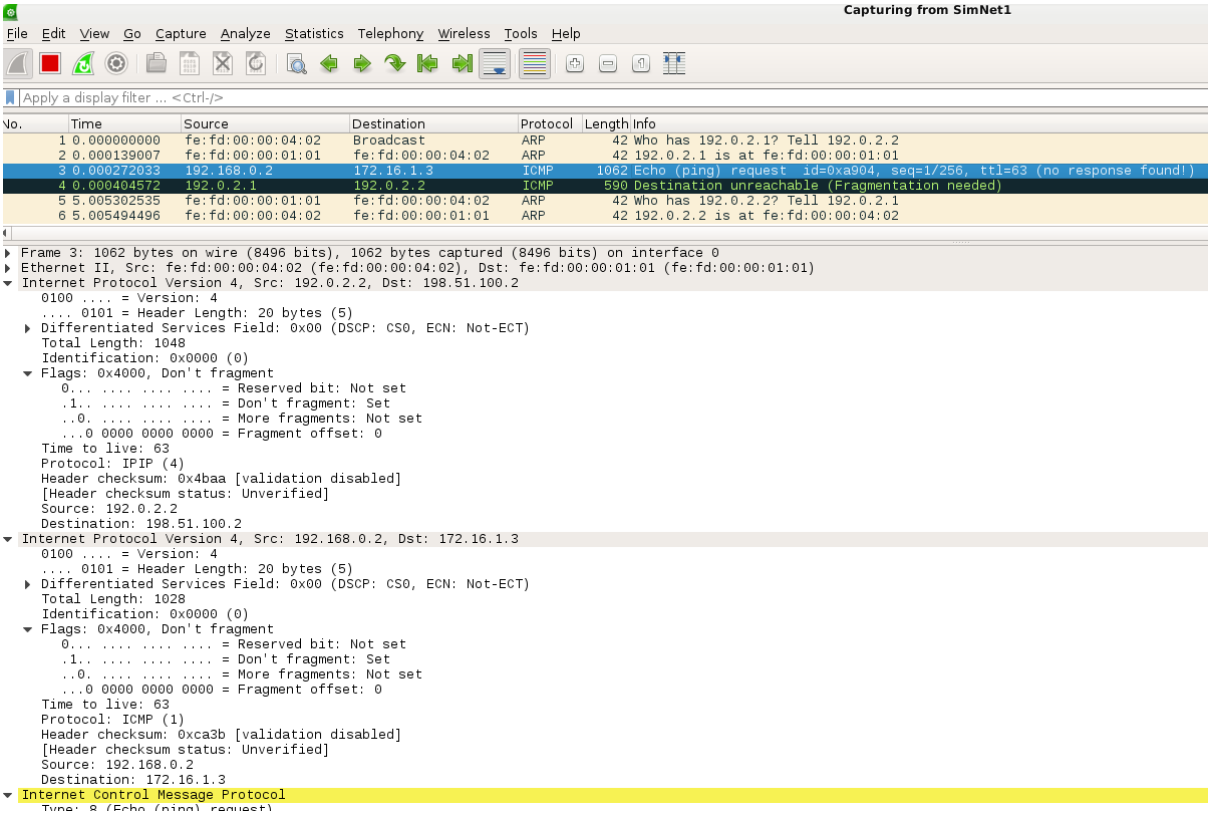
//from terminal
sudo ip route flush cache
simctl iptunnel exec flushcache #para el tunnel

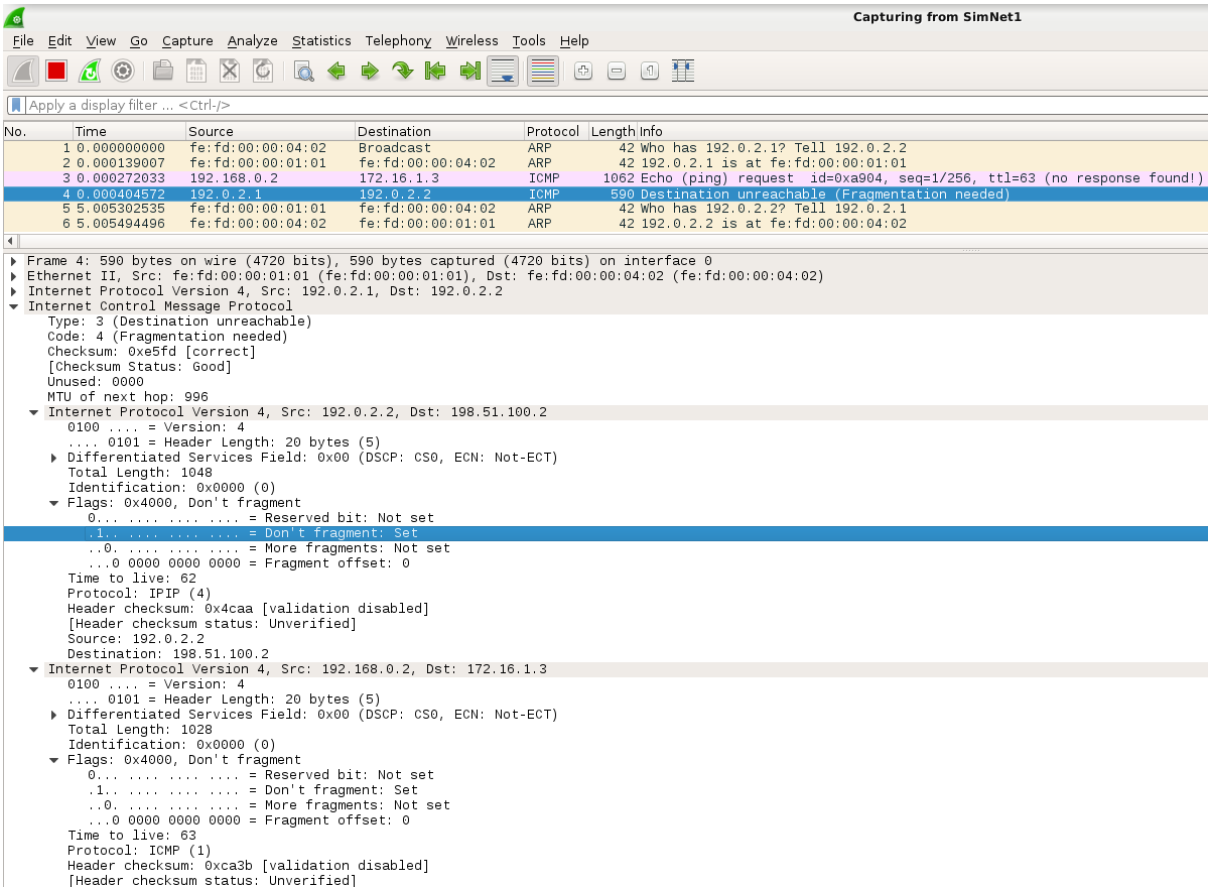
2. Put all wiresharks listening traffic in all SimNet interfaces.

3. Execute the following command in host2:

//from host2
ping -c1 -s 1000 -M want 172.16.1.3




Capturing from SimNet1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 fe:fd:00:00:04:02 Broadcast ARP 42 Who has 192.0.2.1? Tell 192.0.2.2
2 0.000139007 fe:fd:00:00:01:01 fe:fd:00:00:04:02 ARP 42 192.0.2.1 is at fe:fd:00:00:01:01
3 0.000272033 192.168.0.2 172.16.1.3 ICMP 1062 Echo (ping) request id=0xa904, seq=1/256, ttl=63 (no response found!)
4 0.000404572 192.0.2.1 192.0.2.2 ICMP 590 Destination unreachable (Fragmentation needed)
5 5.005302535 fe:fd:00:00:01:01 fe:fd:00:00:04:02 ARP 42 Who has 192.0.2.2? Tell 192.0.2.1
6 5.005494496 fe:fd:00:00:04:02 fe:fd:00:00:01:01 ARP 42 192.0.2.2 is at fe:fd:00:00:04:02
Frame 3: 1062 bytes on wire (8496 bits), 1062 bytes captured (8496 bits) on interface 0
Ethernet II, Src: fe:fd:00:00:04:02 (fe:fd:00:00:04:02), Dst: fe:fd:00:00:01:01 (fe:fd:00:00:01:01)
Internet Protocol Version 4, Src: 192.0.2.2, Dst: 198.51.100.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1048
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
0... .... = Reserved bit: Not set
.1.. .... = Don't fragment: Set
..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment offset: 0
Time to live: 63
Protocol: IPIP (4)
Header checksum: 0x4baa [validation disabled]
[Header checksum status: Unverified]
Source: 192.0.2.2
Destination: 198.51.100.2
Internet Protocol Version 4, Src: 192.168.0.2, Dst: 172.16.1.3
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1028
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
0... .... = Reserved bit: Not set
.1.. .... = Don't fragment: Set
..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment offset: 0
Time to live: 63
Protocol: ICMP (1)
Header checksum: 0xca3b [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.0.2
Destination: 172.16.1.3
Internet Control Message Protocol


Capturing from SimNet1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 fe:fd:00:00:04:02 Broadcast ARP 42 Who has 192.0.2.1? Tell 192.0.2.2
2 0.000139007 fe:fd:00:00:01:01 fe:fd:00:00:04:02 ARP 42 192.0.2.1 is at fe:fd:00:00:01:01
3 0.000272033 192.168.0.2 172.16.1.3 ICMP 1062 Echo (ping) request id=0xa904, seq=1/256, ttl=63 (no response found!)
4 0.000404572 192.0.2.1 192.0.2.2 ICMP 590 Destination unreachable (Fragmentation needed)
5 5.005302535 fe:fd:00:00:01:01 fe:fd:00:00:04:02 ARP 42 Who has 192.0.2.2? Tell 192.0.2.1
6 5.005494496 fe:fd:00:00:04:02 fe:fd:00:00:01:01 ARP 42 192.0.2.2 is at fe:fd:00:00:04:02
Frame 4: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits) on interface 0
Ethernet II, Src: fe:fd:00:00:01:01 (fe:fd:00:00:01:01), Dst: fe:fd:00:00:04:02 (fe:fd:00:00:04:02)
Internet Protocol Version 4, Src: 192.0.2.1, Dst: 192.0.2.2
Internet Control Message Protocol
Type: 3 (Destination unreachable)
Code: 4 (Fragmentation needed)
Checksum: 0xe5fd [correct]
[Checksum Status: Good]
Unused: 0000
MTU of next hop: 996
Internet Protocol Version 4, Src: 192.0.2.2, Dst: 198.51.100.2
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1048
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
0... .... = Reserved bit: Not set
.1.. .... = Don't fragment: Set
..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment offset: 0
Time to live: 62
Protocol: IPIP (4)
Header checksum: 0x4caa [validation disabled]
[Header checksum status: Unverified]
Source: 192.0.2.2
Destination: 198.51.100.2
Internet Protocol Version 4, Src: 192.168.0.2, Dst: 172.16.1.3
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1028
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
0... .... = Reserved bit: Not set
.1.. .... = Don't fragment: Set
..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment offset: 0
Time to live: 63
Protocol: ICMP (1)
Header checksum: 0xca3b [validation disabled]
[Header checksum status: Unverified]

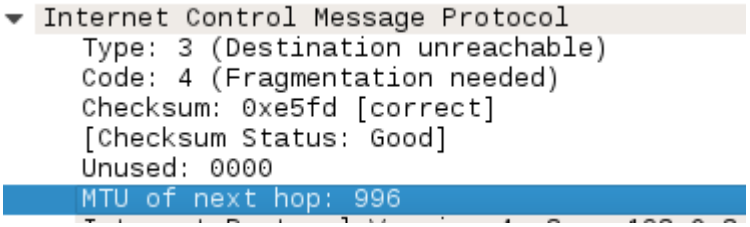
(a) Is the ping working?
No, el paquete se pierde en RC, porque necesita fragmentar. Rc envia el error (Fragmentacion needed) al R1 porque R1 teine DF=1. Pero R1 no lo envia al Host2 (no hay ICMP relying),

(b) How many ICMP packets can you see in SimNet0? What kind of packets? Is set the DF flag in the IP header of the ICMP echo-request?
Solo tenemos 1 echo-request. EL DF esta a 1, que por defecto arranca asi.

(c) And in SimNet1, what packets can you see? Is the DF flag set in the IP headers (inner and outer)?
Aqui tenemos el echo-request y tambien el ICMP error.

As you can see, there is an error produced by the MTU by means of a message ICMP fragmentation-needed. This error apears due to the DF flaf is Set in the Outer IP header of the ICMP echo-request message.

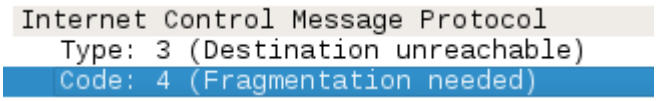
(d) What device is the origin of the ICMP fragmentation-needed message? And the destination? Which is themaximum MTU notified in this message?



RC es el dispositivo que envía el fragmentation-needed a R1, en el mensaje ICMP hay un campo “MTU of next hop: 996” que corresponde con la MTU de SimNet2.

(e) In your opinion, what entity should be the proper destination of this message?
host2

(f) According to RFC2003 (see Section 4.1), is R1 acting as relay of the message and uests can you see? Describe sizes and headers.



entonces miro la documentacion la parte 4.1 y el codigo 4.
Datagram too big. (segun el documento).

the encapsulator is not performing ICMP Relaying. Instead, the encapsulator uses soft state to inform the origin host about the error.
(g) In SimNet3, how many ICMP echo-reply messages can you see? Describe sizes and headers.
https://www.youtube.com/shorts/s8j6zW0SAyo

1. Delete the kernel routing cache executing the label flushcache.

//from terminal

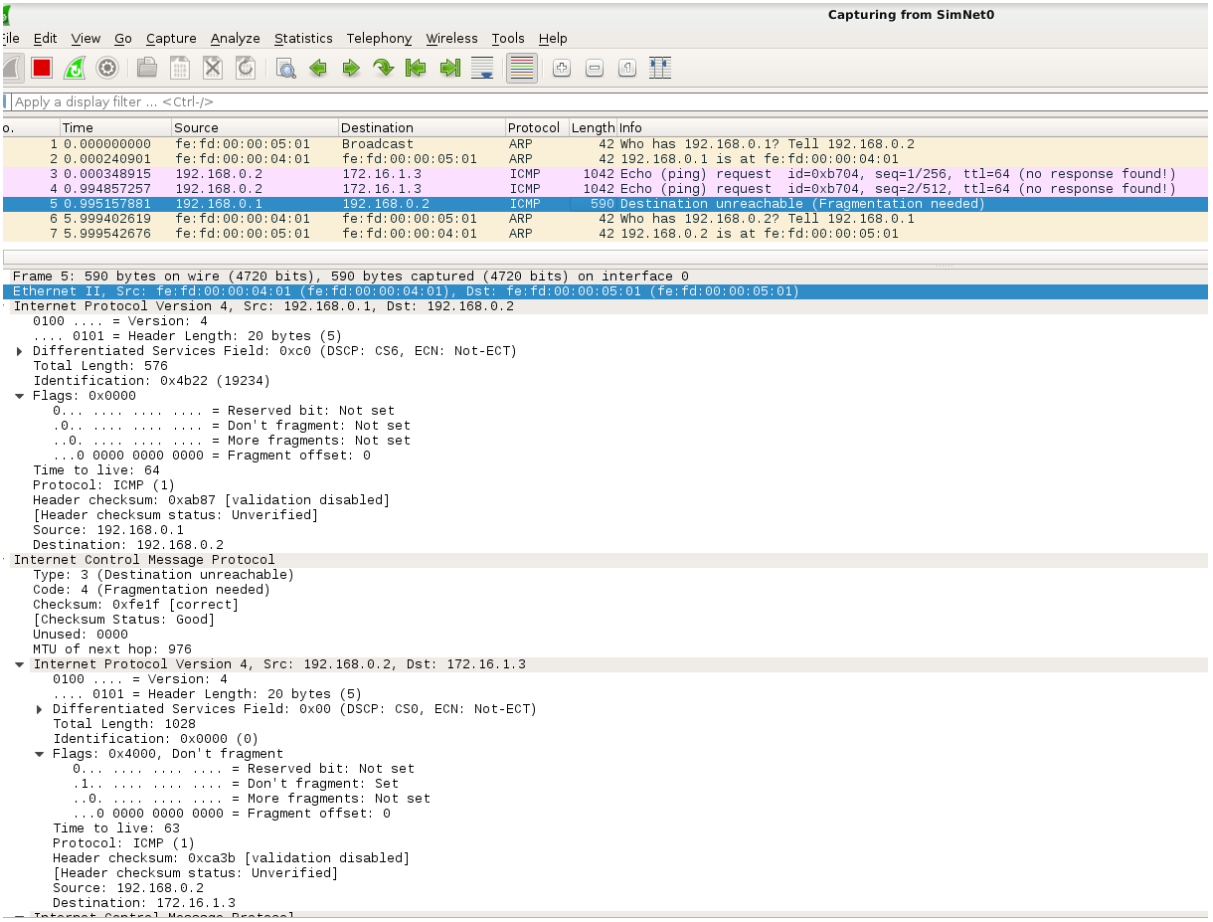
simctl iptunnel exec flushcache

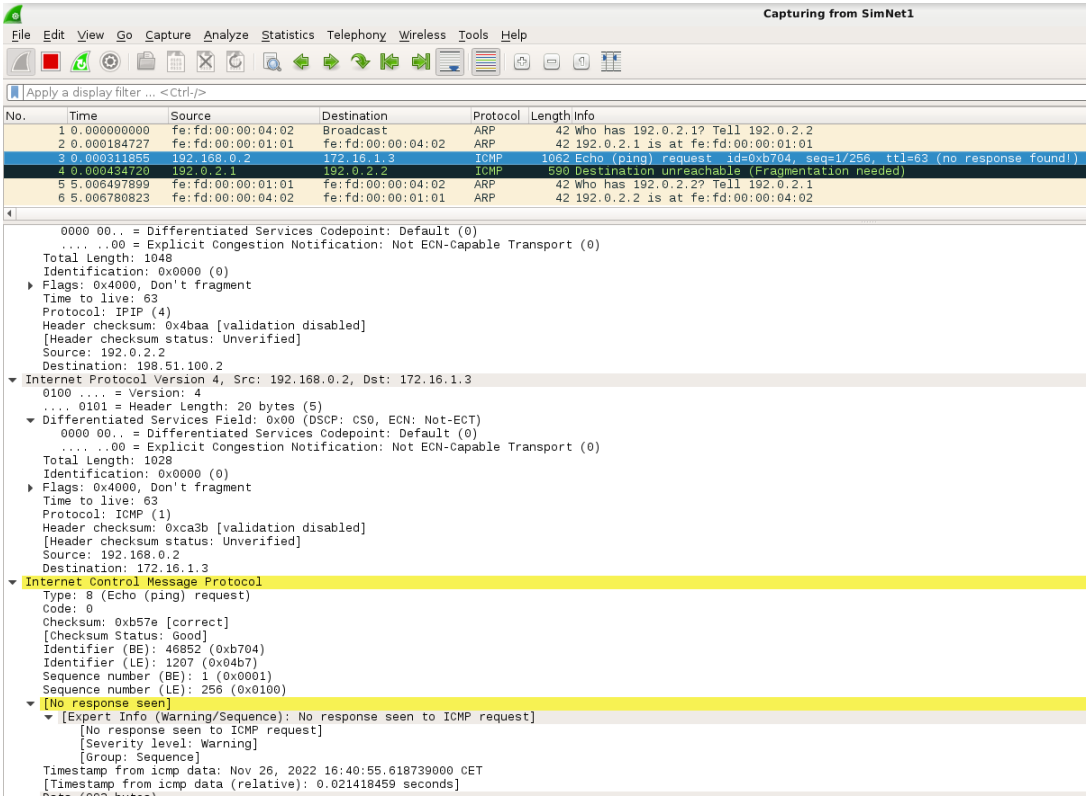
2. Put all wiresharks listening traffic in all SimNet interfaces.
3. Execute the following command in host2:

//from host2

ping -c2 -s 1000 -M want -i1 172.16.1.3

```
host2:~# ping -c2 -s 1000 -M want -i1 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 1000(1028) bytes of data.
From 192.168.0.1 icmp_seq=2 Frag needed and DF set (mtu = 976)

--- 172.16.1.3 ping statistics ---
2 packets transmitted, 0 received, +1 errors, 100% packet loss, time 1015ms
```





(a) Is the ping working? How many ICMP messages can you see in SimNet0? Is the DF flag set in the IP header of these messages?
No, el ping no funciona. En SimNet0 tenemos 3 mensajes ICMP, 2 del echo request y 1 de error.
El DF de los echo request es 1, en cambio el DF de error es 1. Es logico, porque el error es debido a eso, RC necesita fragmentar el paquete porque supera su MTU, pero le llega como DF=1 entonces envia un error diciendo que DF tiene que ser 0 porque necesita fragmentar.

(b) In SimNet1, how many ICMP echo-requests can you see? What about the DF flag?
En SimNet1, tenemos 2 mensajes ICMP, 1 de echo-request y 1 de error.
Sucede lo mismo, RC necesita fragmentar pero le llegan con DF=1 (inner & outer).
Pero aqui solo tenemos 1 ping, porque el otro ha sido descartado en el R1.

(c) Can you see an ICMP fragmentation-needed message in SimNet0? Which is the origin and destination of this message? Which is the maximum MTU notified in this message?
ip.src == 192.0.2.1, ip.dst == 192.0.2.2. icmp.mtu == 996

(d) Do you think that R1 maintains a soft state of the MTU of the tunel?

```
host2:~# ping -c2 -s 1000 -M want -i1 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 1000(1028) bytes of data.
From 192.168.0.1 icmp_seq=2 Frag needed and DF set (mtu = 976)

--- 172.16.1.3 ping statistics ---
2 packets transmitted, 0 received, +1 errors, 100% packet loss, time 1015ms
```

Yes. because host get message of error.
soft state= hay memoria, porque R1 sabe que en RC se tendra que fragmentar, entonces R1 informa al host de eso.

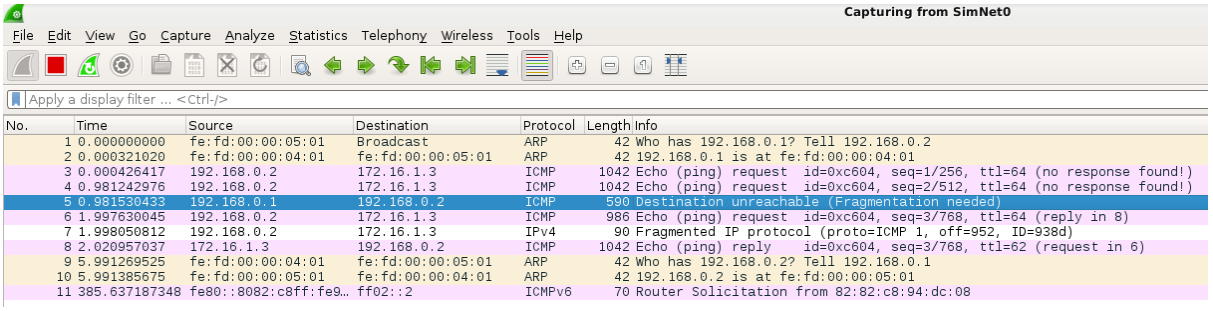
This soft state can be seen by executing the following command in R1:
ip route show cache —--- esto no me enseña nada.

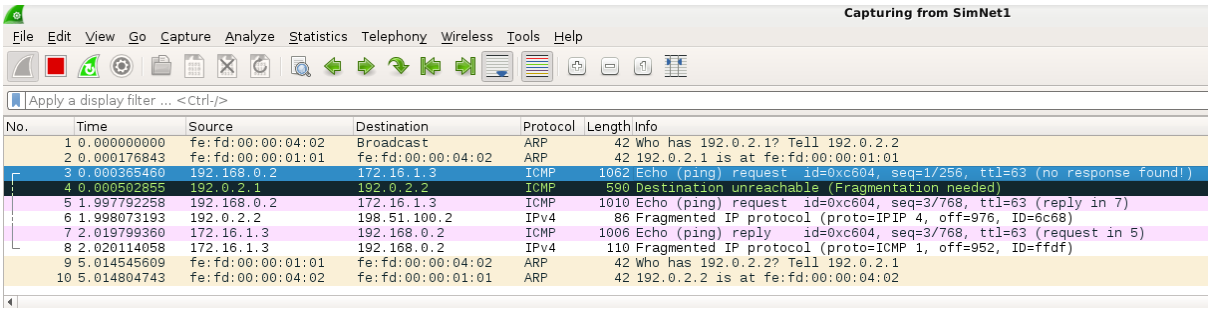
Now, we will send three ICMP echo-request messages:
1. Delete the kernel routing cache executing the label flushcache.
//from terminal
simctl iptunnel exec flushcache

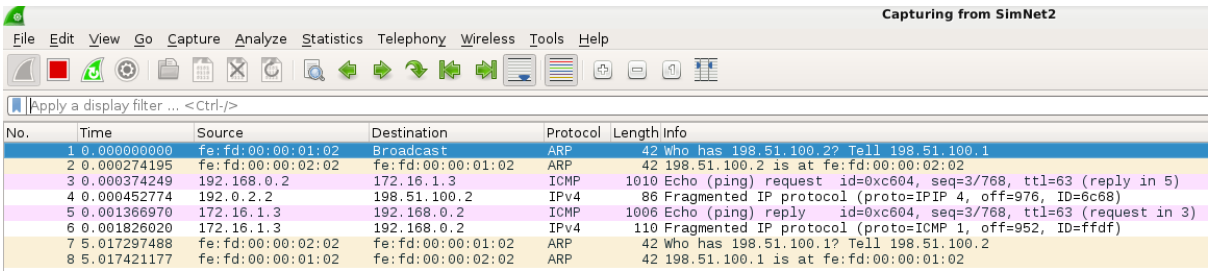
2. Put all wiresharks listening traffic in all SimNet interfaces.

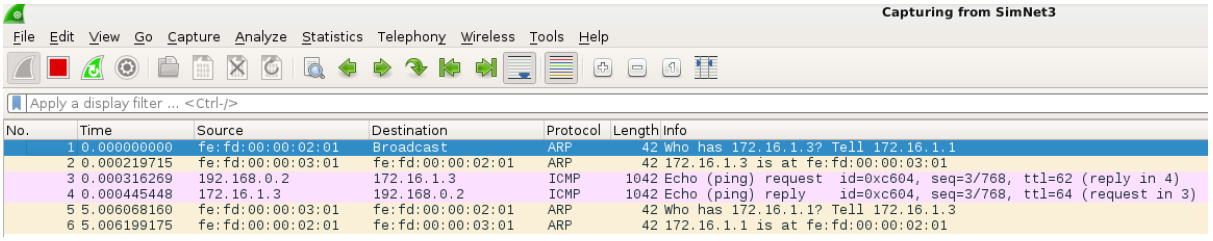
3. Execute the following command in host2:
//from host2
ping -c3 -s 1000 -M want -i1 172.16.1.3 #i= intervalo (sec)

```
host2:~# ping -c3 -s 1000 -M want -i1 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 1000(1028) bytes of data.
From 192.168.0.1 icmp_seq=2 Frag needed and DF set (mtu = 976)
1008 bytes from 172.16.1.3: icmp_seq=3 ttl=62 time=23.6 ms

--- 172.16.1.3 ping statistics ---
3 packets transmitted, 1 received, +1 errors, 66% packet loss, time 2018ms
rtt min/avg/max/mdev = 23.600/23.600/23.600/0.000 ms
```





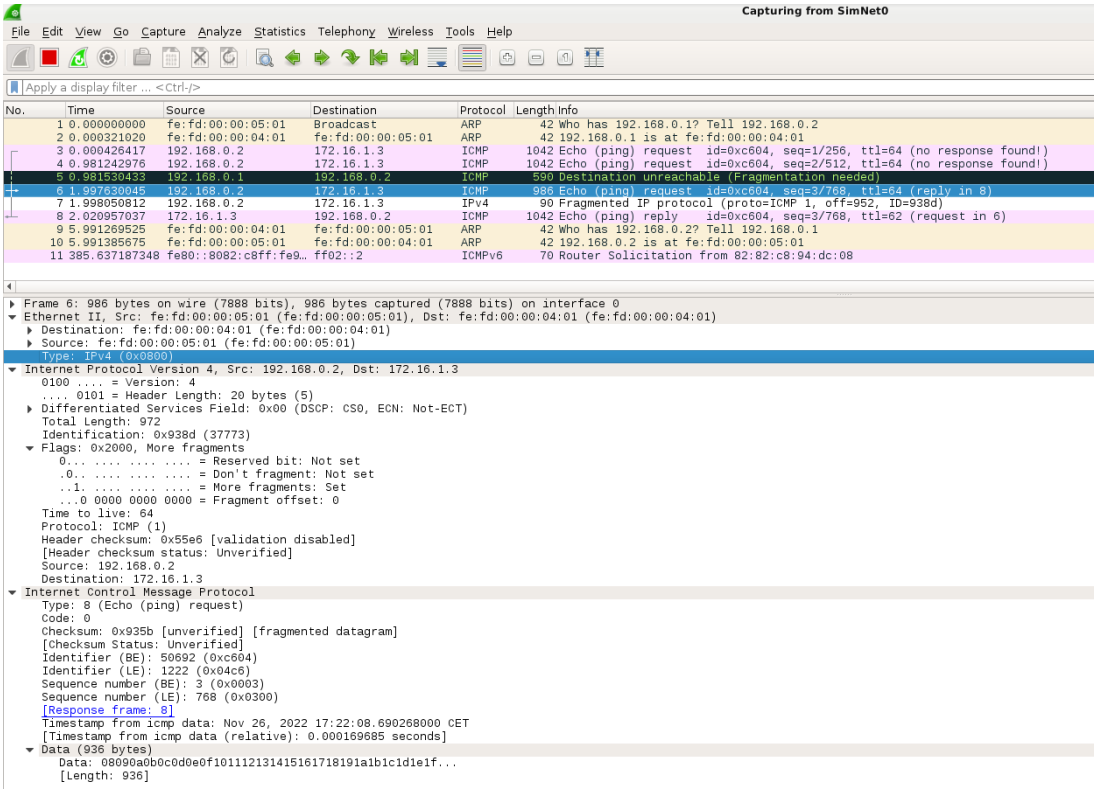




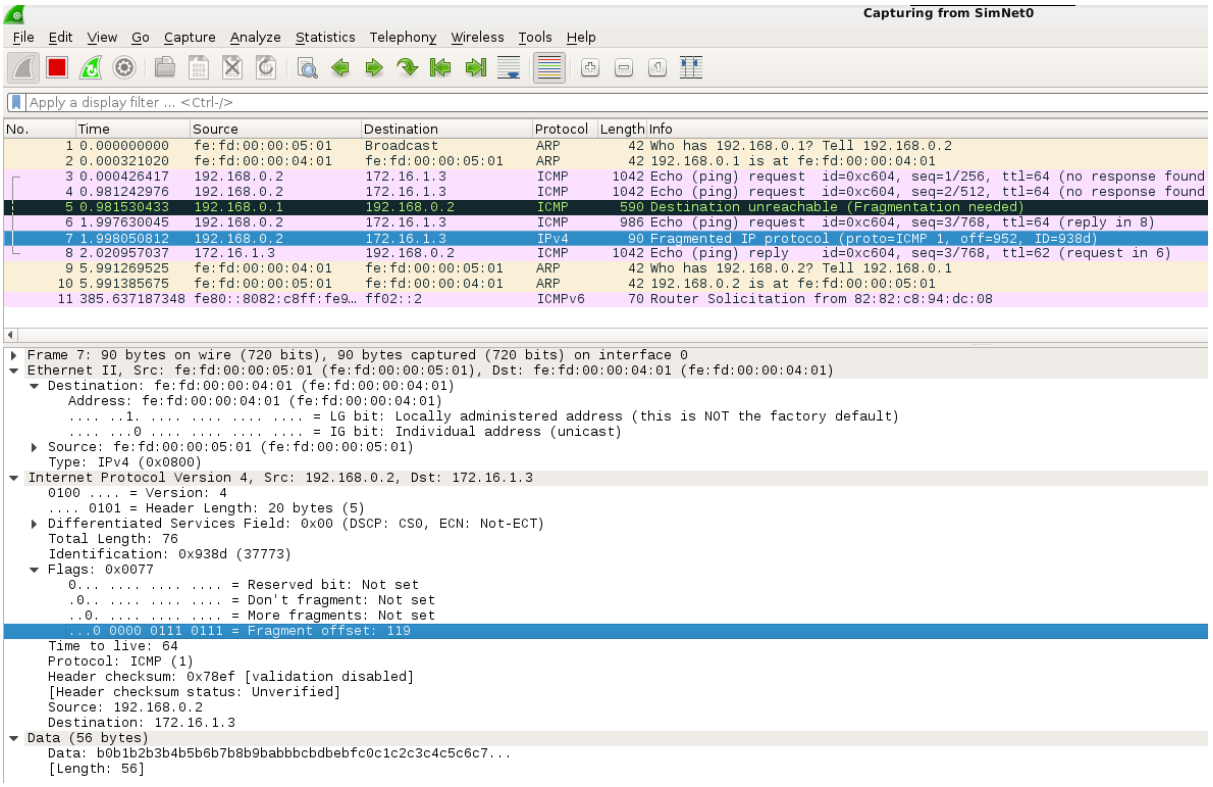
Now let us analyze this test in detail:
(a) Is the ping working? How do you know that?
Si que funciona, porque el host2 recibe un echo-reply. El ping exitoso es el 3ero ya que se ha enviado fragmentado. los primeros 2 pings son descartados en RC, pq su MTU es 996 y le llegan paquetes de 1000 con DF=1.

(b) In SimNet0, have a look at the DF flag in all the ICMP echo-request messages. What can you deduce? Is there any difference in the third ICMP Echo request message?
En SimNet0 los 2 pings tienen DF=1 (set) y el 3ero no, por tanto se fragmenta. Esta fragmentacion la hace el host2 que es el emisor.

(c) In SimNet0, who is performing the fragmentation of the third ICMP echo-request? Which are the sizes of the packets that compose this third ICMP echo-request? Take notes of the existing headers.
El host2 hace la fragmentacion, el tamaño de paquete es de 986, (936 datos + 14 eth+8 icmp+ 20 ip). falta 8 byes.
los 8 bytes estos son los timestamps, que wireshark lo interpreta como cabecera.



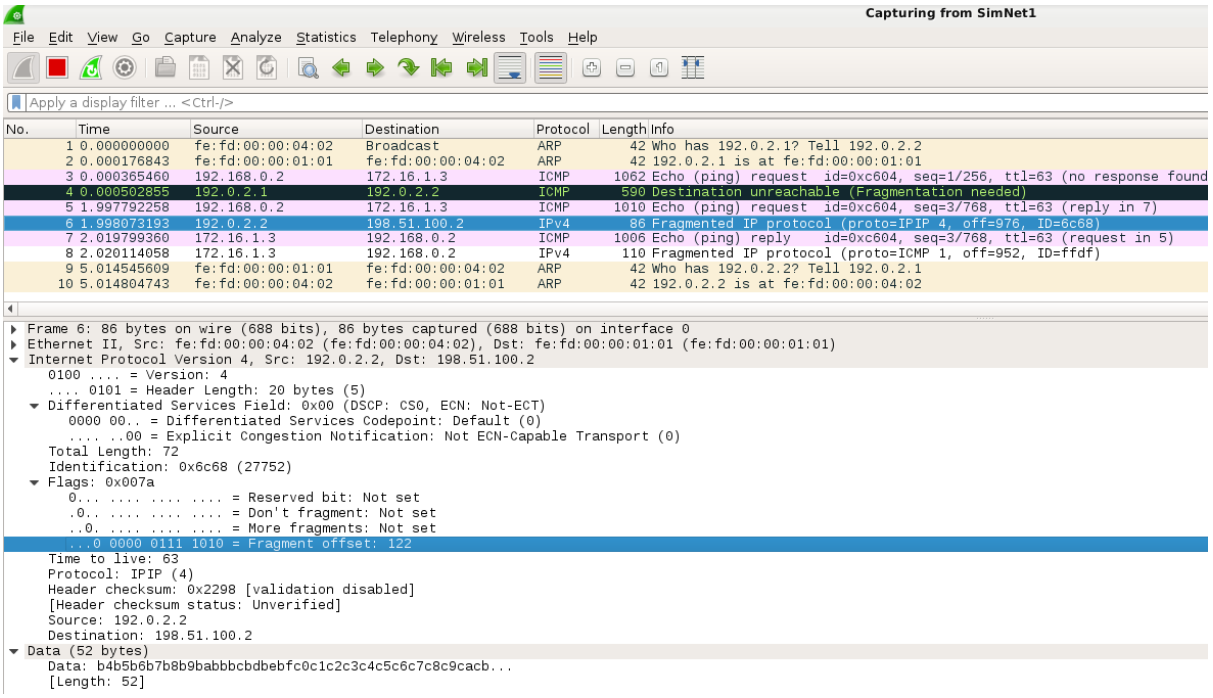
en el examen si nos dicen q hagamos paquetes de x tamaño, allí incluimos cabecera ethernet.

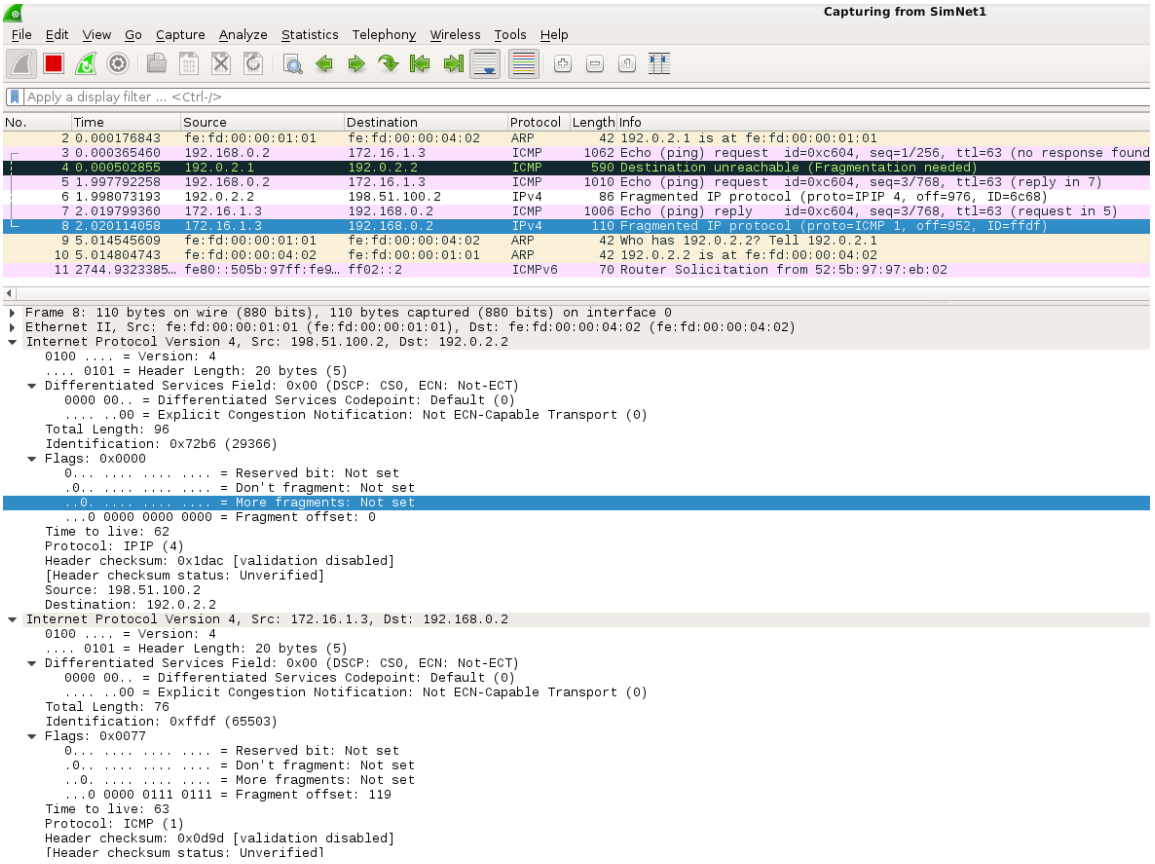


Este es el paquete fragmentado, 90B=56 data+20ip+14 eth.

Cada router recibe los paquetes fragmentados, los junta y lo fragmenta con su criterio /o no y lo envia.

(d) In SimNet1, how many fragmented packets can you see that correspond to this third ICMP echo-request? What about the DF flag in the Inner and Outer IP headers?





Hay 2 paquetes fragmentados. En las cuales el DF=0, tanto en inner header como en el outer header.

(e) Describe in detail the sizes and headers of these packets. Is fragmentation performed in the same way than SimNet0? Why the difference? Notice that the Fragment Offset field in the IP header is meassured in units of 8 bytes.
No porque en el SImNet0 tenemos la MTU es un poco mas grande que en la SimNet1, aparte en la SimNet 1 se hace la encapsulacion y en el 0 no.

(f) In SimNet3, how many ICMP echo-requests can you see? Describe sizes and headers.
1 echo request, paquete (992 dato+8) + 8 +8+14+20 = 1042.

(g) In SimNet3, how many ICMP echo-reply messages can you see? Describe sizes and headers.
1 echo reply, paquete (992 dato+8)+8+14+20 = 1042.

(h) In SimNet2, how many fragmented packets can can you see that correspond to this third ICMP echo-reply? Describe in detail sizes of these packets and headers. Is fragmentation performed in the same way than for the ICMP Request message? Describe in detail sizes of these packets and headers.
Hay 2 fragmentaciones, pero de tamaño diferente que el SimNet0 ya que ahora se ha de añadir 20 bytes de protocolo IPIP. Como las longitudes seran distintas y la fragmentacion se hace con numeros multiplos de 8 pues salen fragmentaciones distinatas.

Finally, we will send the latest ping of this series:
1. Delete the kernel routing cache executing the label flushcache.
//from terminal
simctl iptunnel exec flushcache
2. Put all wiresharks listening traffic in all SimNet interfaces.
3. Execute the following command in host2:
//from host2
ping -c2 -s 1460 -M want -i1 172.16.1.3

```
host2:~# ping -c2 -s 1460 -M want -i1 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 1460(1488) bytes of data.
From 192.168.0.1 icmp_seq=1 Frag needed and DF set (mtu = 1480)
1468 bytes from 172.16.1.3: icmp_seq=2 ttl=62 time=23.3 ms

--- 172.16.1.3 ping statistics ---
2 packets transmitted, 1 received, +1 errors, 50% packet loss, time 1008ms
rtt min/avg/max/mdev = 23.312/23.312/23.312/0.000 ms
```

Capturing from SimNet0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe:fd:00:00:05:01	Broadcast	ARP	42	Who has 192.168.0.1? Tell 192.168.0.2
2	0.000267319	fe:fd:00:00:04:01	fe:fd:00:00:05:01	ARP	42	192.168.0.1 is at fe:fd:00:00:04:01
3	0.000379535	192.168.0.2	172.16.1.3	ICMP	1502	Echo (ping) request id=0xd004, seq=1/256, ttl=64 (no response found!)
4	0.000572874	192.168.0.1	192.168.0.2	ICMP	590	Destination unreachable (Fragmentation needed)
5	0.988078947	192.168.0.2	172.16.1.3	ICMP	1490	Echo (ping) request id=0xd004, seq=2/512, ttl=64 (reply in 7)
6	0.988149555	192.168.0.2	172.16.1.3	IPv4	46	Fragmented IP protocol (proto=ICMP 1, off=1456, ID=214c)
7	1.011168381	172.16.1.3	192.168.0.2	ICMP	1502	Echo (ping) reply id=0xd004, seq=2/512, ttl=62 (request in 5)
8	5.009630569	fe:fd:00:00:04:01	fe:fd:00:00:05:01	ARP	42	Who has 192.168.0.2? Tell 192.168.0.1
9	5.009742576	fe:fd:00:00:05:01	fe:fd:00:00:04:01	ARP	42	192.168.0.2 is at fe:fd:00:00:05:01

Capturing from SimNet1

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe:fd:00:00:04:02	Broadcast	ARP	42	Who has 192.0.2.1? Tell 192.0.2.2
2	0.000260484	fe:fd:00:00:01:01	fe:fd:00:00:04:02	ARP	42	192.0.2.1 is at fe:fd:00:00:01:01
3	0.000693417	192.168.0.2	172.16.1.3	ICMP	1510	Echo (ping) request id=0xd004, seq=2/512, ttl=63 (reply in 5)
4	0.000947034	192.168.0.2	172.16.1.3	IPv4	66	Fragmented IP protocol (proto=ICMP 1, off=1456, ID=214c)
5	0.022208605	172.16.1.3	192.168.0.2	ICMP	1006	Echo (ping) reply id=0xd004, seq=2/512, ttl=63 (request in 3)
6	0.022374711	172.16.1.3	192.168.0.2	IPv4	570	Fragmented IP protocol (proto=ICMP 1, off=952, ID=68bb)
7	5.028866835	fe:fd:00:00:01:01	fe:fd:00:00:04:02	ARP	42	Who has 192.0.2.2? Tell 192.0.2.1
8	5.028993253	fe:fd:00:00:04:02	fe:fd:00:00:01:01	ARP	42	192.0.2.2 is at fe:fd:00:00:04:02

Capturing from SimNet2

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe:fd:00:00:01:02	Broadcast	ARP	42	Who has 198.51.100.2? Tell 198.51.100.1
2	0.000170227	fe:fd:00:00:02:02	fe:fd:00:00:01:02	ARP	42	198.51.100.2 is at fe:fd:00:00:02:02
3	0.000266174	192.168.0.2	172.16.1.3	ICMP	1010	Echo (ping) request id=0xd004, seq=2/512, ttl=63 (reply in 6)
4	0.000308443	192.0.2.2	198.51.100.2	IPv4	534	Fragmented IP protocol (proto=IPIP 4, off=976, ID=6c68)
5	0.000345323	192.168.0.2	172.16.1.3	IPv4	66	Fragmented IP protocol (proto=ICMP 1, off=1456, ID=214c)
6	0.001112497	172.16.1.3	192.168.0.2	ICMP	1006	Echo (ping) reply id=0xd004, seq=2/512, ttl=63 (request in 3)
7	0.001283980	172.16.1.3	192.168.0.2	IPv4	570	Fragmented IP protocol (proto=ICMP 1, off=952, ID=68bb)
8	5.015154547	fe:fd:00:00:02:02	fe:fd:00:00:01:02	ARP	42	Who has 198.51.100.1? Tell 198.51.100.2
9	5.015297527	fe:fd:00:00:01:02	fe:fd:00:00:02:02	ARP	42	198.51.100.1 is at fe:fd:00:00:01:02

Capturing from SimNet3

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe:fd:00:00:02:01	Broadcast	ARP	42	Who has 172.16.1.3? Tell 172.16.1.1
2	0.000124963	fe:fd:00:00:03:01	fe:fd:00:00:02:01	ARP	42	172.16.1.3 is at fe:fd:00:00:03:01
3	0.000245583	192.168.0.2	172.16.1.3	ICMP	1502	Echo (ping) request id=0xd004, seq=2/512, ttl=62 (reply in 4)
4	0.000347933	172.16.1.3	192.168.0.2	ICMP	1502	Echo (ping) reply id=0xd004, seq=2/512, ttl=64 (request in 3)
5	5.017375463	fe:fd:00:00:03:01	fe:fd:00:00:02:01	ARP	42	Who has 172.16.1.1? Tell 172.16.1.3
6	5.017525094	fe:fd:00:00:02:01	fe:fd:00:00:03:01	ARP	42	172.16.1.1 is at fe:fd:00:00:02:01

(a) Is there any error message? Who is the sender of this error?
Si, lo envia R1 a host2.

(b) Which is the value of MTU that is reporting this error message? Is this value the real path MTU value? Why there are no more errors during the transmission? Have a look to the DF flag

icmp.mtu == 1480, es es lo que indica el error icp. Si es el valor maximo de datos que puede pasar por el R1 ya que su MTU es 1500 (1480 datos+ 20 cabecera).
En el primer ping tenemos DF=1, pero despues de recibir el error, el host2 pone en el segundo ping DF=0.
Por eso el segundo ping es existoso, ya que viaja con el DF=0, entonces en routers que se tenga que fragmentar se fragmenta y se envia paquetes fragmentados.

The -M do option (TO DO AT HOME)

0.7 Testing tunnels: pmtudisc (TO DO AT HOME)

0.8 Testing tunnels: TCP and MSS

Maximum Segment Size (MSS) is the maximum amount of data (in bytes) that TCP can receive in a TCP segment.
In the 3-way handshake (SYN, SYN/ACK, ACK) of TCP, each side informs its MSS value to the other side.
This is done by means of the “Options“ field of the TCP headers in the SYN segments.
Each host calculates its MSS by deducting the minimum IP and TCP headers to the MTU value. In case of using Ethernet, a host will advertise a MSS of 1460 bytes (1500 bytes of MTU - 20 bytes of minimum IP header - 20 bytes of minimum TCP header).
Now you are going to test how the TCP protocol behaves when the connection go through a tunnel. First, reestablish the tunnel with the option nomptudisc. Leave the MTU=996 in SimNet2.

//from terminal
simctl iptunnel exec deltun
simctl iptunnel exec addtun_nopmtu

//from R1 and RC
ifconfig eth2 mtu 996

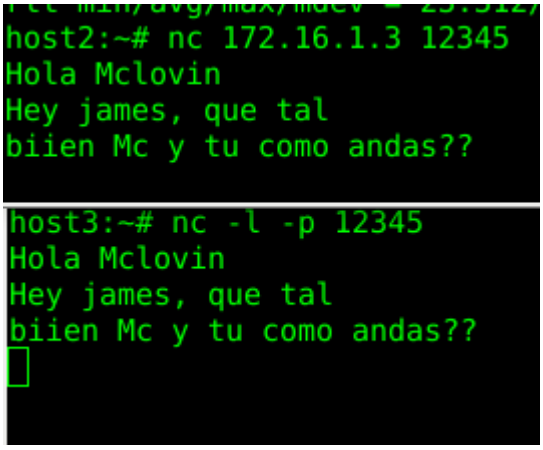
0.8.1 Netcat and TCP

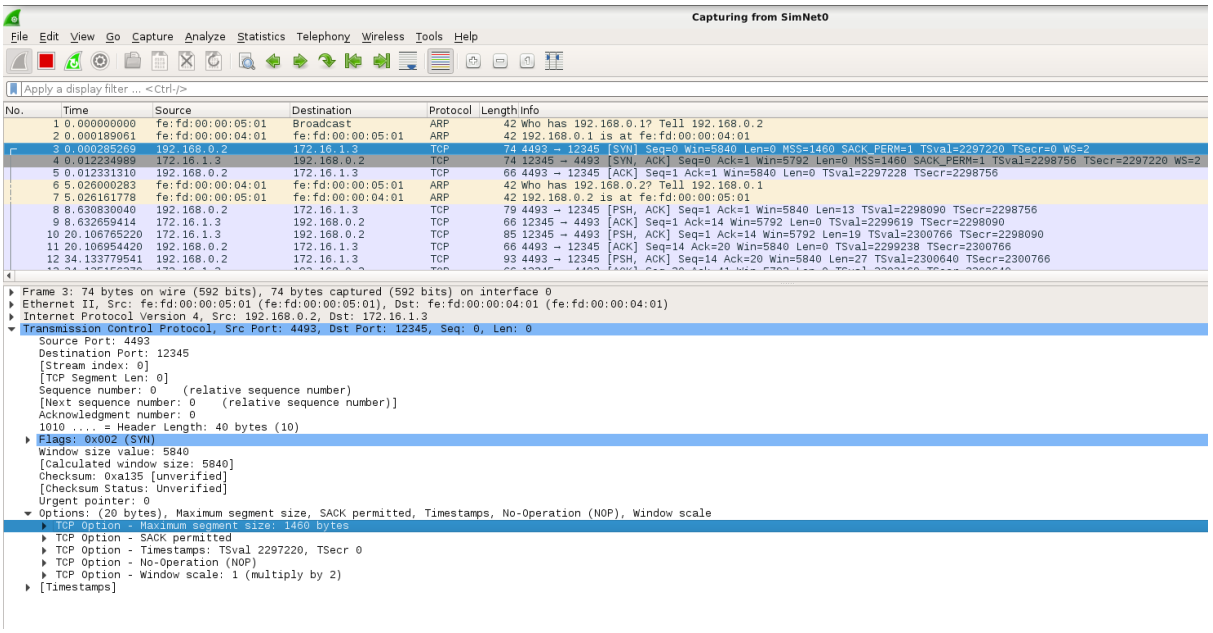
First, we will start doing a correct transmission (without filtering rules) using netcat:

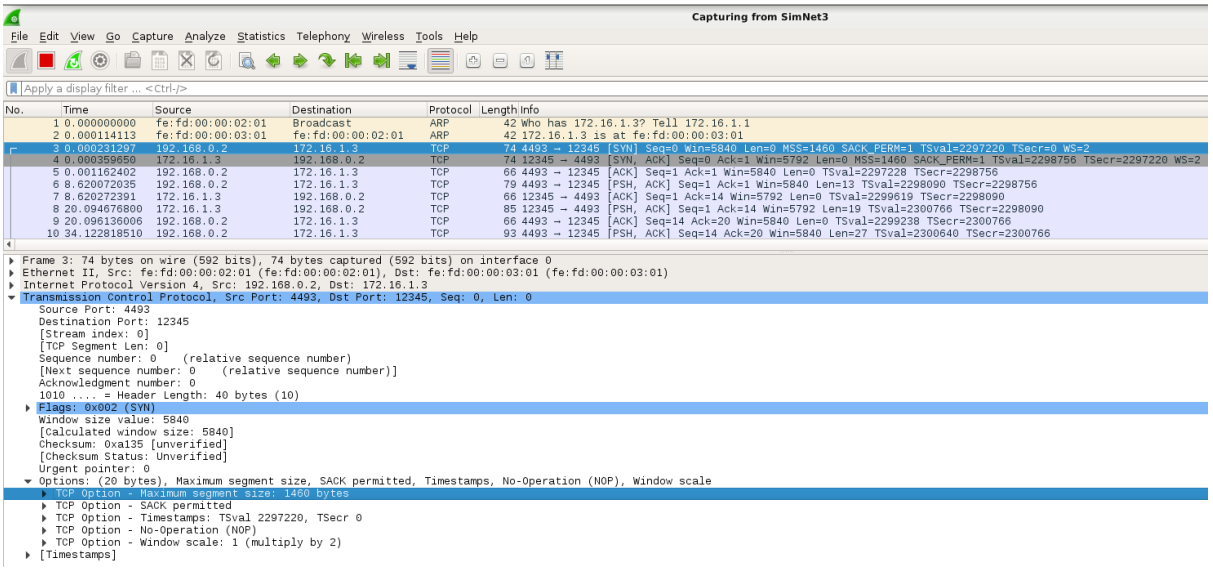
1. Put all wiresharks listening traffic in all SimNet interfaces.
2. Start a netcat server in host3 listening in port 12345 (the -l option means listen, that is, this host is behaving as server):

//from host3
nc -l - p 12345

//from host2
nc 172.16.1.3







(a) Can you see the classical 3-way handshake of TCP?
SIUUUU

(b) Which is the value of the MSS (Maximum Segment Size) advertised in this handshake?
1460 Bytes tantto el cliente como el servidor.

(c) What is the meaning of this MSS? Can have the client a MSS different from the server?
El MSS es el máximo número de bytes de datos que un host puede recibir a través de TCP en un segmento TCP. El cliente podría tener un MSS diferente al servidor ya que depende del MTU de la capa de enlace, de todas formas hay que recordar que no se negocia con el MSS sino que se informa de éste y el que envía los datos está obligado a usarlo.

0.8.2 Netcat to transfer files

Netcat can also be used to transfer files:

1. Delete the kernel routing cache executing the label flushcache.

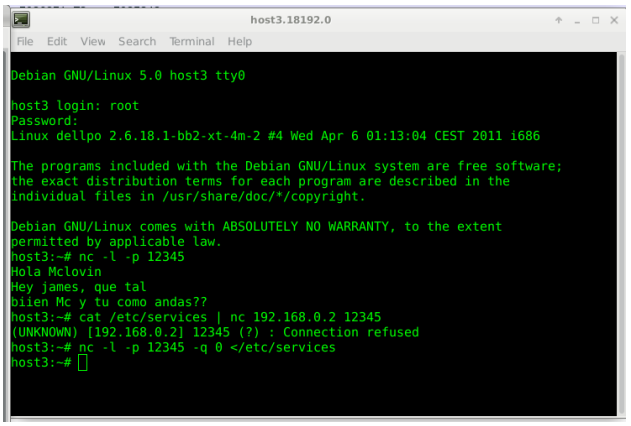
//from terminal
simctl iptunnel exec deltun
simctl iptunnel exec addtun_nopmtu

//from R1 and RC
ifconfig eth2 mtu 996

2. Put all wiresharks listening traffic in all SimNet interfaces.

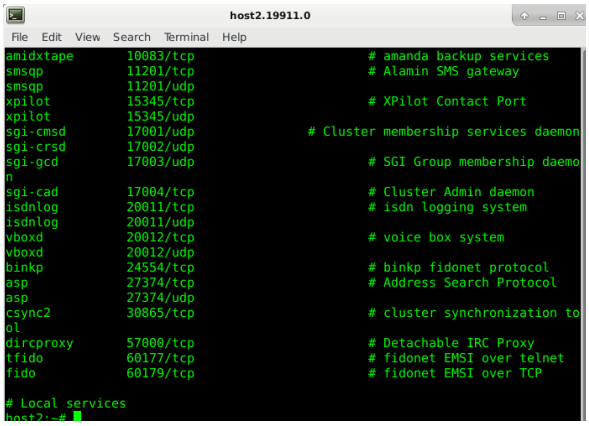
3. Start the netcat server in host3, that will send the file /etc/services towards the client when the connection has been established:

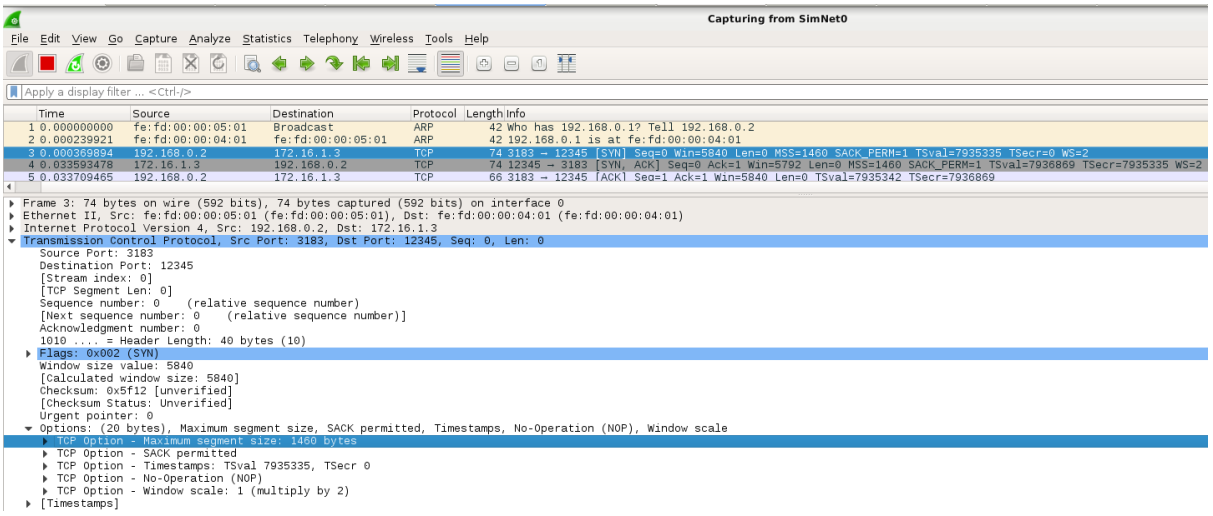
//from host3
nc -l -p 12345 -q 0 </etc/services

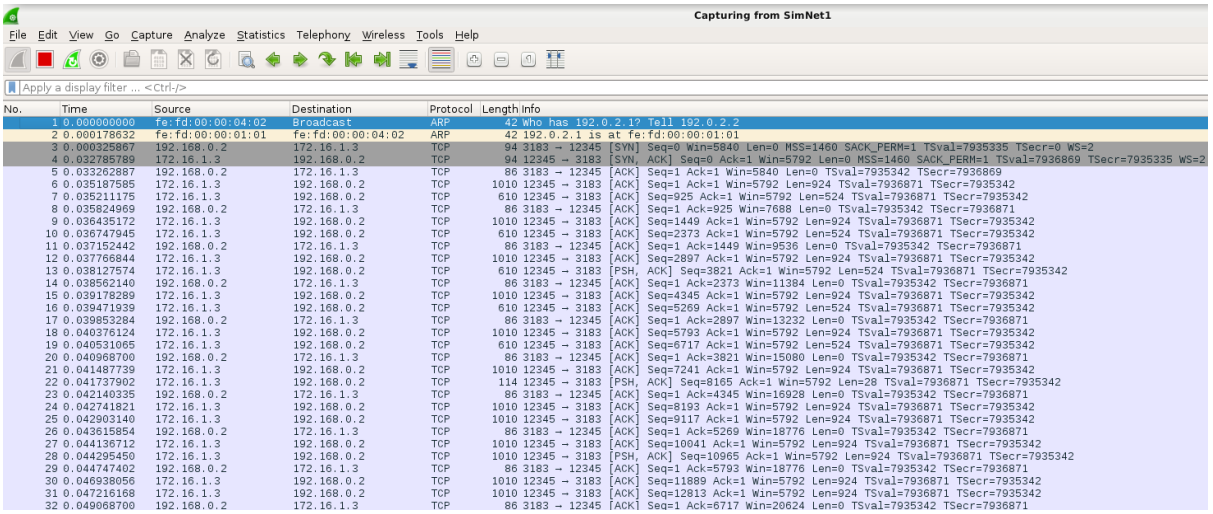


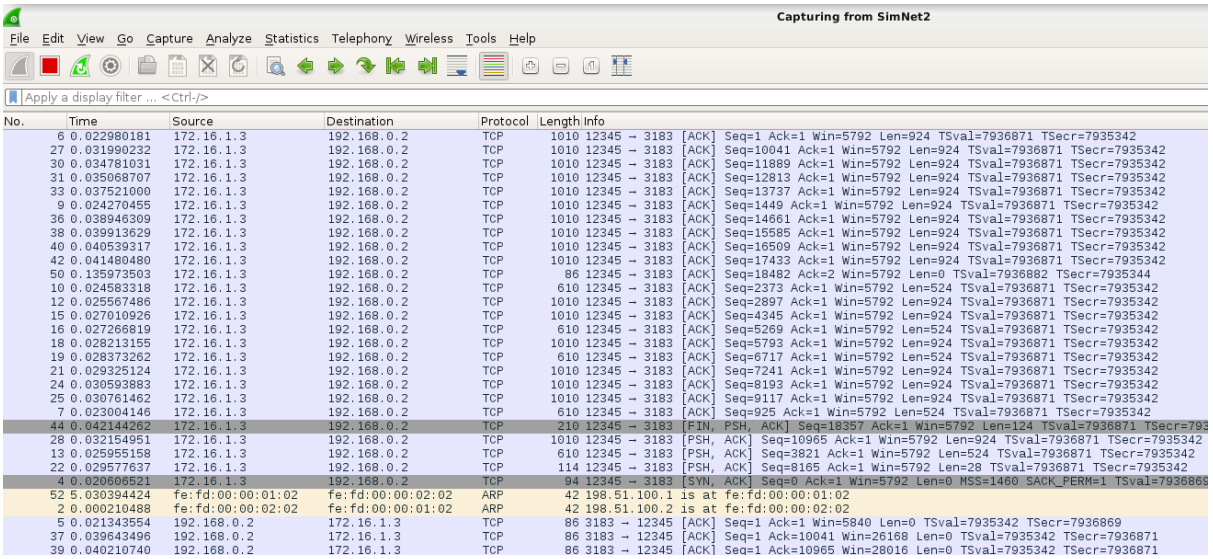
4. Start the netcat client in host2, that will display in the screen the file /etc/services of the server.

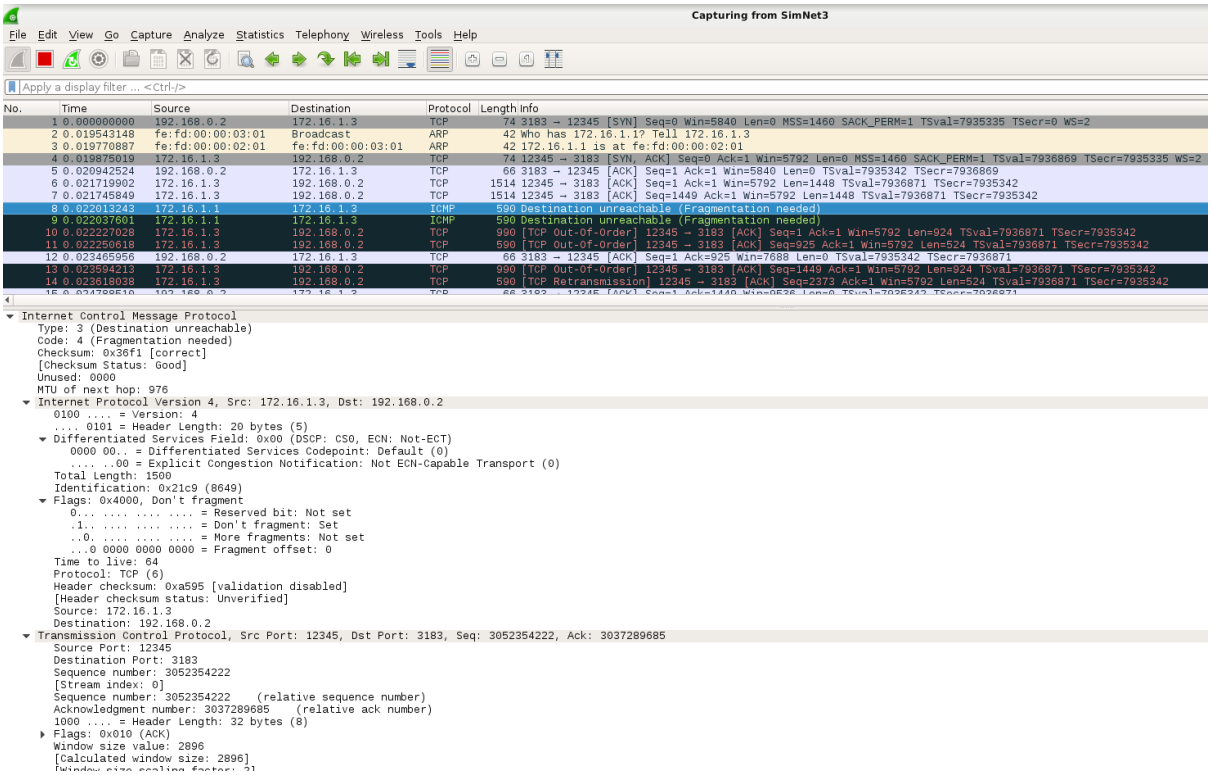
//from host2
nc 172.16.1.3 12345


host2.19911.0
File Edit View Search Terminal Help
amidxtape 10083/tcp # amanda backup services
smsqp 11201/tcp # Alamin SMS gateway
smsqp 11201/udp
xpilot 15345/tcp # XPilot Contact Port
xpilot 15345/udp
sgi-cmsd 17001/udp # Cluster membership services daemon
sgi-crsd 17002/udp
sgi-gcd 17003/udp # SGI Group membership daemo
n
sgi-cad 17004/tcp # Cluster Admin daemon
isdnlog 20011/tcp # isdn logging system
isdnlog 20011/udp
vboxd 20012/tcp # voice box system
vboxd 20012/udp
binkp 24554/tcp # binkp fidonet protocol
asp 27374/tcp # Address Search Protocol
asp 27374/udp
csync2 30865/tcp # cluster synchronization to
ol
dircproxy 57000/tcp # Detachable IRC Proxy
tfido 60177/tcp # fidonet EMSI over telnet
fido 60179/tcp # fidonet EMSI over TCP
Local services
host2:~#


Capturing from SimNet0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 fe:fd:00:00:05:01 Broadcast ARP 42 Who has 192.168.0.1? Tell 192.168.0.2
2 0.000239921 fe:fd:00:00:04:01 fe:fd:00:00:05:01 ARP 42 192.168.0.1 is at fe:fd:00:00:04:01
3 0.000369894 192.168.0.2 172.16.1.3 TCP 74 3183 → 12345 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=7935335 TSecr=0 WS=2
4 0.033593478 172.16.1.3 192.168.0.2 TCP 74 12345 → 3183 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=7936869 TSecr=7935335 WS=2
5 0.033709465 192.168.0.2 172.16.1.3 TCP 66 3183 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=7935342 TSecr=7936869
Frame 3: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: fe:fd:00:00:05:01 (fe:fd:00:00:05:01), Dst: fe:fd:00:00:04:01 (fe:fd:00:00:04:01)
Internet Protocol Version 4, Src: 192.168.0.2, Dst: 172.16.1.3
Transmission Control Protocol, Src Port: 3183, Dst Port: 12345, Seq: 0, Len: 0
Source Port: 3183
Destination Port: 12345
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
[Next sequence number: 0 (relative sequence number)]
Acknowledgment number: 0
1010 .... = Header Length: 40 bytes (10)
Flags: 0x002 (SYN)
Window size value: 5840
[Calculated window size: 5840]
Checksum: 0x5f12 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
TCP Option - Maximum segment size: 1460 bytes
TCP Option - SACK permitted
TCP Option - Timestamps: TSval 7935335, TSecr 0
TCP Option - No-Operation (NOP)
TCP Option - Window scale: 1 (multiply by 2)
[Timestamps]


Capturing from SimNet1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 fe:fd:00:00:04:02 Broadcast ARP 42 Who has 192.0.2.1? Tell 192.0.2.2
2 0.000178632 fe:fd:00:00:01:01 fe:fd:00:00:04:02 ARP 42 192.0.2.1 is at fe:fd:00:00:01:01
3 0.000325867 192.168.0.2 172.16.1.3 TCP 94 3183 → 12345 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=7935335 TSecr=0 WS=2
4 0.032785789 172.16.1.3 192.168.0.2 TCP 94 12345 → 3183 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=7936869 TSecr=7935335 WS=2
5 0.033262887 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=7935342 TSecr=7936869
6 0.035187585 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=1 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
7 0.035211175 172.16.1.3 192.168.0.2 TCP 610 12345 → 3183 [ACK] Seq=925 Ack=1 Win=5792 Len=524 TSval=7936871 TSecr=7935342
8 0.035824969 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=925 Win=7688 Len=0 TSval=7935342 TSecr=7936871
9 0.036435172 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=1449 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
10 0.036747945 172.16.1.3 192.168.0.2 TCP 610 12345 → 3183 [ACK] Seq=2373 Ack=1 Win=5792 Len=524 TSval=7936871 TSecr=7935342
11 0.037152442 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=1449 Win=9536 Len=0 TSval=7935342 TSecr=7936871
12 0.037766844 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=2897 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
13 0.038127574 172.16.1.3 192.168.0.2 TCP 610 12345 → 3183 [PSH, ACK] Seq=3821 Ack=1 Win=5792 Len=524 TSval=7936871 TSecr=7935342
14 0.038562140 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=2373 Win=11384 Len=0 TSval=7935342 TSecr=7936871
15 0.039178289 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=4345 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
16 0.039471939 172.16.1.3 192.168.0.2 TCP 610 12345 → 3183 [ACK] Seq=5269 Ack=1 Win=5792 Len=524 TSval=7936871 TSecr=7935342
17 0.039853284 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=2897 Win=13232 Len=0 TSval=7935342 TSecr=7936871
18 0.040376124 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=5793 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
19 0.040531065 172.16.1.3 192.168.0.2 TCP 610 12345 → 3183 [ACK] Seq=6717 Ack=1 Win=5792 Len=524 TSval=7936871 TSecr=7935342
20 0.040968700 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=3821 Win=15080 Len=0 TSval=7935342 TSecr=7936871
21 0.041487739 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=7241 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
22 0.041737902 172.16.1.3 192.168.0.2 TCP 114 12345 → 3183 [PSH, ACK] Seq=8165 Ack=1 Win=5792 Len=28 TSval=7936871 TSecr=7935342
23 0.042140335 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=4345 Win=16928 Len=0 TSval=7935342 TSecr=7936871
24 0.042741821 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=8193 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
25 0.042903140 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=9117 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
26 0.043615854 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=5269 Win=18776 Len=0 TSval=7935342 TSecr=7936871
27 0.044136712 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=10041 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
28 0.044295450 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [PSH, ACK] Seq=10965 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
29 0.044747402 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=5793 Win=18776 Len=0 TSval=7935342 TSecr=7936871
30 0.046938056 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=11889 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
31 0.047216168 172.16.1.3 192.168.0.2 TCP 1010 12345 → 3183 [ACK] Seq=12813 Ack=1 Win=5792 Len=924 TSval=7936871 TSecr=7935342
32 0.049068700 192.168.0.2 172.16.1.3 TCP 86 3183 → 12345 [ACK] Seq=1 Ack=6717 Win=20624 Len=0 TSval=7935342 TSecr=7936871





(a) In the 3-way handshake, which was the value of the MSS?

Si, el valor de MSS es de 1460 Bytes para los 2 (cliente, servidor).

(b) Did you notice any problem during the transmission according to the wiresharks?

Si, en el SimNet3 hay mensaje de error ICMP (fragmentation-needed).

(c) Can you see any ICMP message in SimNet3? Who is sending these error messages and why?
El mensaje de error lo envia R2 al host3 avisando que se envie paquetes mas pequeños.(indicando en el mensaje ICMP que ponga DF=0).

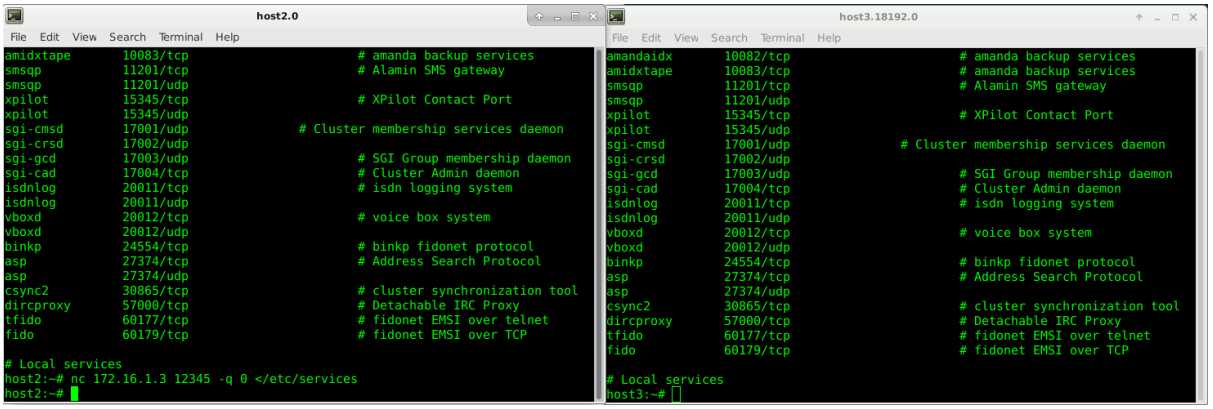
(d) Was the file transmitted properly? Who solved the problem?
Si, pero con retransmision. R2 ha resuelto el problema, ya que ha avisado al host3 que envie paquetes pequeños.

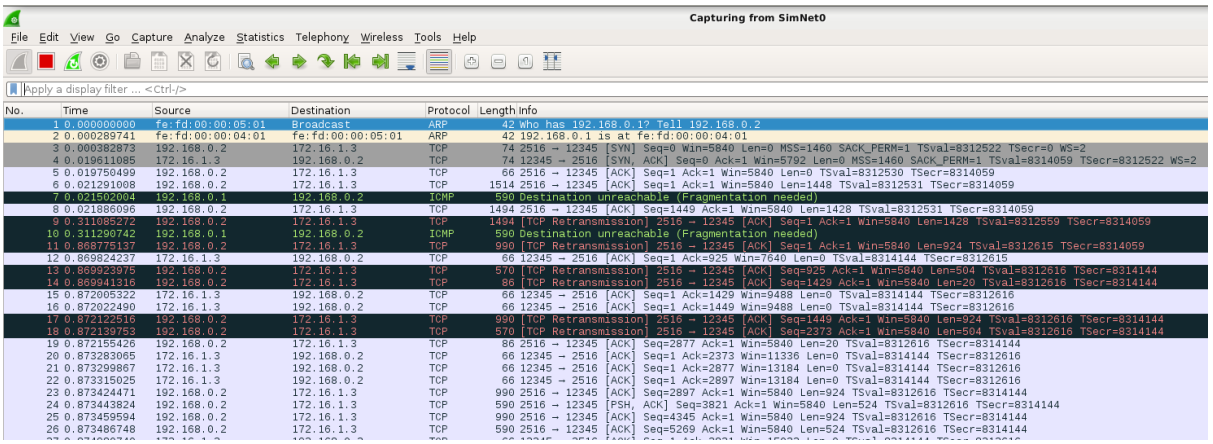
(e) Is this meaning that the MSS has changed during the transmission?.
No cambia, solo se ve la principio al establecer la comunicación,

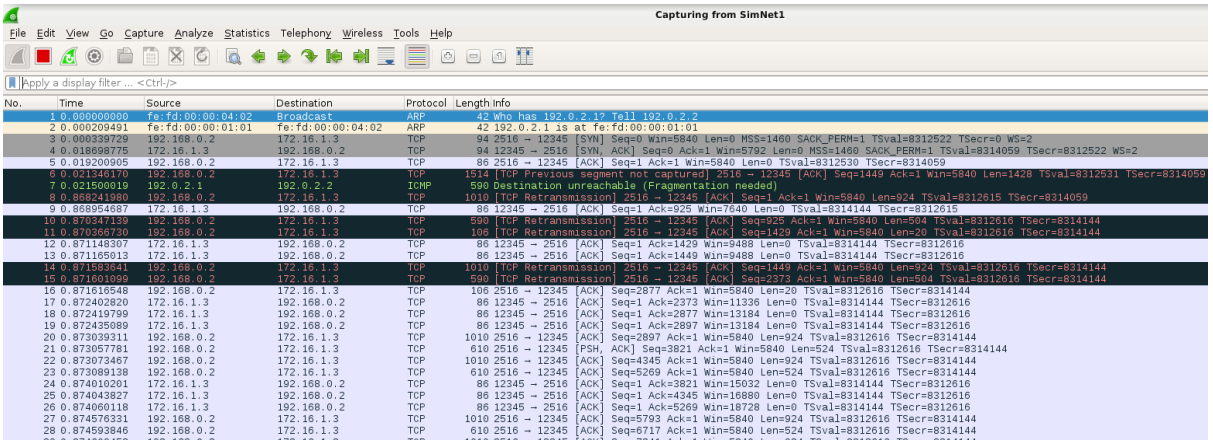
As you can see from the previous example, TCP is a reliable protocol. This means that it will retransmit lost information until it is properly received (or after a certain number of unsuccessful retransmissions). TCP is able to change the lenght of the segments in case that the network reports problems of size, but this does not mean that the MSS changes, as it is a fixed value.

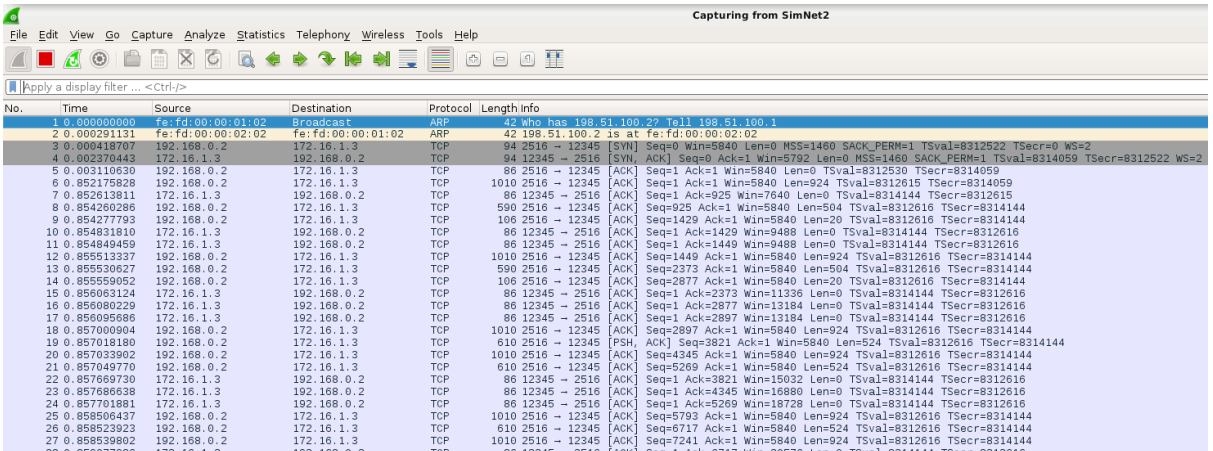
Obviously, files can be transmitted in both directions using netcat. In the previous example, we made a transmission in which the server transferred a file towards the client. Now we will test this same operation in the reverse direction.
1. Delete the kernel routing cache executing the label flushcache.
//from terminal
simctl iptunnel exec flushcache
2. Put all wiresharks listening traffic in all SimNet interfaces.

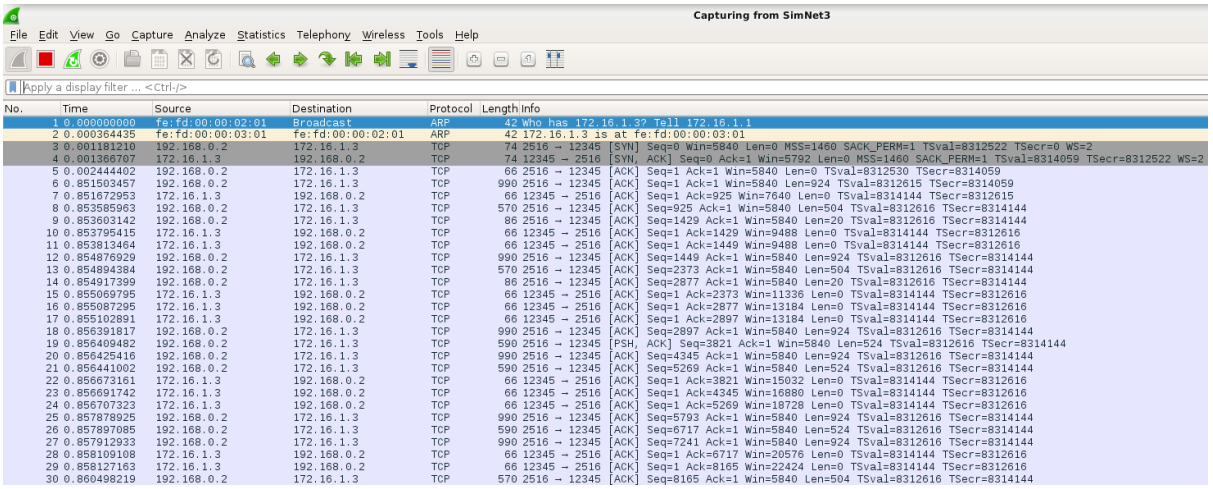
3. Start the netcat server in host3:
//from host3
nc -l -p 12345
4. Start the netcat client in host2, but in this case inject the file /etc/services of the client towards the server using the TCP connection.
//from host2
nc 172.16.1.3 12345 -q 0 </etc/services


host2.0
File Edit View Search Terminal Help
amidxtape 10083/tcp # amanda backup services
smsqp 11201/tcp # Alamin SMS gateway
smsqp 11201/udp
xpilot 15345/tcp # XPilot Contact Port
xpilot 15345/udp
sgi-cmsd 17001/udp # Cluster membership services daemon
sgi-crsd 17002/udp
sgi-gcd 17003/udp # SGI Group membership daemon
sgi-cad 17004/tcp # Cluster Admin daemon
isdnlog 20011/tcp # isdn logging system
isdnlog 20011/udp
vboxd 20012/tcp # voice box system
vboxd 20012/udp
binkp 24554/tcp # binkp fidonet protocol
asp 27374/tcp # Address Search Protocol
asp 27374/udp
csync2 30865/tcp # cluster synchronization tool
dircproxy 57000/tcp # Detachable IRC Proxy
tfido 60177/tcp # fidonet EMSI over telnet
fido 60179/tcp # fidonet EMSI over TCP
Local services
host2:~# nc 172.16.1.3 12345 -q 0 </etc/services
host2:~#
host3.18192.0
File Edit View Search Terminal Help
amandaidx 10082/tcp # amanda backup services
amidxtape 10083/tcp # amanda backup services
smsqp 11201/tcp # Alamin SMS gateway
smsqp 11201/udp
xpilot 15345/tcp # XPilot Contact Port
xpilot 15345/udp
sgi-cmsd 17001/udp # Cluster membership services daemon
sgi-crsd 17002/udp
sgi-gcd 17003/udp # SGI Group membership daemon
sgi-cad 17004/tcp # Cluster Admin daemon
isdnlog 20011/tcp # isdn logging system
isdnlog 20011/udp
vboxd 20012/tcp # voice box system
vboxd 20012/udp
binkp 24554/tcp # binkp fidonet protocol
asp 27374/tcp # Address Search Protocol
asp 27374/udp
csync2 30865/tcp # cluster synchronization tool
dircproxy 57000/tcp # Detachable IRC Proxy
tfido 60177/tcp # fidonet EMSI over telnet
fido 60179/tcp # fidonet EMSI over TCP
Local services
host3:~#


Capturing from SimNet0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 fe:fd:00:00:05:01 Broadcast ARP 42 Who has 192.168.0.1? Tell 192.168.0.2
2 0.000289741 fe:fd:00:00:04:01 fe:fd:00:00:05:01 ARP 42 192.168.0.1 is at fe:fd:00:00:04:01
3 0.000382873 192.168.0.2 172.16.1.3 TCP 74 2516 → 12345 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=8312522 TSecr=0 WS=2
4 0.019611085 172.16.1.3 192.168.0.2 TCP 74 12345 → 2516 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=8314059 TSecr=8312522 WS=2
5 0.019750499 192.168.0.2 172.16.1.3 TCP 66 2516 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=8312530 TSecr=8314059
6 0.021291008 192.168.0.2 172.16.1.3 TCP 1514 2516 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=1448 TSval=8312531 TSecr=8314059
7 0.021502004 192.168.0.1 192.168.0.2 ICMP 590 Destination unreachable (Fragmentation needed)
8 0.021886096 192.168.0.2 172.16.1.3 TCP 1494 2516 → 12345 [ACK] Seq=1449 Ack=1 Win=5840 Len=1428 TSval=8312531 TSecr=8314059
9 0.311085272 192.168.0.2 172.16.1.3 TCP 1494 [TCP Retransmission] 2516 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=1428 TSval=8312559 TSecr=8314059
10 0.311290742 192.168.0.1 192.168.0.2 ICMP 590 Destination unreachable (Fragmentation needed)
11 0.869775137 192.168.0.2 172.16.1.3 TCP 990 [TCP Retransmission] 2516 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=924 TSval=8312615 TSecr=8314059
12 0.869824237 172.16.1.3 192.168.0.2 TCP 66 12345 → 2516 [ACK] Seq=1 Ack=925 Win=7640 Len=0 TSval=8314144 TSecr=8312615
13 0.869923975 192.168.0.2 172.16.1.3 TCP 570 [TCP Retransmission] 2516 → 12345 [ACK] Seq=925 Ack=1 Win=5840 Len=504 TSval=8312616 TSecr=8314144
14 0.869941316 192.168.0.2 172.16.1.3 TCP 86 [TCP Retransmission] 2516 → 12345 [ACK] Seq=1429 Ack=1 Win=5840 Len=20 TSval=8312616 TSecr=8314144
15 0.872005322 172.16.1.3 192.168.0.2 TCP 66 12345 → 2516 [ACK] Seq=1 Ack=1429 Win=9488 Len=0 TSval=8314144 TSecr=8312616
16 0.872022490 172.16.1.3 192.168.0.2 TCP 66 12345 → 2516 [ACK] Seq=1 Ack=1449 Win=9488 Len=0 TSval=8314144 TSecr=8312616
17 0.872122516 192.168.0.2 172.16.1.3 TCP 990 [TCP Retransmission] 2516 → 12345 [ACK] Seq=1449 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
18 0.872139753 192.168.0.2 172.16.1.3 TCP 570 [TCP Retransmission] 2516 → 12345 [ACK] Seq=2373 Ack=1 Win=5840 Len=504 TSval=8312616 TSecr=8314144
19 0.872155426 192.168.0.2 172.16.1.3 TCP 86 2516 → 12345 [ACK] Seq=2877 Ack=1 Win=5840 Len=20 TSval=8312616 TSecr=8314144
20 0.873283065 172.16.1.3 192.168.0.2 TCP 66 12345 → 2516 [ACK] Seq=1 Ack=2373 Win=11336 Len=0 TSval=8314144 TSecr=8312616
21 0.873299867 172.16.1.3 192.168.0.2 TCP 66 12345 → 2516 [ACK] Seq=1 Ack=2877 Win=13184 Len=0 TSval=8314144 TSecr=8312616
22 0.873315025 172.16.1.3 192.168.0.2 TCP 66 12345 → 2516 [ACK] Seq=1 Ack=2897 Win=13184 Len=0 TSval=8314144 TSecr=8312616
23 0.873424471 192.168.0.2 172.16.1.3 TCP 990 2516 → 12345 [ACK] Seq=2897 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
24 0.873443824 192.168.0.2 172.16.1.3 TCP 590 2516 → 12345 [PSH, ACK] Seq=3821 Ack=1 Win=5840 Len=524 TSval=8312616 TSecr=8314144
25 0.873459594 192.168.0.2 172.16.1.3 TCP 990 2516 → 12345 [ACK] Seq=4345 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
26 0.873486748 192.168.0.2 172.16.1.3 TCP 590 2516 → 12345 [ACK] Seq=5269 Ack=1 Win=5840 Len=524 TSval=8312616 TSecr=8314144


Capturing from SimNet1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 fe:fd:00:00:04:02 Broadcast ARP 42 Who has 192.0.2.1? Tell 192.0.2.2
2 0.000209491 fe:fd:00:00:01:01 fe:fd:00:00:04:02 ARP 42 192.0.2.1 is at fe:fd:00:00:01:01
3 0.000339729 192.168.0.2 172.16.1.3 TCP 94 2516 → 12345 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=8312522 TSecr=0 WS=2
4 0.018698775 172.16.1.3 192.168.0.2 TCP 94 12345 → 2516 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=8314059 TSecr=8312522 WS=2
5 0.019200905 192.168.0.2 172.16.1.3 TCP 86 2516 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=8312530 TSecr=8314059
6 0.021346170 192.168.0.2 172.16.1.3 TCP 1514 [TCP Previous segment not captured] 2516 → 12345 [ACK] Seq=1449 Ack=1 Win=5840 Len=1428 TSval=8312531 TSecr=8314059
7 0.021500019 192.0.2.1 192.0.2.2 ICMP 590 Destination unreachable (Fragmentation needed)
8 0.868241980 192.168.0.2 172.16.1.3 TCP 1010 [TCP Retransmission] 2516 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=924 TSval=8312615 TSecr=8314059
9 0.868954687 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=925 Win=7640 Len=0 TSval=8314144 TSecr=8312615
10 0.870347139 192.168.0.2 172.16.1.3 TCP 590 [TCP Retransmission] 2516 → 12345 [ACK] Seq=925 Ack=1 Win=5840 Len=504 TSval=8312616 TSecr=8314144
11 0.870366730 192.168.0.2 172.16.1.3 TCP 106 [TCP Retransmission] 2516 → 12345 [ACK] Seq=1429 Ack=1 Win=5840 Len=20 TSval=8312616 TSecr=8314144
12 0.871148307 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=1429 Win=9488 Len=0 TSval=8314144 TSecr=8312616
13 0.871165013 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=1449 Win=9488 Len=0 TSval=8314144 TSecr=8312616
14 0.871503841 192.168.0.2 172.16.1.3 TCP 1010 [TCP Retransmission] 2516 → 12345 [ACK] Seq=1449 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
15 0.871601099 192.168.0.2 172.16.1.3 TCP 590 [TCP Retransmission] 2516 → 12345 [ACK] Seq=2373 Ack=1 Win=5840 Len=504 TSval=8312616 TSecr=8314144
16 0.871616548 192.168.0.2 172.16.1.3 TCP 106 2516 → 12345 [ACK] Seq=2877 Ack=1 Win=5840 Len=20 TSval=8312616 TSecr=8314144
17 0.872402820 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=2373 Win=11336 Len=0 TSval=8314144 TSecr=8312616
18 0.872419799 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=2877 Win=13184 Len=0 TSval=8314144 TSecr=8312616
19 0.872435089 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=2897 Win=13184 Len=0 TSval=8314144 TSecr=8312616
20 0.873039311 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=2897 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
21 0.873057781 192.168.0.2 172.16.1.3 TCP 610 2516 → 12345 [PSH, ACK] Seq=3821 Ack=1 Win=5840 Len=524 TSval=8312616 TSecr=8314144
22 0.873073467 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=4345 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
23 0.873089138 192.168.0.2 172.16.1.3 TCP 610 2516 → 12345 [ACK] Seq=5269 Ack=1 Win=5840 Len=524 TSval=8312616 TSecr=8314144
24 0.874010201 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=3821 Win=15032 Len=0 TSval=8314144 TSecr=8312616
25 0.874043827 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=4345 Win=16880 Len=0 TSval=8314144 TSecr=8312616
26 0.874060118 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=5269 Win=18728 Len=0 TSval=8314144 TSecr=8312616
27 0.874576331 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=5793 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
28 0.874593846 192.168.0.2 172.16.1.3 TCP 610 2516 → 12345 [ACK] Seq=6717 Ack=1 Win=5840 Len=524 TSval=8312616 TSecr=8314144


Capturing from SimNet2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 fe:fd:00:00:01:02 Broadcast ARP 42 Who has 198.51.100.2? Tell 198.51.100.1
2 0.000291131 fe:fd:00:00:02:02 fe:fd:00:00:01:02 ARP 42 198.51.100.2 is at fe:fd:00:00:02:02
3 0.000418707 192.168.0.2 172.16.1.3 TCP 94 2516 → 12345 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=8312522 TSecr=0 WS=2
4 0.002370443 172.16.1.3 192.168.0.2 TCP 94 12345 → 2516 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=8314059 TSecr=8312522 WS=2
5 0.003110630 192.168.0.2 172.16.1.3 TCP 86 2516 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=8312530 TSecr=8314059
6 0.852175828 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=1 Ack=1 Win=5840 Len=924 TSval=8312615 TSecr=8314059
7 0.852613811 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=925 Win=7640 Len=0 TSval=8314144 TSecr=8312615
8 0.854260286 192.168.0.2 172.16.1.3 TCP 590 2516 → 12345 [ACK] Seq=925 Ack=1 Win=5840 Len=504 TSval=8312616 TSecr=8314144
9 0.854277793 192.168.0.2 172.16.1.3 TCP 106 2516 → 12345 [ACK] Seq=1429 Ack=1 Win=5840 Len=20 TSval=8312616 TSecr=8314144
10 0.854831810 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=1429 Win=9488 Len=0 TSval=8314144 TSecr=8312616
11 0.854849459 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=1449 Win=9488 Len=0 TSval=8314144 TSecr=8312616
12 0.855513337 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=1449 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
13 0.855530627 192.168.0.2 172.16.1.3 TCP 590 2516 → 12345 [ACK] Seq=2373 Ack=1 Win=5840 Len=504 TSval=8312616 TSecr=8314144
14 0.855559052 192.168.0.2 172.16.1.3 TCP 106 2516 → 12345 [ACK] Seq=2877 Ack=1 Win=5840 Len=20 TSval=8312616 TSecr=8314144
15 0.856063124 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=2373 Win=11336 Len=0 TSval=8314144 TSecr=8312616
16 0.856080229 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=2877 Win=13184 Len=0 TSval=8314144 TSecr=8312616
17 0.856095686 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=2897 Win=13184 Len=0 TSval=8314144 TSecr=8312616
18 0.857000904 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=2897 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
19 0.857018180 192.168.0.2 172.16.1.3 TCP 610 2516 → 12345 [PSH, ACK] Seq=3821 Ack=1 Win=5840 Len=524 TSval=8312616 TSecr=8314144
20 0.857033902 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=4345 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
21 0.857049770 192.168.0.2 172.16.1.3 TCP 610 2516 → 12345 [ACK] Seq=5269 Ack=1 Win=5840 Len=524 TSval=8312616 TSecr=8314144
22 0.857669730 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=3821 Win=15032 Len=0 TSval=8314144 TSecr=8312616
23 0.857686638 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=4345 Win=16880 Len=0 TSval=8314144 TSecr=8312616
24 0.857701881 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=5269 Win=18728 Len=0 TSval=8314144 TSecr=8312616
25 0.858506437 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=5793 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
26 0.858523923 192.168.0.2 172.16.1.3 TCP 610 2516 → 12345 [ACK] Seq=6717 Ack=1 Win=5840 Len=524 TSval=8312616 TSecr=8314144
27 0.858539802 192.168.0.2 172.16.1.3 TCP 1010 2516 → 12345 [ACK] Seq=7241 Ack=1 Win=5840 Len=924 TSval=8312616 TSecr=8314144
28 0.859077986 172.16.1.3 192.168.0.2 TCP 86 12345 → 2516 [ACK] Seq=1 Ack=6717 Win=20576 Len=0 TSval=8314144 TSecr=8312616



(a) Which are the commands that you used to do the previous behavior?
host3 como server y host2 como cliente con grep del fichero etc/services.

(b) Did you notice any problem during the transmission according to the wiresharks?
Hay acks duplicados, y en SimNet0 y SimNet1 hay mensajes de error ICMP ().

(c) Can you see any ICMP message in SimNet0 and SimNet1? Who is sending these error messages and why?
En SimNet0 hay 2 mensajes (fragmentation needed) que lo envia el R1 al host2.
En SimNet1 hay 1 mensaje que lo envia RC al R1.
Estos mensajes son enviados porque los paquetes necesitan ser fragmentados.

(d) Was the file transmitted properly? Who solved the problem?
Si, porque TCP retransmite hasta que los paquetes llegan correctamente al destino. RC informa al R1 y R1 avisa al host2, y host2 resuelve el problema, enviando paquetes fragmentados.

(e) According to the transmission, who is the TCP client and who is the TCP server?
host3 server, host2 cliente.

Again, TCP solved the problem by adapting the lenght of the segments to the network conditions and retransmitting lost information. Notice that the TCP client is the host that started the TCP dialogue by sending the first message SYN, and the TCP server is the host that answered by sending the SYN-ACK message, no matter of the direction of the data transmission.

0.8.3 TCP, tunnels and filtering

1. Let us imagine that the network administrator of R1 wants to protect this router against typical attacks that use ICMP. For this reason, it avoids sending ICMP messages to this router. So, execute the following command to filter the ICMP traffic whose destination is R1:

//from R1
iptables −A INPUT −p icmp −j DROP

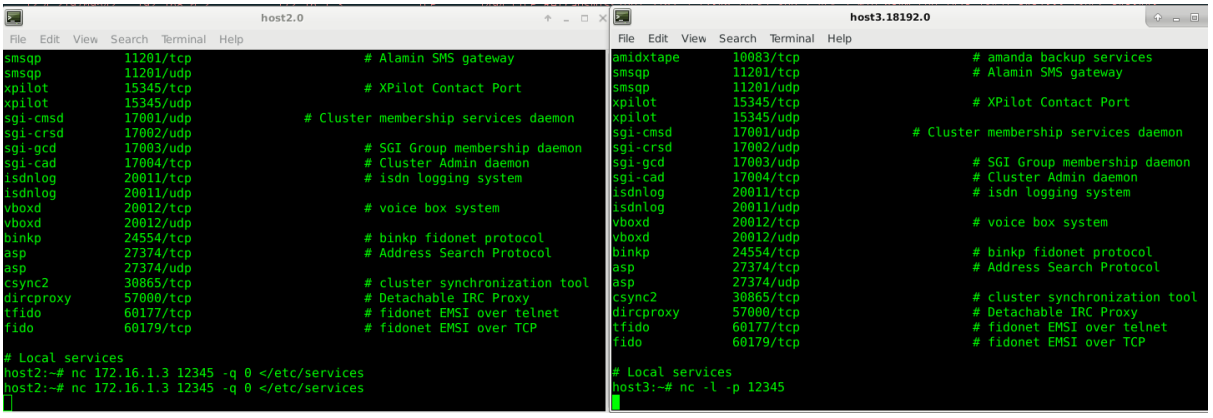
2. Delete the kernel routing cache executing the label flushcache.

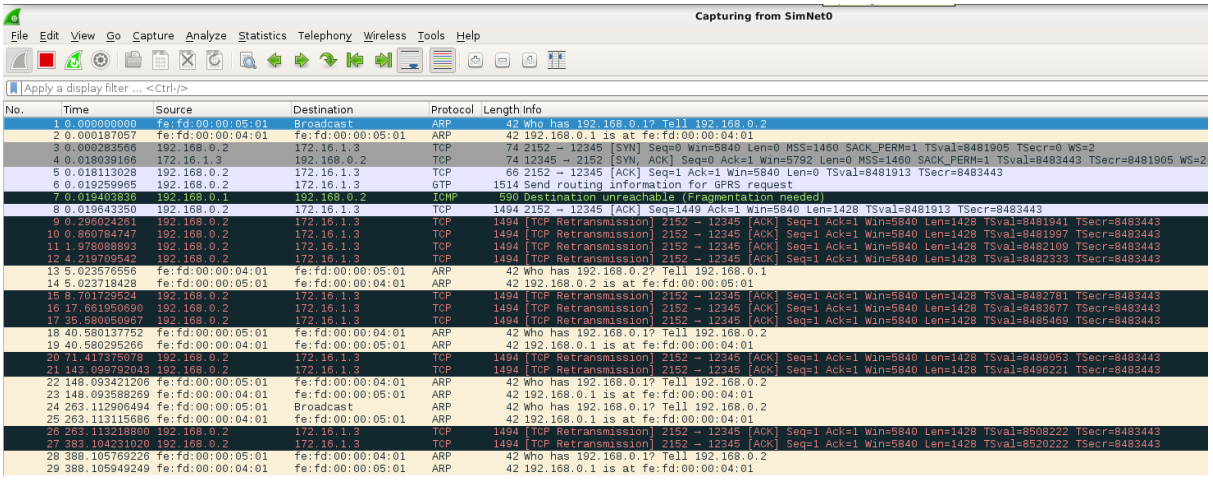
//from terminal
simctl iptunnel exec flushcache

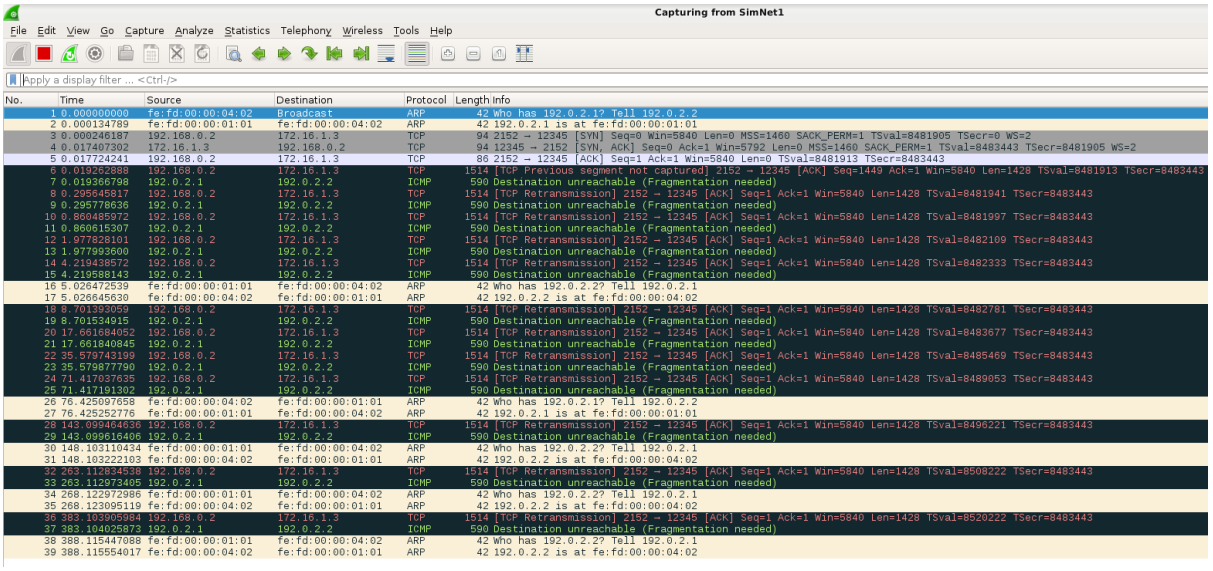
3. Put all wiresharks listening traffic in all SimNet interfaces.
4. Again using netcat, send the file /etc/services from host2 (acting as client) towards host3 (acting as server).

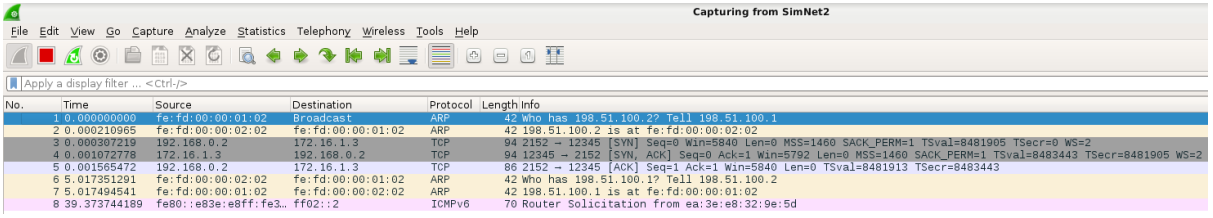
//from host3
nc -l -p 12345

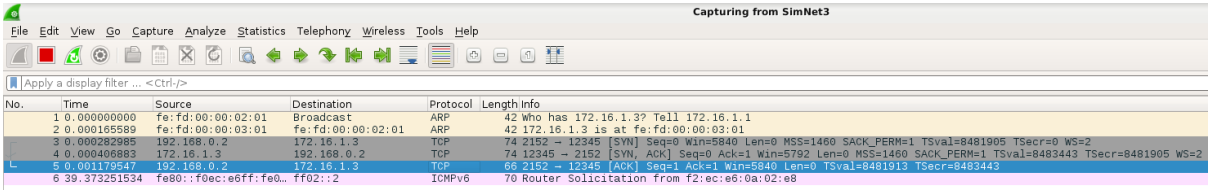
//from host2
nc 172.16.1.3 12345 -q 0 </etc/services











(a) Can you see the 3-way handshake? Did you notice problems with these 3 initial messages?

Si, no hay ningun problema con el handshake, porque no superan la MTU.

(b) Did you notice problems with data segments? Why? Which is the difference with the previous ones?
Si, en la SimNet0 vemos que necesitan paquetes grandes y se necesita fragmentar. Pero ahora el host2 no soluciona el problema porque no le llegan los mensajes ICMP. porque se hemos indicado que las tramas icmp se tiren al llegar al R1.

(c) Can you see any ICMP message in SimNet1? And in SimNet0?
Si, en los 2 casos son mensajes ICMP de fragmentation needed porque se envian paquetes superior a la MTU de R1 y RC.

(d) Was the file transmitted properly? Why?
No porque los paquetes se pierden, como R1 ignora los mensajes ICMP, al host2 nunca le llegan ack y host2 las retransmite y no termina nunca porque nunca recibe un ack.

(e) What will happen if instead of the INPUT chain, we use the FORWARD chain in the filtering rule?
Si usaramos la chain FORWARD (filtra mensajes que tendría que "pasar" más adelante, pero en este caso RC envía directamente a R1) en lugar de la chain INPUT creo que funcionaría, porque realmente R1 no está "pasándole" los mensajes ICMP de RC a host2, sino que en base a los soft state que guarda, el propio R1 genera estos mensajes (pero ahora no tiene el soft state guardado porque rechaza los ICMP antes de procesarlos)

As you can see, the client is not receiving feedback from the errors of the network due to the filtering rule, so it is unable to addapt the TCP transmission to the conditions of the network. TCP assumes that losses are due to congestion, so it will try to retransmit lost segments just waiting more and more time.

0.8.4 Changing the MSS

the MSS counts only DATA octets in the segment, and it does not count the TCP header or the IP header.

ip route add ROUTE advmss VALUE
where ROUTE is the route we want to add (more information using ip route show), and VALUE is the "Advertised MSS" in bytes. In case the route exists, instead of add use change

Change the MSS in the host

1. Delete the kernel routing cache executing the label flushcache.
//from terminal
simctl iptunnel exec flushcache

2. Put all wiresharks listening traffic in all SimNet interfaces.
3. Identify which are the proper parameters to change the mss properly:
• Which is the value of the advertised MSS to avoid fragmentation in path? Notice that the MTU in SimNet2=996.
MSS = 996B (MTUrestrictiva) - 20B (HIPInner) - 20B (HIPOuter) - 20B (Htcp) = 936B.

• Which message of the 3-way handshake of TCP should be change? Why?
Hemos de cambiar el SYN/ACK porque es el que contiene el valor de MSS del servidor , que es el que genera problema.

• If we are changing the MSS in a host, which is the host where we should execute the command?
el host3 porque es el que tiene la información del MSS del cliente.

4. Now that you fully know all the parameters to properly change the MSS, use ip route to change it.
//from host3
ip route add 192.168.0.0/24 via 172.16.1.1 advmss 936

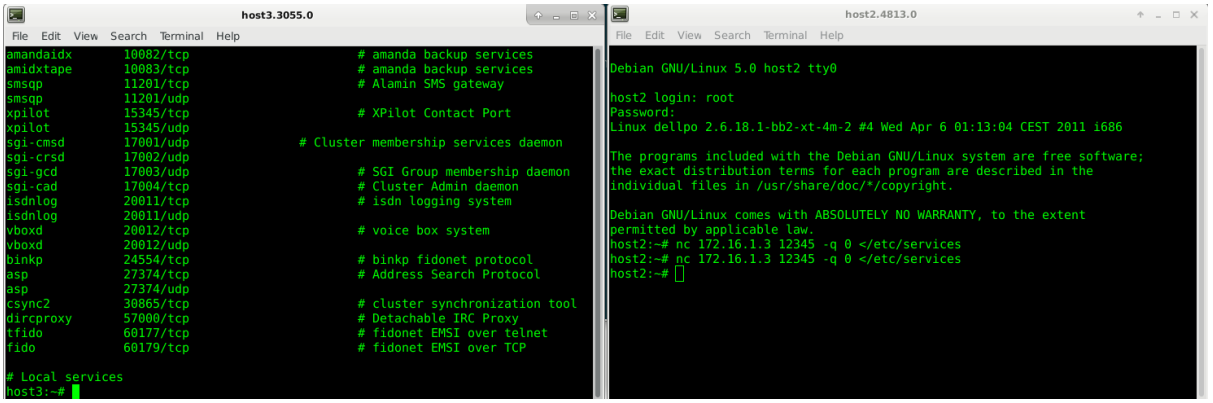
5. Using netcat again, send the file /etc/services from host2 (acting as client) towards host3 (acting as server).

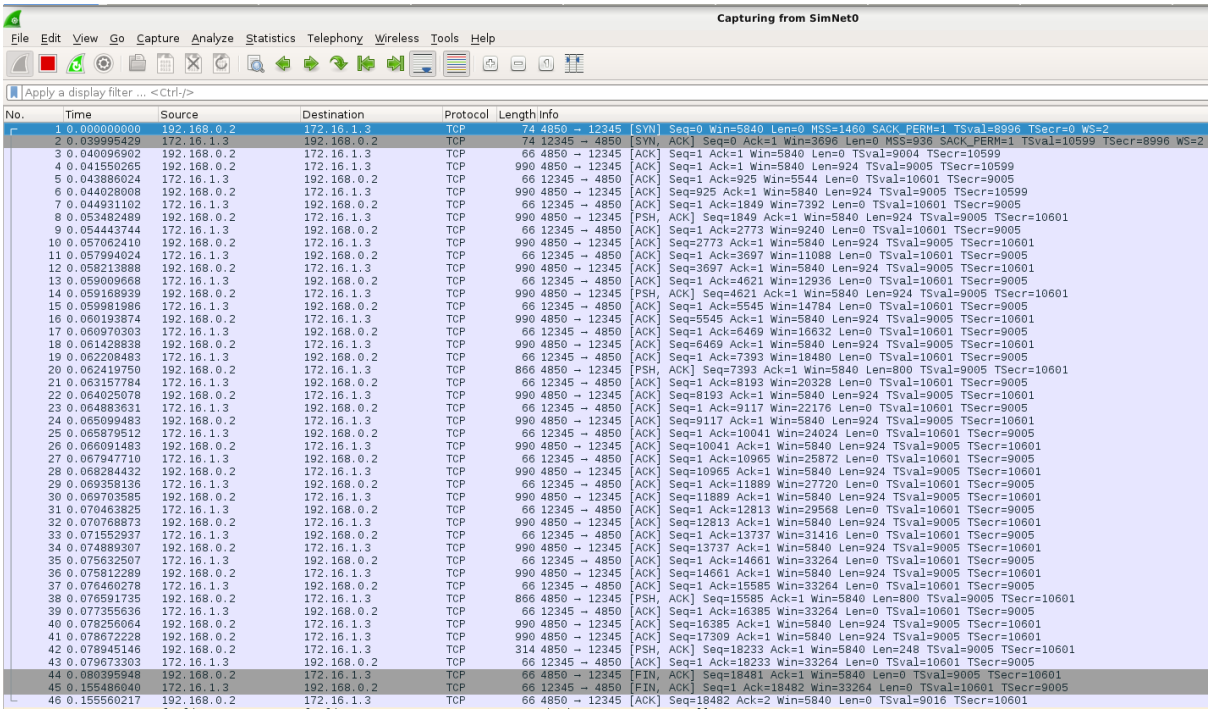
//from host3

nc -l -p 12345

//from host2

nc 172.16.1.3 12345 -q 0 </etc/services





en todas las SimNetx vemos este flujo.

Salvo una diferencia, la MSS de SimNet3 es 936, que es lo que hemos cambiado.

(a) Did you notice problems with data segments?

No ha habido ningun problema.

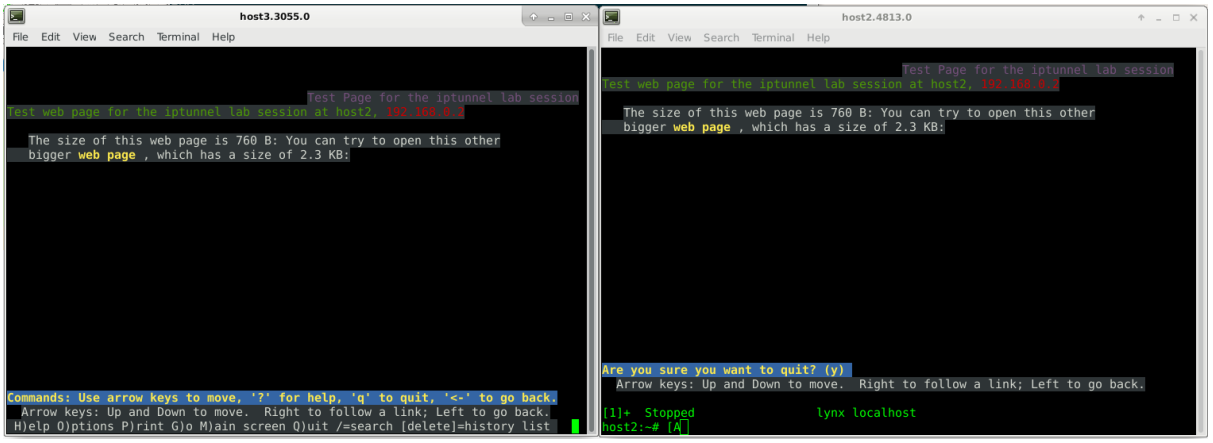
(b) Can you see any ICMP message in SimNet1 and SimNet0?
No porque no ha habido errores, poque no ha habido errores de transmission.

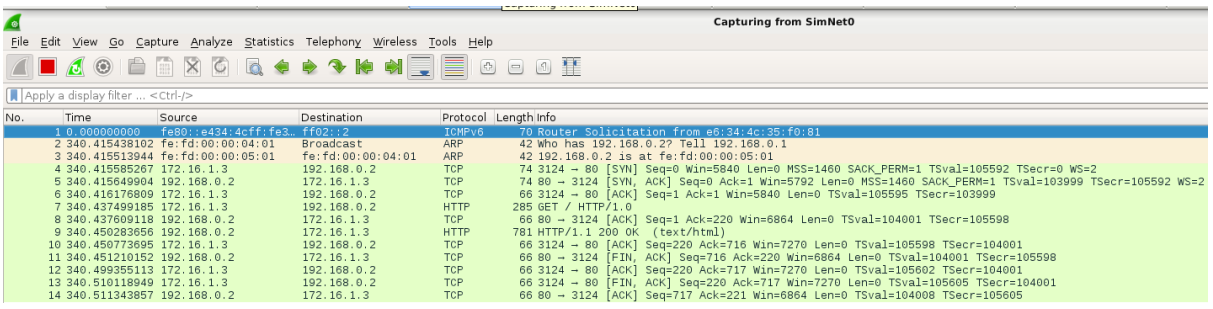
(c) Was the file transmitted properly?
Yes.

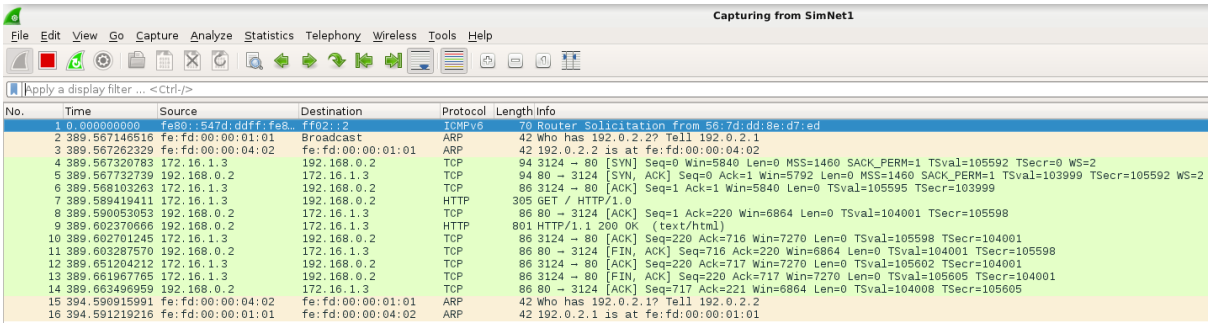
Prior to continue with the following test, delete the route that you configured in the host to change the MSS.
//from host3
ip route delete 192.168.0.0/24 via 172.16.1.1

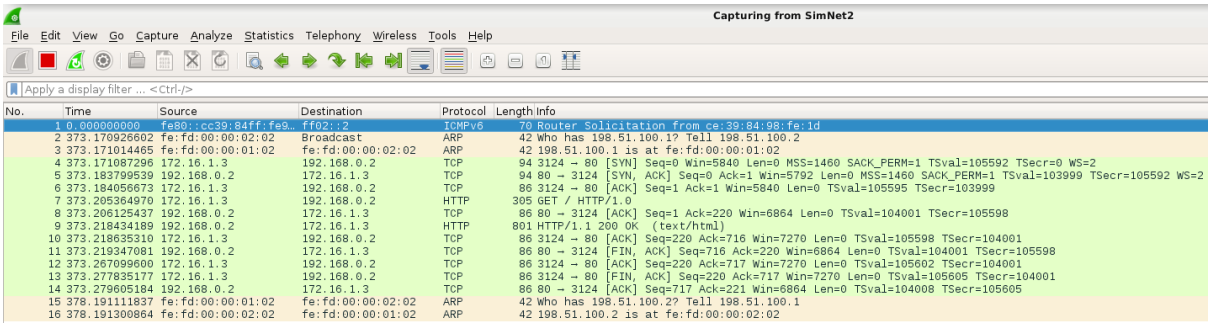
Change the MSS in the router
1. Delete the kernel routing cache executing the label flushcache.
//from terminal
simctl iptunnel exec flushcache
2. Put all wiresharks listening traffic in all SimNet interfaces.
3. Connect to the web server of host2 using lynx from host3:
//from host2
lynx localhost

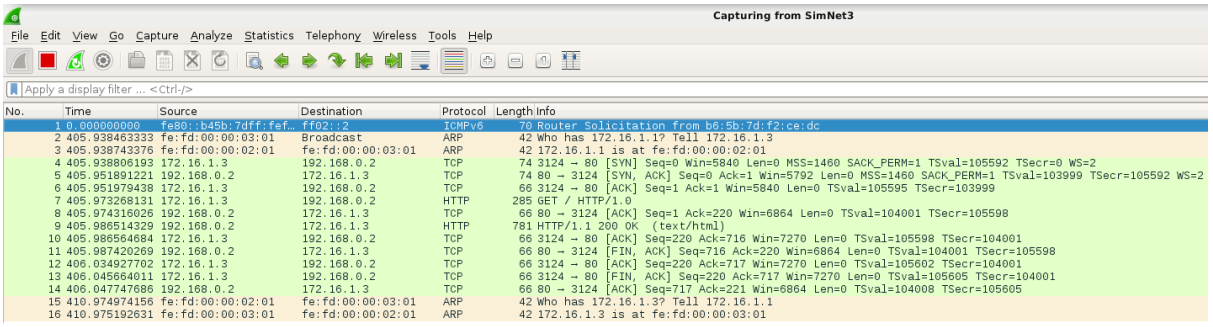
//from host3
lynx 192.168.0.2











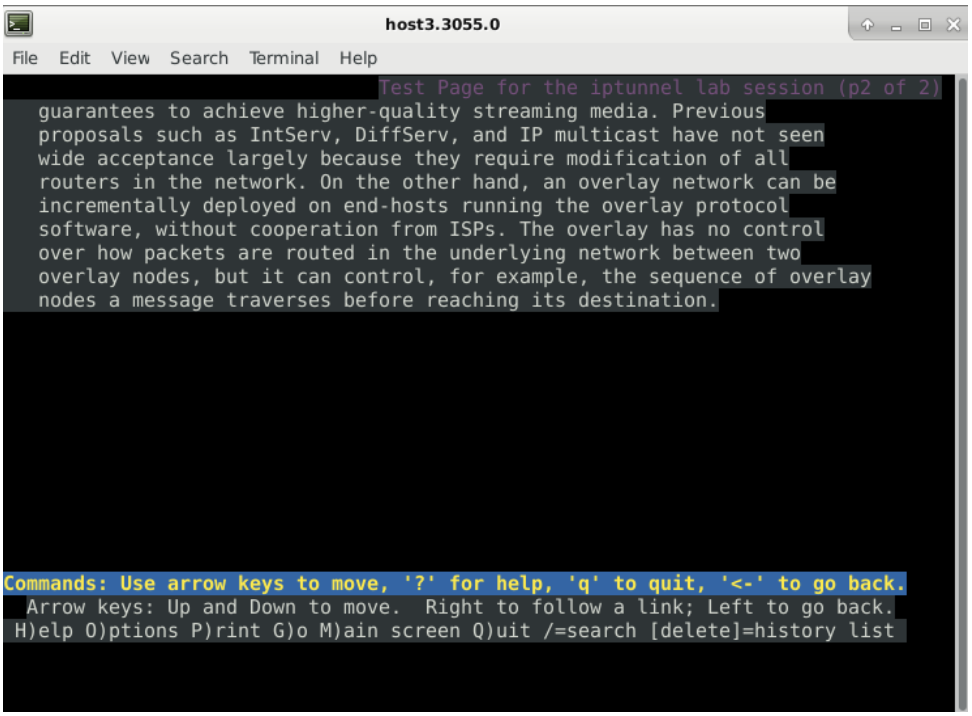
(a) Can you see the main web page of host2?

Si

(b) Can you see any error message in the wiresharks?

No

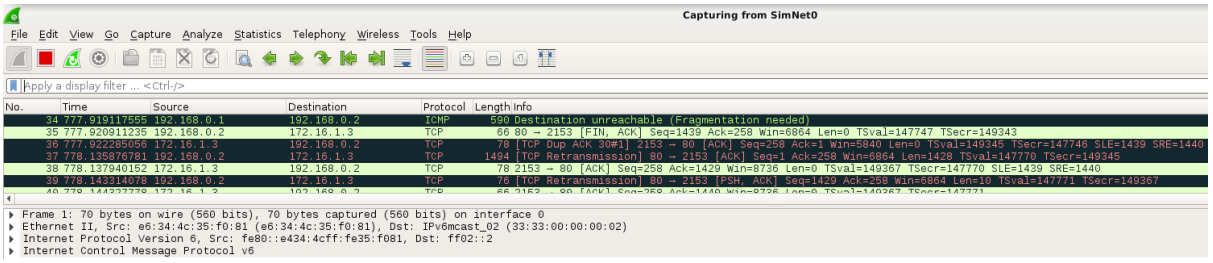
As you can see there is no problem to see the main web page of this host. However, browse the web in order to see the other local web page:



(a) Did you noticed any error?

Si, podemos ver que no carga la segunda parte.

(b) What do you see in the wiresharks?



Ahora vemos que hay mensajes ICMPs, enviado por R1 al host2.
Retransmisiones enviados por host2 al host3.
Tambien tenemos Acks, duplicados.
El problema es similar al de hace dos apartados, habría que fragmentar pero R1 descarta los mensajes ICMP así que host2 no sabe que tiene que fragmentar.

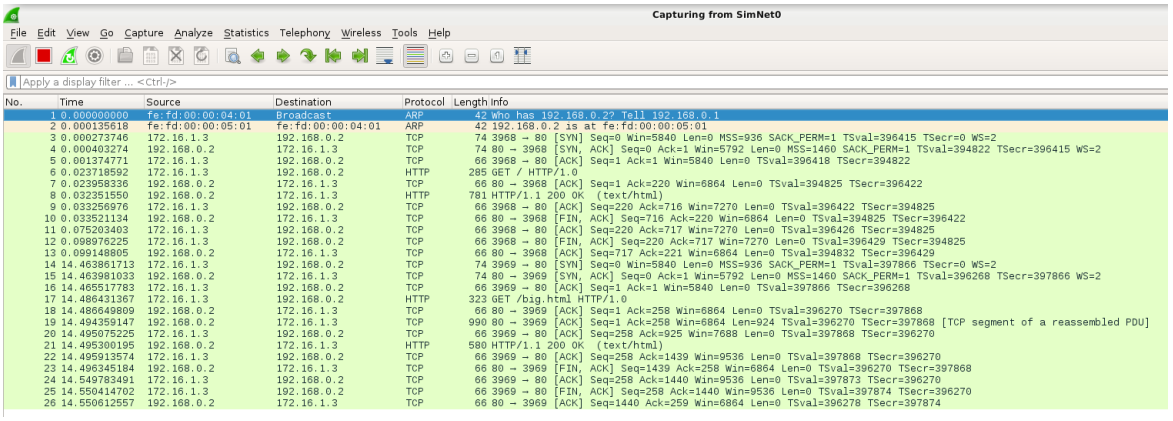
(c) Why do you think that the main web page is working, but the second one cannot be downloaded?

Por el tamaño de las paginas. La principal no tiene muchos datos (es mas ligera) que la pagina siguiente.(se puede ver en la length de los paquetes.)

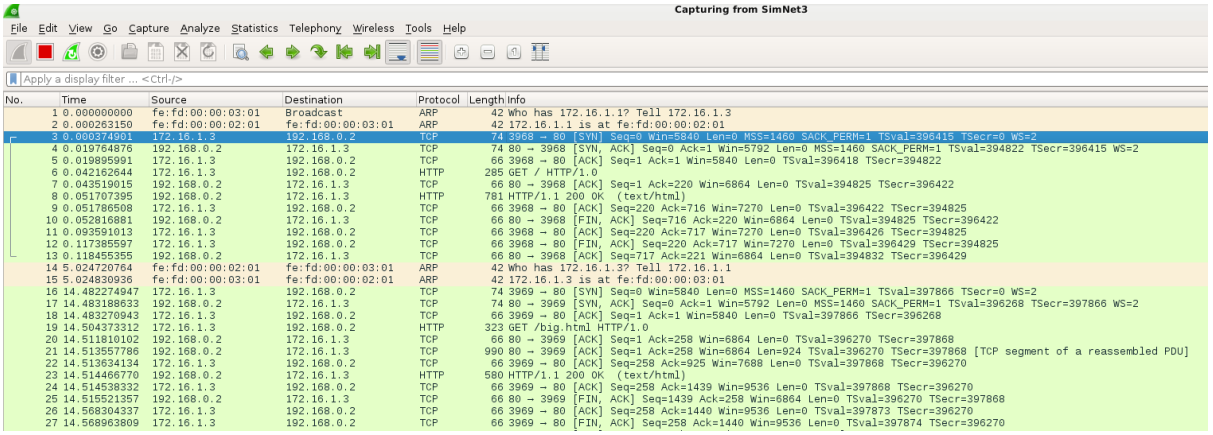
(d) Try to solve the problem by changing the MSS in R2 using the MANGLE table of iptables. Remember that you have to change the MSS in the proper message of the 3-way handshake.

el cambio de mms se hace como en el ejercicio anterior,
pero ahora el cliente es el host3 .
(ya ta hecho)

//from R2
iptables -t mangle -A FORWARD -o tunnel0 -p tcp --syn -j TCPMSS --set-mss 936



Ahora, no hay retransmisiones
En todas las SimNetx vemos este flujo. Salvo una diferencia, en SimNet3 MMS es 1452.



We hope that all these experiments helped you to understand how complicated is to solve problems of communication networks when using tunnels, even in the case of using the most simplistic IPIP tunnel. In following lab sessions, maybe you will be asked to create GRE tunnels, so try to remember all these concepts.