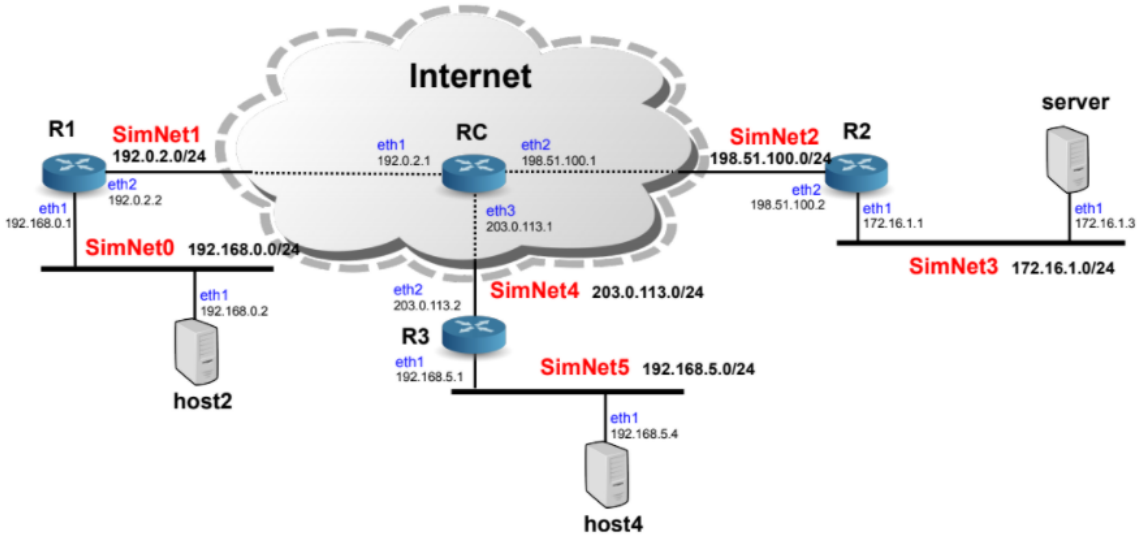


To start the scenario:

```
host# simctl ipmulticast start
```

En multicast, el camp de la trama IP que conté @destí, conté una @IP que identifica a un grup de receptors que volen rebre el flux multicast. En IPv4, el bloc 224.0.0.0/4 (classe D) esta reservat per a l'ús de multicast i per al Link-Local. 224.0.0.1 identifica a tots els hosts i la 224.0.0.2 a tots els routers.

IP Multicast addresses and DNS

```
HOST$ host all-systems.mcast.net
```

As you can see, the command host allows to send requests to the DNS server. In this particular case, we are asking to the DNS the address that corresponds to the name all-systems.mcast.net. As you can see, this name corresponds to the address 224.0.0.1. As we mentioned previously, this particular address is managed by the IANA, and it identifies all (multicast) hosts on the same network segment (TTL=1). Later, you will ping this address.

```
telem@debian:~$ host all-systems.mcast.net
all-systems.mcast.net has address 224.0.0.1
```

- Execute the following command in your HOST:

```
HOST$ host 224.0.0.22
```

As you can see, this address matches the name igmp.mcast.net, as it is used to identify routers using the Internet Group Management Protocol (IGMP) in its current version (v3). Later during the lab session, you will see some IGMP messages that use this address.

```
telem@debian:~$ host 224.0.0.22
22.0.0.224.in-addr.arpa domain name pointer igmp.mcast.net.
```

MAC multicast addresses

Per encapsular els paquets IP es necessita una @MAC destí multicast a la trama Ethernet. Només necessites 28 bits (@IP té 32bits) ja que els primers estan fixats (1110).
Per construir una @MAC multicast, esta prefixat que els primers 24bits siguin 01:00:5E que seran els bits menys significants.

Now you are going to send multicast traffic. Notice that multicast is mainly aimed to send a flow of packets towards a group of receivers, (for instance video streaming). However, just for simplicity, you will start sending a simple ping.
So:

- Put a wireshark listening in SimNet3.
- Execute the following command in the server:

```
server# ping -c 1 224.0.0.1
```

```
1 0.000000000   172.16.1.3          224.0.0.1           ICMP       98 Echo (ping) request  id=0xc402, seq=1/25
```

1. How many ICMP packets can you see in SimNet3? What type of messages?
   Podem veure un sol missatge ICMP echo-request

2. In the IP header, which is the TTL of this packets? What does this mean?
   El TTL d'aquest paquet TTL=1, això significa que només pot passar per un node (R2) més abans de ser descartat per la xarxa o tornar al seu origen. Per tant, només podrà fer un intent, si aquest falla, es descarta el paquet.

3. In the IP header, which are the source and destination addresses?
   Les @IP d'aquest paquet són: @s:172.16.1.3
   @d: 224.0.0.1 (server → hosts multicast)

4. In the Ethernet header, which are the source and destination addresses?
   Les adreces MAC són: @s: fe:fd:00:00:03:01
   @d: 01:00:5e:00:00:01 (IPv4mcast_01)

5. Can you see the direct mapping in the MAC destination address?
   Podem observar Mapeig Directe en l'@MACd:

@IPdmcst		1110 0000	0	000 0000	0000 0000	0000 0001
@MACdm	**0000 0001**	**0000 0000**	**0101 1110**	**0**000 0000	0000 0000	0000 0001

As you can see, the destination MAC address is constructed by using the prefix 01:00:5E (in hex). The 24 remaining bits correspond to 0+23 least significant bits of the multicast IP address. In the particular case of the IP address 224.0.0.1, this corresponds to the MAC address 01:00:5E:00:00:01.

- Put a wireshark listening in SimNet3.
- Execute the following command in the server:

```
server# ping -c 1 232.0.0.1
```

```
1 0.000000000   172.16.1.3          232.0.0.1           ICMP       98 Echo (ping) request  id=0xc502, seq=1/25
```

1. In this ICMP packet, which is the destination MAC addresses?
   @MACd: 01:00:5e:00:00:01 (la mateixa que abans)
2. Can you see any ambiguity in this mapping? Which other addresses cause also this ambiguity?
   Podem observar una ambigüitat en el procés de Direct-Mapping, ja que assigna la mateix @MAC a dos @IP diferents. Totes les adreces multicast que tinguin una estructura X.X000 0000.0000 0000.0000 0001 tindran aquest problema.
3. How can the system resolve this ambiguity?

El sistema resol aquesta ambigüitat fent que el host comprovi l'adreça m/c IP i descartar els paquets amb una @IP no registrada.

I suppose that you also noticed that there is no reply message for these messages. Just remark that the main purpose of multicasting is not sending pings, and most multicast applications do not require any answer. Anyway, as we want to send multicast pings and receive and answer for them, we have to disable a flag in your Linux system in order not to ignore broadcast and multicast ICMP packets.

- Put a wireshark listening in SimNet3.
- Execute the following commands in the server:

```
server# echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
server# ping -c3 224.0.0.1
```

```
server:~# ping -c1 224.0.0.1
PING 224.0.0.1 (224.0.0.1) 56(84) bytes of data.
64 bytes from 172.16.1.3: icmp_seq=1 ttl=64 time=0.024 ms

--- 224.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.024/0.024/0.024/0.000 ms
```

1	0.000000000	172.16.1.3	224.0.0.1	ICMP	98 Echo (ping) request id=0xc802, seq=1/25

1. Is the ping working? Who is answering to the ping?
   Ara els ping els respon 172.16.1.3 (ell mateix)

2. Can you see any reply message in SimNet3?
   No podem veure els echo-reply ja que el server es contesta a ell mateix.

- Now dissable the same flag in R2, and ping from server:

```
R2# echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
server# ping -c3 224.0.0.1
```

```
server:~# ping -c3 224.0.0.1
PING 224.0.0.1 (224.0.0.1) 56(84) bytes of data.
64 bytes from 172.16.1.3: icmp_seq=1 ttl=64 time=0.163 ms
64 bytes from 172.16.1.1: icmp_seq=1 ttl=64 time=0.296 ms (DUP!)
64 bytes from 172.16.1.3: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 172.16.1.1: icmp_seq=2 ttl=64 time=0.188 ms (DUP!)
64 bytes from 172.16.1.3: icmp_seq=3 ttl=64 time=0.026 ms

--- 224.0.0.1 ping statistics ---
3 packets transmitted, 3 received, +2 duplicates, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 0.026/0.144/0.296/0.098 ms
```

1	0.000000000	172.16.1.3	224.0.0.1	ICMP	98 Echo (ping) request id=0xcf02, seq=1/25
2	0.000060432	172.16.1.1	172.16.1.3	ICMP	98 Echo (ping) reply id=0xcf02, seq=1/25
3	1.009206035	172.16.1.3	224.0.0.1	ICMP	98 Echo (ping) request id=0xcf02, seq=2/51
4	1.009283100	172.16.1.1	172.16.1.3	ICMP	98 Echo (ping) reply id=0xcf02, seq=2/51
5	2.012068235	172.16.1.3	224.0.0.1	ICMP	98 Echo (ping) request id=0xcf02, seq=3/76
6	2.012139002	172.16.1.1	172.16.1.3	ICMP	98 Echo (ping) reply id=0xcf02, seq=3/76
7	5.010528934	fe:fd:00:00:02:01	fe:fd:00:00:03:01	ARP	42 Who has 172.16.1.3? Tell 172.16.1.1
8	5.010612745	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

1. Who is now answering to the ping?
   Ara els pings els respon 172.16.1.1 (R2)
2. Can you see any reply message in SimNet3?
   Podem veure els 3 request i els 3 reply correctament.
3. Are the answer packets sent in a multicast way?
   Els paquets de resposta (Reply) no s'estan enviat amb multicast, sinó directament al host d'origen (server) des de R2. Els request tenen un TTL=1 ja que al ser un servei multicast, si el paquet no es retransmet bé a

la primera no ens interessa reenviar-lo, però els reply tenen TTL=64 perquè és el de per defecte, ja que aquest en una resposta forçada.

Multicast transmission in a subnet

We pretend to make a multicast transmission of a file between server and R2, which are located in the same subnet, but before this, let us present you some commands and files that may help you to analyze multicast.

Multicast configuration

There are some files in the system that offers to you some information about multicast interfaces and groups. Have a look to the following configuration commands and files3:

- /proc/net/igmp: this file contains the groups to which the host has joined. This file shows you information about the interfaces that are joined to certain groups (the loopback interface lo and eth1 are joined to the group 010000E0. This notation seems odd, but it is quite typical. If you translate this hex string 010000E0 to decimal you will see that is equal to 1.0.0.224. Does it sound familiar to you? This way provides another way of denoting IP addresses, just changing the order of the octets. You can also see this same information by executing the command netstat -gn.

- /proc/net/ip_mr_vif : this file contains the interfaces that are involved in multicast operations in the multicast router, and some statistics of usage (e.g. number of packets sent and received). If you see the contents of this file right now you will see the file is empty. This is totally normal, as multicast information can only be seen meanwhile there is a multicast flow being transmitted.

- /proc/net/ip_mr_vif : this file contains the interfaces that are involved in multicast operations in the multicast router, and some statistics of usage (e.g. number of packets sent and received). If you see the contents of this file right now you will see the file is empty. This is totally normal, as multicast information can only be seen meanwhile there is a multicast flow being transmitted.

- /proc/net/ip_mr_cache : this file shows the contents of the Multicast Forwarding Cache. As it happens in the previous file, if you have a look to this file right now, you will see nothing, as the file only shows active routes. Anyway, the way in which this file displays information is sometimes hard to read, so we recommend to use the command ip mroute show, which shows just the same information.

Transport protocol

As you should know, UDP is an unreliable protocol by nature, meaning that UDP packets can be lost, duplicated or delivered out of order. In case of requiring reliability to distribute critical data during the transmission of multicast packets, it should be provided by the application layer, or by using other transport protocols, for instance the Pragmatic General Multicast (PGM) protocol (RFC 3208), which have been developed to detect losses and retransmit packets.

udp-sender

You are going to use udp-sender and upd-receiver to make the transmission of a video file via multicast.

udp-sender offers several options to send information via broadcast or multicast. It can be used in a unidirectional or bidirectional way, and it is possible to tune many application and network parameters, like the TTL, the maximum bitrate, retransmissions, FEC (Forward Error Correction) codes, etc. Do not worry because in the lab session we are not going to use all these options, but it would be a good idea to have a look to them so you can understand their meaning. You must know that this software uses two multicast groups to enable the transmission:

- A multicast group to send data: in our case you will use the multicast group 232.43.211.234.
- A multicast group to control and add reliability, and you will use 225.1.2.3.

So let us start testing udp-sender:

- Open another console of server, that will be used to see the status of this host meanwhile you are sending the video. Go to the HOST machine and execute:

```
HOST$ simctl ipmulticast get server 1
```

This command will open console 1 of server (you where previously working using console 0). Now, in this console 1, see the contents of the file /proc/net/igmp:

```
server# cat /proc/net/igmp
```

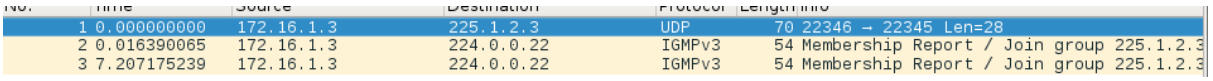
```
server:~# cat /proc/net/igmp
Idx     Device    : Count Querier       Group    Users Timer    Reporter
1       lo        :     1      V3
                             010000E0     1 0:00000000          0
5       eth1      :     1      V3
                             010000E0     1 0:00000000          0
```

- Put a wireshark listening in SimNet3.
- In this case, server will act as video server. So, go to console 0 and execute the following command:

```
server# udp-sender --file=./big_664.mpg --min-clients 1
--portbase 22345 \
--nopointopoint --interface eth1 --ttl 1 --mcast-addr
232.43.211.234 \
--mcast-all-addr 225.1.2.3
```

The previous command prepares server to transmit via multicast the locally stored video file big_664.mpg. Have a look to the manual of this command (man udp-sender) and explain each of the options used.
(--file) llegeix la informació que s'ha de retransmetre des de "file".
(--min-clients) s'inicia automàticament tan aviat com el mínim de clients sigui l'establert.
(--port-base) establir el port per a UDP multicast. Default: 9000.
(--nopointtopint) està permès més d'un receptor, i per molt que només hi hagi un sol receptor, no es permet la recepció punt a punt (unicast).
(--mcast-addr) utilitza @IPmc per enviar la informació.
(--mcast-all-add) utilitza @IPmc per fer el control de connexió. (TTL=1 per defecte)

```
server:~# udp-sender --file=./big_664.mpg --min-clients 1 --portbase 22345 \
> --nopointopoint --interface eth1 --ttl 1 --mcast-addr 232.43.211.234 \
> --mcast-all-addr 225.1.2.3
MC=x 0xffbc22a0 0xffbc22c0
Udp-sender 2004-05-31
UDP sender for ./big_664.mpg at 172.16.1.3 on eth1
Broadcasting control to 225.1.2.3
```



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.1.3	225.1.2.3	UDP	70	22346 → 22345 Len=28
2	0.016390065	172.16.1.3	224.0.0.22	IGMPv3	54	Membership Report / Join group 225.1.2.3
3	7.207175239	172.16.1.3	224.0.0.22	IGMPv3	54	Membership Report / Join group 225.1.2.3

- In SimNet3:
    1. How many packets can you see? Which type of packets?
    2. In the UDP packet, which is the destination address? and the TTL?
    3. Which are the port numbers?
    4. Can you see any IGMP packet? Which type of packet? Which is the IP destination address of this IGMP messages?

    Si ens fixem al wireshark, veiem 3 paquets:
    - Paquet **UDP**
        - Src: 172.16.1.3
        - Dst: 225.1.2.3
        - TTL: 1
        - Port number: 17
    - Paquets **IGMPv3**
        - 2 Membership report packets
        - Src: 172.16.1.3
        - Dst: 224.0.0.22

As you can see, there is an UDP packet sent towards the control multicast group, whose objective is contacting with the destination. As the client has not been launched, the server just waits to send the video. Notice that the TTL value of this IP packet is 1, as specified by the application. In this particular case, the -ttl 1 option is optional, because applications that do not specify the TTL should put 1 as default value for multicast packets (have a look to RFC 1112). However, remember to put the proper TTL when trying to reach remote destinations.
There are also a couple of IGMP messages, in which the server tries to join the control multicast group. As there is no multicast router, there is no answer to these messages.

- Now have a look to the console 1 of server, and again see the contents of the file /proc/net/igmp.

```
server:~# cat /proc/net/igmp
Idx     Device    : Count Querier       Group    Users Timer    Reporter
1       lo        :     1      V3
                              010000E0     1 0:00000000              0
5       eth1      :     2      V3
                              030201E1     1 0:00000000              0
                              010000E0     1 0:00000000              0
```

    1. Can you see any difference?
    2. What is the hex and the dot-decimal notation of the newly added multicast group?
       Veiem com a eth1 tenim un nou grup, **030201E1**, el qual passat a notació decimal és **3.2.1.225**.

- Go to console 0, and close the server by typing Control+C.

4	453.396397734	172.16.1.3	224.0.0.22	IGMPv3	54 Membership Report / Leave group 225.1.2.
5	461.200066147	172.16.1.3	224.0.0.22	IGMPv3	54 Membership Report / Leave group 225.1.2.

    1. Can you see in SimNet3 any new IGMP message? Which type?
    2. After closing, can you see again any change in the /proc/net/igmp file?
       Quan tanquem el server des de la consola 0, si tornem a visualitzar el fitxer igmp veurem que ja no estem afegits al grup (pq aquest s'ha tancat).

As you can see, when closing udp-sender the sender leaves the control group by sending two IGMP Leave Messages. No answer will be sent to these messages as there is no multicast router.

upd-receiver

Let us start testing udp-receiver:

- Open another console of R2, that will be used to see the status of this host meanwhile you are launching udp-receiver. Go to the HOST machine and execute:

```
HOST$ simctl ipmulticast get R2 1
```

This command will open console 1 of R2 (you where previously working using console 0). Now, in this console 1, see the contents of the file /proc/net/igmp:

```
server# cat /proc/net/igmp
```

```
R2:~# cat /proc/net/igmp
Idx     Device    : Count Querier       Group    Users Timer    Reporter
1       lo        :     1      V3
                              010000E0     1 0:00000000              0
5       eth2      :     1      V3
                              010000E0     1 0:00000000              0
6       eth1      :     1      V3
                              010000E0     1 0:00000000              0
```

1. Which interfaces in R2 are joined to group 224.0.0.1? Les interfícies de loopback, eth1 i eth2.lo, eth2 i eth1

- Now open two wiresharks, one listening in SimNet2 and another one in SimNet3.
- In this case, R2 will act as a host, receiving the video. So, go to console 0 and execute the following command:

```
R2# udp-receiver --file=big_664.mpg --mcast-all-addr
225.1.2.3 --ttl 1 \
--portbase 22345
```

```
R2:~# udp-receiver --file=big_664.mpg --mcast-all-addr 225.1.2.3 --ttl 1 \
> --portbase 22345
Udp-receiver 2004-05-31
UDP receiver for big_664.mpg at 198.51.100.2 on eth2
```

SimNet2:

1	0.000000000	198.51.100.2	225.1.2.3	UDP	54 22345 → 22346 Len=12
2	0.000085995	198.51.100.2	225.1.2.3	UDP	54 22345 → 22346 Len=12
3	0.021670645	198.51.100.2	224.0.0.22	IGMPv3	54 Membership Report / Join group 225.1.2.3
4	1.894935435	198.51.100.2	224.0.0.22	IGMPv3	54 Membership Report / Join group 225.1.2.3

This command prepares R2 to receive a video via multicast, which should be stored locally with name big_664.mpg.

- Have a look to the wiresharks.
    1. How many packets can you see in SimNet3? and in SimNet2?
        - **SimNet2** (xarxa de R2 a Internet): 2 paquets UDP i 2 IGMPv3
        - **SimNet3** (xarxa del servidor a R2): No veiem cap paquet
    2. Why do you think that these packets are in this network and not in the other?

According to RFC 1112, applications sending multicast packets should choose which are the output interfaces of these packets, and if not, the operating system will choose them. The application udp-receiver has an option to select the interface to send packets, but you did

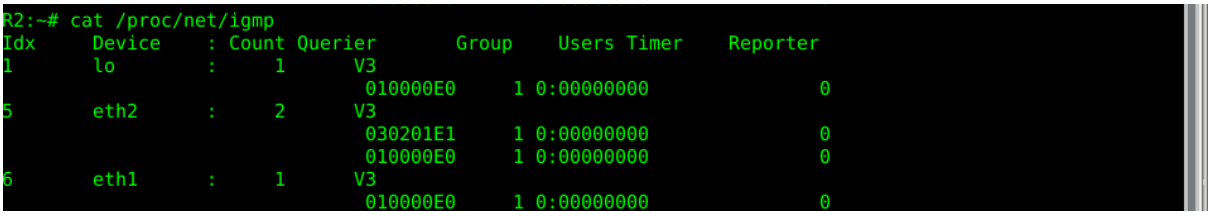
not use it, so the application has chosen the interface that corresponds to the default unicast route.

Later you will change this to make multicast packets to go through SimNet3, but now let us have a look to packets in SimNet2 using wireshark.

1. In the UDP packets, which is the destination address? and the TTL?
2. Which are the port numbers? Are these numbers familiar to you?
3. Can you see any IGMP packet?
   Els paquets UDP, l'@IP de destí és 225.1.2.3 i el TTL=1.
   El port que utilitzen és el 17
   Els paquets IGMPv3 són 2 Membership Reports (**Join Group**) que envien però no reben resposta pq no hi ha router multicast.

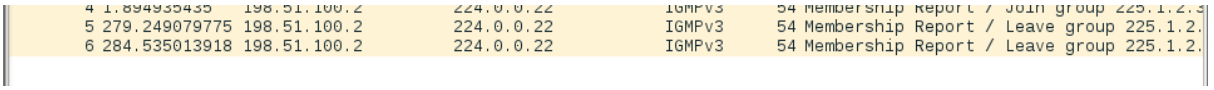
As you can see, there is a couple of UDP packets sent towards the control multicast group, whose objective is contacting with the server. As the server has not been launched, nothing happens. There are also a couple of IGMP Membership Report messages, in which R2 tries to join the control multicast group. As there is no multicast router, there is no answer to these messages.

- Now have a look to the console 1 of R2, and again see the contents of the file /proc/net/igmp.



   1. Can you see any difference?

- Go to console 0, and close udp-receiver by typing Control+C.



   1. Can you see in SimNet2 any new IGMP message? Which type?
   2. After closing, can you see again any change in the /proc/net/igmp file?

Si ara tornem a veure els grups que tenim a R2 des de la consola 1, veiem com a la interfície eth2 tenim el nou grup al que estem afegits, i si sortim del udp-receiver a la consola 0, veurem com aquest grup desapareix, i així mateix com a la SimNet2 arriben **2 paquets IGMPv3** de Membership Report/**Leave Group** (els quals tampoc obtindran resposta).

As you can see, when closing udp-receiver R2 leaves the control group by sending two IGMP Leave Messages. No answer will be sent to these messages as there is no multicast router.

<u>Sending the video file in the subnet</u>

Now that you know how udp-sender and udp-receiver behave, it is time to make a real distribution of a video file via multicast:

- Put a wireshark listening in SimNet3.

- Go to console 0 of server and execute again the following command:

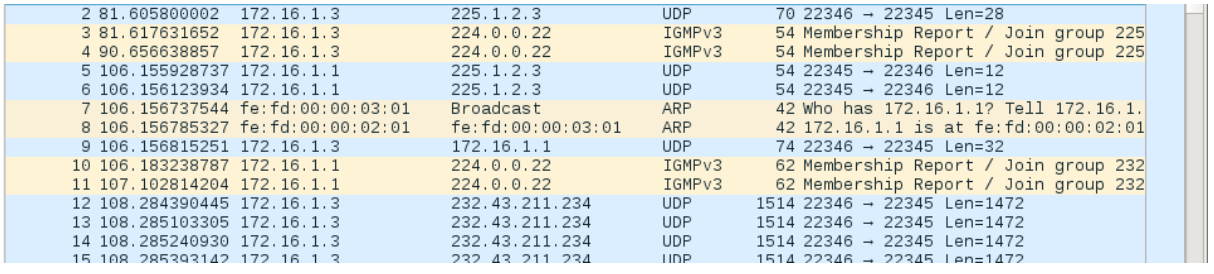
```
server# udp-sender --file=./big_664.mpg --min-clients 1
--portbase 22345 \
--nopointopoint --interface eth1 --ttl 1 --mcast-addr
232.43.211.234 \
--mcast-all-addr 225.1.2.3
```

- Go to console 0 of R2 and execute the following command, now forcing the packets to be sent to the proper interface:

```
R2# udp-receiver --file=big_664.mpg --mcast-all-addr
225.1.2.3 --ttl 1 \
--interface eth1 --portbase 22345
```

```
server:~# udp-sender --file=./big_664.mpg --min-clients 1 --portbase 22345 \
> --nopointopoint --interface eth1 --ttl 1 --mcast-addr 232.43.211.234 \
> --mcast-all-addr 225.1.2.3
MC=x 0xffb22910 0xffb22930
Udp-sender 2004-05-31
UDP sender for ./big_664.mpg at 172.16.1.3 on eth1
Broadcasting control to 225.1.2.3
New connection from 172.16.1.1  (#0) 00000019
Ready. Press any key to start sending data.
Starting transfer: 00000019
bytes=        425 152 re-xmits=000034 ( 11.6%) slice=0159       6 385 664 -   0
Timeout notAnswered=[0] notReady=[0] nrAns=0 nrRead=0 nrPart=1 avg=9148
bytes=     17 408 000 re-xmits=000124 (  1.0%) slice=0073      17 408 000 -   0
Transfer complete.
Disconnecting #0 (172.16.1.1)
```

```
R2:~# udp-receiver --file=big_664.mpg --mcast-all-addr 225.1.2.3 --ttl 1 \
> --interface eth1 --portbase 22345
Udp-receiver 2004-05-31
UDP receiver for big_664.mpg at 172.16.1.1 on eth1
received message, cap=00000019
Connected as #0 to 172.16.1.3
Listening to multicast on 232.43.211.234
Press any key to start receiving data!
bytes=     17 408 000  ( 62.60 Mbps)     17 408 000
Transfer complete.
```



2	81.605800002	172.16.1.3	225.1.2.3	UDP	70 22346 → 22345 Len=28
3	81.617631652	172.16.1.3	224.0.0.22	IGMPv3	54 Membership Report / Join group 225
4	90.656638857	172.16.1.3	224.0.0.22	IGMPv3	54 Membership Report / Join group 225
5	106.155928737	172.16.1.1	225.1.2.3	UDP	54 22345 → 22346 Len=12
6	106.156123934	172.16.1.1	225.1.2.3	UDP	54 22345 → 22346 Len=12
7	106.156737544	fe:fd:00:00:03:01	Broadcast	ARP	42 Who has 172.16.1.1? Tell 172.16.1.
8	106.156785327	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
9	106.156815251	172.16.1.3	172.16.1.1	UDP	74 22346 → 22345 Len=32
10	106.183238787	172.16.1.1	224.0.0.22	IGMPv3	62 Membership Report / Join group 232
11	107.102814204	172.16.1.1	224.0.0.22	IGMPv3	62 Membership Report / Join group 232
12	108.284390445	172.16.1.3	232.43.211.234	UDP	1514 22346 → 22345 Len=1472
13	108.285103305	172.16.1.3	232.43.211.234	UDP	1514 22346 → 22345 Len=1472
14	108.285240930	172.16.1.3	232.43.211.234	UDP	1514 22346 → 22345 Len=1472
15	108.285393142	172.16.1.3	232.43.211.234	UDP	1514 22346 → 22345 Len=1472

12445	109.998669688	172.16.1.3	232.43.211.234	UDP	1514 22346 → 22345 Len=1472
12446	109.998697700	172.16.1.3	232.43.211.234	UDP	1514 22346 → 22345 Len=1472
12447	109.998760641	172.16.1.3	232.43.211.234	UDP	186 22346 → 22345 Len=144
12448	109.999808639	172.16.1.1	172.16.1.3	UDP	50 22345 → 22346 Len=8
12449	110.000523375	172.16.1.3	232.43.211.234	UDP	1514 22346 → 22345 Len=1472
12450	110.000552959	172.16.1.3	232.43.211.234	UDP	186 22346 → 22345 Len=144
12451	110.001463428	172.16.1.1	172.16.1.3	UDP	50 22345 → 22346 Len=8
12452	110.003972721	172.16.1.3	232.43.211.234	UDP	186 22346 → 22345 Len=144
12453	110.004108590	172.16.1.1	172.16.1.3	UDP	50 22345 → 22346 Len=8
12454	110.039870712	172.16.1.3	224.0.0.22	IGMPv3	54 Membership Report / Leave group 22
12455	110.510610766	172.16.1.1	172.16.1.3	UDP	46 22345 → 22346 Len=4
12456	110.510679826	172.16.1.3	172.16.1.1	ICMP	74 Destination unreachable (Port unre
12457	110.532134273	172.16.1.1	224.0.0.22	IGMPv3	62 Membership Report / Leave group 22
12458	113.302367781	fe:fd:00:00:02:01	fe:fd:00:00:03:01	ARP	42 Who has 172.16.1.3? Tell 172.16.1.
12459	113.302426299	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
12460	114.558624142	172.16.1.1	224.0.0.22	IGMPv3	62 Membership Report / Leave group 22
12461	115.616246229	172.16.1.3	224.0.0.22	IGMPv3	54 Membership Report / Leave group 22

1. Was the transmission ok?
2. How many packets can you see in wireshark?

As you have seen, the transmission went ok, but now it is time to focus on some key point in this transmission.

- Let us start analyzing the four initial UDP packets of short-length.
    1. What do you think is the purpose of these initial packets?
    2. In the first UDP packet sent by server, which is the IP destination address?
    3. Do you recognize anything inside of the DATA field of this packet? Try to convert 232.43.211.234 in hex.
    4. At the time this first packet was sent by the server, was R2 listening at the control multicast address?
    5. What do you think is the purpose of the second and third UDP packets (the ones sent by R2)?
    6. At this time, does R2 know which is the data multicast group that server is going to use to send the video?
    7. Have a look to the fourth UDP packet. Is it a multicast or unicast packet? What is the purpose of this message? Do you recognize anything in the data field of the packet?

  Al principi de tot tenim 4 paquets petits UDP, els quals serveixen perquè l'emissor i el receptor es coneguin mútuament i puguin intercanviar informació clau per la transmissió.
  Ara l'emissor sap quin és el grup d'enviament multicast, però no sap qui és el receptor concret. L'últim dels 4 paquets li indica al receptor quin és el grup al que s'ha d'unir. Veiem com és un missatge Unicast enviat directament al receptor.

  Als següents paquets es transmet la DATA del paquet. Si convertim 232.43.211.234 a hex obtenim: E8:2B:D3:EA <-- Els últims 24 bits de l'@IP, són els que mapegem directament a la @MAC de destí multicast: 01:00:5E:2B:D3:EA.
  Els dos paquets que s'envien després del Membership Report suposo que són per veure si tenim establerta la comunicació.
  El quart paquet enviat pel server, el qual és UNICAST, deu ser per indicar que s'ha establert la comunicació, i ja es pot unir el host interessat al grup indicat per la recepció de paquets.
  Llavors veiem finalment com s'envia un últim Membership Report / Join Group definitiu.

As you can see, this first dialogue of four short-length UDP packets is used for sender and receiver to mutually recognize themselves and exchange key information about the transmission. The sender knows which is the data multicast group, but not the receiver. Until the last fourth packet the receiver is not informed (in a unicast way) about the data multicast group that has to join. For this reason, the following message is an IGMP Membership Report of R2 joining both groups. You can verify that R2 is joined to both groups by having a look to file /proc/net/igmp meanwhile the file is downloading.

- Regarding the transmission of the video (look at packets that are full of information):
    1. Which is the destination IP address of these packets?
    2. What is the size of most of this data packets?
    3. There are some unicast packets from R2 to server. What do you think is the purpose of these feedback packets?

L'@IP de destí d'aquests paquets és la 232.43.211.234, la qual pertany a una adreça multicast, del host en concret proporcionat pels 24 bits menys significatius.
El tamany de la majoria d'aquests paquets és de 1514 bits.
De tant en tant tenim missatges UNICAST des de R2 al server. Aquests deuen ser missatges generals de MR, per continuar indicant que el host està interessat en aquell grup multicast.

Finalment tant l'emissor com el receptor acaben la transmissió i abandonen els corresponents grups enviant un **IGMP Leave Group**.

As you have seen, server sends the video by means of several UDP packets (full of data) with destination address 232.43.211.234. There are also some feedback unicast packets from R2 to server, used for reliability purposes. Finally, both sender and receiver leave the corresponding multicast groups by sending IGMP Leave Messages.

Now you know how to transmit information via multicast in a subnet. However, you know that most remote locations are far away, so routers should be used to reach them. It is time you go a step further, trying to do the same thing, but now involving multicast routers during the transmission.

Multicast and routing
In previous sections you only used the virtual machines in SimNet3, but now you will use the rest of the scenario.
Remember that in the scenario of Fig 1 you have three multicast islands (SimNet0, SimNet3 and SimNet5) connected through a public network. There are some multicast routers (R1, R2 and R3), and a router without multicast capacities (RC). We mentioned that multicast islands were connected via GRE tunnels, but in fact you are going to create them right now. There should be a GRE tunnel between R1 and R2 called tunnel0, and another one between R3 and R2 called tunnel1.

Configuring GRE tunnels
In a previous lab session you learned how to configure and test IPIP tunnels. Unfortunately, Linux does not directly support multicast transmission via IPIP tunnels, and for this reason we are going to use GRE tunnels. If you want to know more about GRE tunnels, you can read RFC 1701 and RFC 1702.
You are not going to configure these tunnels manually, but using a script. Anyway, here you have a summary of the command that should be executed in the encapsulator and decapsulator to do so (you DO NOT have to execute them):

- Configure the tunnel interface by means of the command ip tunnel. Tunnels have to work in gre mode. As an example, in R2 the command to execute:

```
R2# ip tunnel add tunnel0 mode gre local 198.51.100.2 remote 192.0.2.2 dev eth2
```

- Activate the new virtual interface of type tunnel. To do so, it is necessary to assign an IP address to it (for instance, by means of the ifconfig command). In Linux, it is necessary to assign an IP address to any interface if we want it to properly work. Just as an example:

```
R2# ifconfig tunnel0 192.168.110.1
```

- Update the routing table to make the proper packets to be routed towards the tunnel (for instance, by means of the route command). Just as an example:

```
R2# route add -net 192.168.0.0/24 dev tunnel0
```

To create both tunnels automatically, go to the console of your HOST machine, and execute:

```
HOST$ simctl ipmulticast exec addtun
```

After creating the tunnels, you have to verify that they properly work.

- Put a wireshark listening to interface SimNet2.
- Send a ping from the server to host2.

```
server:~# ping -c1 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=62 time=0.817 ms

--- 192.168.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.817/0.817/0.817/0.000 ms
```

1	0.000000000	fe:fd:00:00:02:02	Broadcast	ARP	42	Who has 198.51.100.1? Tell 198.51.100.2
2	0.000057302	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
3	0.000094868	172.16.1.3	192.168.0.2	ICMP	122	Echo (ping) request id=0xeb02, seq=1/25
4	0.000554165	192.168.0.2	172.16.1.3	ICMP	122	Echo (ping) reply id=0xeb02, seq=1/25
5	5.017108644	fe:fd:00:00:01:02	fe:fd:00:00:02:02	ARP	42	Who has 198.51.100.2? Tell 198.51.100.1
6	5.017362866	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

1. Is the ping working?
2. Can you see packets in the wireshark? What type of packets? Describe the headers you see.
3. Regarding the IP outer header, which are the source and destination addresses?
4. Which is the size of the GRE header?
5. Which is the source and destination addresses in the inner header?

Veiem com el ping funciona correctament, s'envia bé el request i el reply.
Si ens fixem en **Src** i **Dst** del **paquet IP exterior**, veiem com aquest s'envia directament des de R2 fins a R1, i de tornada igual, sense passar per internet.
Al **paquet IP interior** veiem com les adreces de destí i origen són les del host2 i server respectivament.

- Send also a ping from the server to host4 to verify that tunnel1 works.

```
server:~# ping -c1 192.168.5.4
PING 192.168.5.4 (192.168.5.4) 56(84) bytes of data.
64 bytes from 192.168.5.4: icmp_seq=1 ttl=62 time=0.713 ms

--- 192.168.5.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.713/0.713/0.713/0.000 ms
```

7	228.583719522	172.16.1.3	192.168.5.4	ICMP	122	Echo (ping) request id=0xec02, seq=1/25
8	228.584168856	192.168.5.4	172.16.1.3	ICMP	122	Echo (ping) reply id=0xec02, seq=1/25
9	233.585841632	fe:fd:00:00:01:02	fe:fd:00:00:02:02	ARP	42	Who has 198.51.100.2? Tell 198.51.100.1
10	233.586023371	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

Si enviem un ping de server a host4, comprovem com també funciona el tunel ja que el paquet s'envia directament entre routers.

Multicast and ICMP

These error messages are in fact the most typical ones, so this means that ICMP is not going to help you during the sending process of multicast packets, sending errors in case you commit a mistake. You are not going to receive any message like “Host Unreachable”,

“Fragmentation Needed”, or “Time Exceeded” due to an error generated by a multicast packet. Next, you are going to deal with multicast and IP tunneling, so the scenario is prone to errors generated by TTL and MTU problems (as you noticed in the IP tunneling lab session). So, take this into account when configuring the network and when sending multicast packets. You should pay special attention when defining the TTL in each multicast packet (the scope of the packet), and also the size of them so they can traverse tunnels properly.

Multicast static routing: smcroute

So you are ready to create the multicast routing tree from the sender to all the receivers. You are not going to use dynamical multicast routing, but static one by means of the smcroute command. This smcroute command is able to manipulate the multicast static routes of the Linux kernel. We recommend to you to consult the manual of smcroute to become familiar with it.

You will start configuring the multicast static routing tree to communicate the two multicast islands SimNet0 and SimNet3, which remember that are connected via the gre tunnel tunnel0. Later you will connect the other island of SimNet5. Notice that this configuration pretends to be similar to the one created some years ago in the MBONE.

- Have a look to the file /proc/net/ip_mr_vif in router R2. This file should be empty because the router has not yet taken any action.
- Open four wiresharks, listening from SimNet0 to SimNet3.
- Start the daemon smcroute in router R2.

```
R2# smcroute -d
```

See again the contents of /proc/net/ip_mr_vif. What changes do you see?

```
R2:~# cat /proc/net/ip_mr_vif
Interface      BytesIn  PktsIn  BytesOut PktsOut Flags Local    Remote
 0 eth2             0       0         0       0 00000 026433C6 00000000
 1 eth1             0       0         0       0 00000 010110AC 00000000
 2 tunnel0          0       0         0       0 00000 016EA8C0 00000000
 3 tunnel1          0       0         0       0 00000 026EA8C0 00000000
```

This file /proc/net/ip_mr_vif will show you which are the interfaces which are able to use multicast in the multicast router, and some statistics of their use. As you have not sent any multicast packet, you will see that the statistics are empty.

- Execute the following command in R2 to add a new multicast static route:

```
R2# smcroute -a eth1 0.0.0.0 232.43.211.234 tunnel0
```

This command will route IP datagrams entering through eth1 with any origin address and destination address the multicast group 232.43.211.234 towards the tunnel (so they will reach the other island).

- Execute the following command to make R2 to join the entry interface eth1 to the multicast group 232.43.211.234:

```
R2# smcroute -j eth1 232.43.211.234
```

1. View the content of the file /proc/net/igmp. What changes do you see in this file?

```
Idx     Device    : Count Querier       Group    Users Timer    Reporter
1       lo        :     1      V3
                                010000E0     1 0:00000000              0
5       eth2      :     1      V3
                                010000E0     1 0:00000000              0
6       eth1      :     2      V3
                                EAD32BE8     1 0:00000000              0
                                010000E0     1 0:00000000              0
7       tunnel0   :     1      V3
                                010000E0     1 0:00000000              0
8       tunnel1   :     1      V3
                                010000E0     1 0:00000000              0
```

Veiem com la interfície **eth1** s'ha afegit a un grup (EAD32BE8 --> El qual passat a binari és 234.211.43.232)

2. Have a look to SimNet3. Can you see any IGMP message in that interface?

```
     1 0.000000000   172.16.1.1        224.0.0.22        IGMPv3      54 Membership Report / Join group 232.43.21
     2 7.228697706   172.16.1.1        224.0.0.22        IGMPv3      54 Membership Report / Join group 232.43.21
```

Si mirem SimNet3, veiem com han aparegut un parell de **missatges IGMP** Membership Reports / Join the group 232.43.211.234

- Now send a ping from server to the multicast group 232.43.211.234 with a scope of three hops.

```
server# ping -t3 -c1 232.43.211.234
```

```
server:~# ping -t3 -c1 232.43.211.234
PING 232.43.211.234 (232.43.211.234) 56(84) bytes of data.
64 bytes from 172.16.1.1: icmp_seq=1 ttl=64 time=0.252 ms

--- 232.43.211.234 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.252/0.252/0.252/0.000 ms
```

SImNet3:

```
     1 0.000000000   172.16.1.3        232.43.211.234    ICMP        98 Echo (ping) request  id=0xf202, seq=1/25
     2 0.000091172   fe:fd:00:00:02:01 Broadcast         ARP         42 Who has 172.16.1.3? Tell 172.16.1.1
     3 0.000130740   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
     4 0.000182585   172.16.1.1        172.16.1.3        ICMP        98 Echo (ping) reply    id=0xf202, seq=1/25
```

SImNet2:

```
     1 0.000000000   fe:fd:00:00:02:02 Broadcast         ARP         42 Who has 198.51.100.1? Tell 198.51.100.2
     2 0.000059593   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
     3 0.000118295   172.16.1.3        232.43.211.234    ICMP       122 Echo (ping) request  id=0xf202, seq=1/25
```

1. Is the ping working? In which SimNets can you see ICMP packets?
   Veiem com el ping funciona. Aquest es veu tant a SimNet2 com a SimNet3, *però hi ha una diferència*: A SimNet3 veiem tant el echo request com el echo reply, pero a SimNet2 només veiem el echo request i cap reply. Això es deu a que el grup multicast, no retorna missatges de confirmació del ping, però en canvi el router si que el retorna al host.

2. Can you see any encapsulation in the ICMP packet in SimNet1 and SimNet2? What is the size of a GRE header?
   Només veiem encapsulació IP a SimNet2 --> pq a SimNet3 ho envia amb enrutament unicast, mentres que a SimNet2 al enviar el paquet del R2 al grup, aquest anirà encapsulat.

3. Have a look again to file /proc/net/ip_mr_vif. Do you see any change in the statistics related to tunnel0?

```
R2:~# cat /proc/net/ip_mr_vif
Interface      BytesIn  PktsIn  BytesOut PktsOut Flags Local    Remote
 0 eth2              0       0         0       0 00000 026433C6 00000000
 1 eth1             84       1         0       0 00000 010110AC 00000000
 2 tunnel0           0       0        84       1 00000 016EA8C0 00000000
 3 tunnel1           0       0         0       0 00000 026EA8C0 00000000
```

Si ara mirem el fitxer /proc/net/ip_mr_vif veiem com a la interfície eth1 se li han afegit dades a les seves estadístiques.

4. Execute ip mroute show to see the Multicast Forwarding Cache. What is the meaning of all these parameters?

```
R2:~# ip mroute show
(172.16.1.3, 232.43.211.234)     Iif: eth1       Oifs: tunnel0
```

Si executem ip mroute show veurem com tenim una ruta creada, amb @origen, @destí, Interfície d'entrada i interfícies de sortida.

5. Why do you think R1 is not forwarding the packet?

A SimNet1 no veiem res pq no tenim R1 configurat per fer forwarding de Paquets IP amb destí al grup 232.43.211.234.

According to wireshark, you will see that the packet has been introduced in the tunnel, but the decapsulator R1 has not been configured as multicast router, so the ICMP packet is not forwarded to SimNet0. Notice also that you have configured in R2 a unidirectional static multicast route, that is, from the server in SimNet3 to possible receivers in SimNet0.

- Using smcroute, configure R1 to route IP datagrams entering through the tunnel with any origin address and destination address the multicast group 232.43.211.234 towards eth1. To do so, just follow the same procedure that you used in R2.

```
R1#: smcroute -d
R1#: smcroute -a tunnel0 0.0.0.0 232.43.211.234 eth1
R1#: smcroute -j tunnel0 232.43.211.234
```

1. Can you see the packets in SimNet0? Which type of packets?
2. Can you see any ICMP Echo Reply? Why?
3. Which is the TTL value of this packet in SimNet0?
4. If you send the same ping from server but with TTL=2, would it work?

```
server#: ping -t3 -c1 232.43.211.234
```

SimNet0:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.1.3	232.43.211.234	ICMP	98	Echo (ping) request id=0xf302, seq=1/25

If you made the previous configuration properly, when you ping the multicast group from server with TTL=3, you will see the packet to reach SimNet0. IThere will be no ICMP Echo Reply message, as there is not an application running there ready to answer. Notice that it is important to define the proper scope for multicast packets. With TTL=2 you will not reach the destination. With TTL=4 or higher you will reach destination, but maybe you will also send packets to other networks that you did not pretend to reach, wasting resources (bandwidth, CPU, etc.). So be careful when defining the scope of your multicast packets.

Testing tools

ssmping

You are going to use the application ssmping, which allows you to verify both unicast and multicast routing. Have a look to the on-line manual of this program to become familiar with it. Basically, ssmping allows to verify if a host can receive SSM (Source Specific Multicast) packets from another

host, by sending unicast and multicast UDP packets (remember that TCP does not support multicast traffic).

- Put wiresharks listening in all the involved networks.
- In console 0 of server start the daemon of the application executing the command

```
server# ssmpingd
```

When you consider necessary, use console 1 to control if server has joined to any new multicast group (use netstat -gn).

- In console 0 host2, execute the command:

```
host2# ssmping 172.16.1.3
```

```
eceived request from 192.168.0.2
eceived request from 192.168.0.2
eceived request from 192.168.0.2
eceived request from 192.168.0.2
eceived request from 192.168.0.2
eceived request from 192.168.0.2
eceived request from 192.168.0.2
```

```
smping joined (S,G) = (172.16.1.3,232.43.211.234)
inging S from 192.168.0.2
  unicast from 172.16.1.3, seq=1 dist=2 time=4.465 ms
  unicast from 172.16.1.3, seq=2 dist=2 time=1.091 ms
  unicast from 172.16.1.3, seq=3 dist=2 time=1.871 ms
  unicast from 172.16.1.3, seq=4 dist=2 time=0.606 ms
  unicast from 172.16.1.3, seq=5 dist=2 time=1.145 ms
```

When you consider necessary, use console 1 of host2 to control the same thing.

1. What output can you see in host2? and in server?
2. According to this output, do you consider that the multicast routing is working?

A la consola del server veiem com aquest rep les requests dels pings mcast enviats des de 192.168.0.2 (host2), envia seguidament un altre missatge unicast de tornada cap a host2, i finalment envia també el missatge mcast amb destí el grup (232.43.211.234).

Notice that ssmping is able to test both unicast and multicast routing. Have a look to the UDP packets. Notice that the behavior is periodical, and groups of three UDP packets are sent every second.

1. What is the destination and origin address of the first packet? Is a unicast or multicast packet?
2. And the second one? What is the purpose of these two packets?
3. Is the third packet different? Who is the origin and destination? Unicast or multicast? So, what is this packet testing? Unidirectional or bidirectional?

Regarding IGMP packets:

1. In SimNet0, can you see any IGMP Message? What kind of message? Which is the multicast group involved? (for easiness, put igmp in the Filter option available in the top-left corner of wireshark).
2. Is this information coherent with what you see using netstat -gn?
3. Is there any other IGMP message in the rest of networks?

- Now that you know what is the purpose of ssmping, and how it works, just test the same in the inverse direction (execute ssmping in server and ssmpingd in host2). Prior to execute the commands, try to foresee what is going to happen. Analyze the wireshark traces and extract your own conclusions.

mcsender and emcast

It is also possible to verify the correct use of multicast static routing using the applications mcsender and emcast.

Have a look to the manual of both applications to see how they work.

- Put wiresharks listening in all the involved networks.
- In server (acting as issuer) execute mcsender to send packets to the multicast group 232.43.211.234 using the UDP port 12345 with a scope of three hops.
- In host2 (acting as receiver) execute emcast to join group 232.43.211.234 using UDP port 12345.

```
server#: mcsender -t3 232.43.211.234:12345 // emisor
host2#: emcast 232.43.211.234:12345 // receptor
```

1. What output can you see in host2? and in server?
2. According to this output, do you consider that the multicast routing is working?
3. What is the destination and origin address of the packets? Are they unicast or multicast?
4. Can you see any IGMP Message? Where? What kind of message? Which is the multicast group involved?

A host2 podem veure: this is the test message from mclab/mcsender
A server podem veure: res
Si considerem aquestes sortides podem concloure que el routing multicast funciona.
(172.16.1.3→ 232.43.211.234) multicast
Podem veure a SN0 dos IGMP Membership Report / Join group 232.43.211.234 i dos IGMP Membership Report / Leave group (192.168.0.2→ 224.0.0.22)

Trees with multiple branches

So now you know how to create the multicast static routing tree to communicate the two multicast islands SimNet0 and SimNet3. But, what happens with the other island SimNet5? What should you do to create a tree with multiple branches? Your objective now is to create the multicast static routing tree necessary to perform a multicast transmission from server to receivers located at both SimNet0 and SimNet5.
The command smcroute allows us to create this configuration in an easy way, by only changing the configuration in R2.

- Put wiresharks listening in all the involved networks.
- Kill the smcroute daemon, so all previous configuration in that router will be lost

```
R2# smcroute -k
```

- Start again the daemon smcroute.

```
R2# smcroute -d
```

- Here you will see the main difference regarding the previous configuration. When adding the new multicast static route, instead of an only output interface, you can put a list of interfaces that will be used as output. So, execute the following command:

```
R2# smcroute -a eth1 0.0.0.0 232.43.211.234 tunnel0 tunnel1
```

- Join the entry interface eth1 to the multicast group 232.43.211.234:

```
R2# smcroute -j eth1 232.43.211.234
```

- Use ssmping to test the configuration, executing these commands:

```
server# ssmpingd
host2# ssmping 172.16.1.3
host4# ssmping 172.16.1.3
```

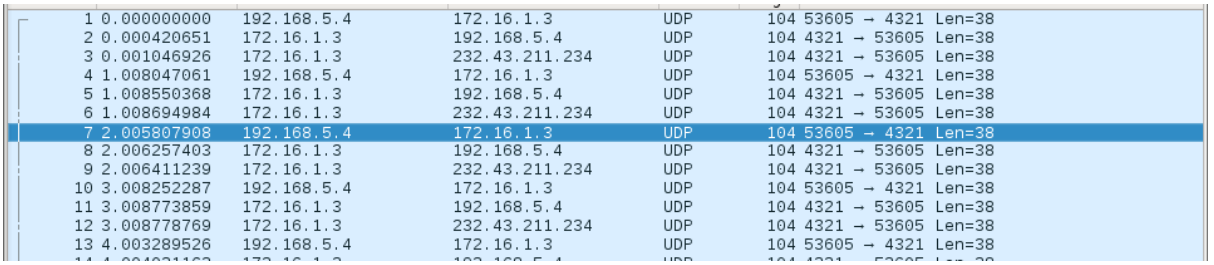
```
server:~# ssmpingd
received request from 192.168.0.2
received request from 192.168.0.2
received request from 192.168.0.2
received request from 192.168.0.2
received request from 192.168.0.2
received request from 192.168.0.2
```

```
host2:~# ssmping 172.16.1.3
ssmping joined (S,G) = (172.16.1.3,232.43.211.234)
pinging S from 192.168.0.2
  unicast from 172.16.1.3, seq=1 dist=2 time=7.094 ms
multicast from 172.16.1.3, seq=1 dist=2 time=8.053 ms
  unicast from 172.16.1.3, seq=2 dist=2 time=2.739 ms
multicast from 172.16.1.3, seq=2 dist=2 time=2.744 ms
  unicast from 172.16.1.3, seq=3 dist=2 time=1.941 ms
multicast from 172.16.1.3, seq=3 dist=2 time=3.166 ms
  unicast from 172.16.1.3, seq=4 dist=2 time=0.840 ms
multicast from 172.16.1.3, seq=4 dist=2 time=1.179 ms
  unicast from 172.16.1.3, seq=5 dist=2 time=2.411 ms
multicast from 172.16.1.3, seq=5 dist=2 time=3.515 ms
  unicast from 172.16.1.3, seq=6 dist=2 time=1.390 ms
multicast from 172.16.1.3, seq=6 dist=2 time=1.393 ms
^C
--- 172.16.1.3 ssmping statistics ---
6 packets transmitted, time 5465 ms
unicast:
   6 packets received, 0% packet loss
   rtt min/avg/max/std-dev = 0.840/2.735/7.094/2.047 ms
multicast:
   6 packets received, 0% packet loss since first mc packet (seq 1) recvd
   rtt min/avg/max/std-dev = 1.179/3.341/8.053/2.277 ms
```

```
host4:~# ssmping 172.16.1.3
ssmping joined (S,G) = (172.16.1.3,232.43.211.234)
pinging S from 192.168.5.4
  unicast from 172.16.1.3, seq=1 dist=2 time=0.685 ms
  unicast from 172.16.1.3, seq=2 dist=2 time=1.008 ms
  unicast from 172.16.1.3, seq=3 dist=2 time=0.977 ms
  unicast from 172.16.1.3, seq=4 dist=2 time=0.863 ms
  unicast from 172.16.1.3, seq=5 dist=2 time=0.986 ms
^C
--- 172.16.1.3 ssmping statistics ---
5 packets transmitted, time 4097 ms
unicast:
   5 packets received, 0% packet loss
   rtt min/avg/max/std-dev = 0.685/0.903/1.008/0.126 ms
multicast:
   0 packets received, 100% packet loss
```

1. Is ssmping working? What output message can you see in host2? host4? And in server?
2. What will deduce from this output messages? What do you consider is missing?
3. Have a look to SimNet4, what type of messages can you see? unicast or multicast? And in SimNet5?

SImNet4:



1	0.000000000	192.168.5.4	172.16.1.3	UDP	104 53605 → 4321 Len=38
2	0.000420651	172.16.1.3	192.168.5.4	UDP	104 4321 → 53605 Len=38
3	0.001046926	172.16.1.3	232.43.211.234	UDP	104 4321 → 53605 Len=38
4	1.008047061	192.168.5.4	172.16.1.3	UDP	104 53605 → 4321 Len=38
5	1.008550368	172.16.1.3	192.168.5.4	UDP	104 4321 → 53605 Len=38
6	1.008694984	172.16.1.3	232.43.211.234	UDP	104 4321 → 53605 Len=38
7	2.005807908	192.168.5.4	172.16.1.3	UDP	104 53605 → 4321 Len=38
8	2.006257403	172.16.1.3	192.168.5.4	UDP	104 4321 → 53605 Len=38
9	2.006411239	172.16.1.3	232.43.211.234	UDP	104 4321 → 53605 Len=38
10	3.008252287	192.168.5.4	172.16.1.3	UDP	104 53605 → 4321 Len=38
11	3.008773859	172.16.1.3	192.168.5.4	UDP	104 4321 → 53605 Len=38
12	3.008778769	172.16.1.3	232.43.211.234	UDP	104 4321 → 53605 Len=38
13	4.003289526	192.168.5.4	172.16.1.3	UDP	104 53605 → 4321 Len=38

La comunicació amb host2 funciona, però la de **host4** no --> R1 està ben configurat per fer forwarding de paquets multicast però **R3** no, li **hem d'afegir la ruta** per fer **forwarding de paquets amb destí el grup** 232.43.211.234.

- Make the necessary changes to make the multicast routing work so you can send a multicast flow from SimNet0 to both SimNet3 and SimNet5.
- Test that the configuration is properly working, with ssmping.
- Use also mcsender and emcast to test the configuration.

```
R3:~# smcroute -d
R3:~# smcroute -a tunnel1 0.0.0.0 232.43.211.234 eth1
R3:~# smcroute -j tunnel1 232.43.211.234
server:# ssmpingd
host4:# ssmping 172.16.1.3
server#: mcsender -t3 232.43.211.234:12345
host4#: emcast 232.43.211.234:12345
```

Si ara tornem a enviar els ssmping des de host4, veiem com ja ens arriben.

Sending the video

Now you are ready to make a real distribution of a video via multicast, but now between remote multicast islands. For the video distribution, you will use udp-sender and udp-receiver again. Remember that this software uses two multicast groups to enable the transmission: a multicast group to send data and a multicast group to control and add reliability to the transmission. In our case, you will use the multicast group 232.43.211.234 (previously configured in our scenario) to send data. As control group you will use 225.1.2.3.

- Configure the multicast routers for the control group 225.1.2.3, exactly in the same way you did previously with the data group 232.43.211.234.

```
R2:~# smcroute -a eth1 0.0.0.0 225.1.2.3 tunnel0 tunnel1
R2:~# smcroute -j eth1 225.1.2.3
R1:~# smcroute -a tunnel0 0.0.0.0 225.1.2.3 eth1
R1:~# smcroute -j tunnel0 225.1.2.3
R3:~# smcroute -a tunnel1 0.0.0.0 225.1.2.3 eth1
R3:~# smcroute -j tunnel1 225.1.2.3
```

- Test that this control group 225.1.2.3 is properly transmitted by the multicast tree from SimNet0 to both SimNet3 and SimNet5.
- Start udp-receiver in both host2 and host4:

```
host2# udp-receiver --file=big_664_in_host2.mpg
--mcast-all-addr 225.1.2.3 --ttl 3 --interface eth1
--portbase 22345
host4# udp-receiver --file=big_664_in_host4.mpg
--mcast-all-addr 225.1.2.3 --ttl 3 --interface eth1
--portbase 22345
```

- Start upd-sender to transmit the video big_664.mpg from server. Remember that your multicast routing tree is unidirectional, and hence you will not receive any feedback from the receivers (for that reason we use the options autostart and max-bitrate. Notice also that you need to put the proper ttl and blocksize values to properly traverse the tunnels. Execute:

```
server# udp-sender --file=./big_664.mpg --portbase 22345
--nopointopoint --interface eth1 --ttl 3 --mcast-addr
232.43.211.234 --mcast-all-addr 225.1.2.3 --nokbd --async
--max-bitrate 900K --autostart 1 --blocksize 1400
```

server:

```
Udp-sender 2004-05-31
UDP sender for ./big_664.mpg at 172.16.1.3 on eth1
Broadcasting control to 225.1.2.3
Starting transfer: 00000039
Bytes=     14 336 000 re-xmits=000000 (  0.0%) slice=1024      17 408 000 -   0
```

host2:

```
Udp-receiver 2004-05-31
UDP receiver for big_664_in_host2.mpg at 192.168.0.2 on eth1
Connected as #0 to 172.16.1.3
Listening to multicast on 232.43.211.234
Bytes=      4 300 800  (  0.86 Mbps)      2 867 200
```

If the network is properly configured, you will see that the file is being transmitted via multicast and it is captured by both receivers. Once finished (after 2-3 minutes), a copy of the file should be stored in both host2 and host4. If you want to verify that these copies are equal to the one stored in server, you can use md5sum, which computes a MD5 hash of the file. The result of this md5sum must be the same in all the hosts.

```
server~# md5sum big_664.mpg
cf88e5d297fc7458e3914c224af4c06f  big_664.mpg
```

Remember that the value may be different as we did not implement any reliability mechanism (there is no feedback to ask for retransmissions, and you do not use any FEC technique to correct errors).

Bidirectional tree

Creating a bidirectional tree is mandatory to offer reliability to the multicast transmission. As we mentioned, we are not going to deal with reliable multicast, but you should be able to create this bidirectional tree using smcroute.

- Configure the multicast routers to create this bidirectional tree for both data and control groups 232.43.211.234 and 225.1.2.3. Notice that in the previous section you created the forward tree (from server to host2 and host4).

Now you only need to create the backward tree (from host2 and host4 to the server).

- Start udp-receiver in both host2 and host4:
  host2# udp-receiver --file=big_664_in_host2.mpg --mcast-all-addr 225.1.2.3 \ --ttl 3 --interface eth1 --portbase 22345
  host4# udp-receiver --file=big_664_in_host4.mpg --mcast-all-addr 225.1.2.3 \ --ttl 3 --interface eth1 --portbase 22345
- Start upd-sender to transmit the video big_664.mpg from server. Now the multicast routing tree is bidirectional, so we can receive feedback from receivers, and the options autostart and max-bitrate can be avoided as it is possible to establish a certain flow control between sender and receiver. Execute:
  server# udp-sender --file=./big_664.mpg --portbase 22345 --nopointopoint \
  --interface eth1 --ttl 3 --mcast-addr 232.43.211.234 --mcast-all-addr 225.1.2.3 \
  --nokbd --autostart 1 --blocksize 1400

As there is no limitation in the bitrate, you will notice that the transmission is much more faster than in the previous case.

- Using wireshark, have a look to the UDP packets sent towards the data multicast group.
  1. Are these UDP packets of maximum size?

2. Do you know all the headers involved? Is there any header related to the application udp-sender?
3. Which is the maximum value of the -blocksize option of udp-sender in order to traverse the tunnel with no problems?
4. Which percentage of packets are feedback packets?
5. Are these feedback packets unicast or multicast?

Live streaming

Until now you just sent files over the network, but notice that there are lots of applications using multicast that send information via streaming. So, you are going to send the same video via multicast from server to both host2 and host4, but at the same time your HOST machine will act as receiver playing the video in streaming. Prior to do this, you have to prepare your HOST machine to interact with the virtualized scenario. Next, we are going to provide some guidelines to play this video in streaming, but this should be considered as an example, and you may need to adapt the commands to the particular configuration of your HOST machine.

- First thing to do, you will need to install in your HOST machine udpcast (Use sudo apt-get install udpcast), which includes udp-sender and udp-receiver. Notice that the version of udpcast that you are installing in your HOST machine may be different to the one installed in the virtual machines.

- You will also need a video player that allows to play in streaming, for instance vlc (Use sudo apt-get install vlc-nox).

- Next, you have to configure the interface you are going to use in your HOST machine to interact with the virtualized scenario. Using ifconfig, assign the IP address 172.16.1.4 to SimNet3.
- Using route, create a new route to send the range 224.0.0.0/4 to SimNet3 (if not, multicast packets will be sent towards the default route).
- Execute udp-receiver in host2 and host4.
  host2# udp-receiver --file=big_664_in_host2.mpg --mcast-all-addr 225.1.2.3 \ --ttl 3 --interface eth1 --portbase 22345
  host4# udp-receiver --file=big_664_in_host4.mpg --mcast-all-addr 225.1.2.3 \ --ttl 3 --interface eth1 --portbase 22345
- Execute udp-receiver in your HOST machine, but redirecting the output of this application to the input video player vlc. As we mentioned, maybe you will need to adapt the command to the particular characteristics of your HOST machine. So, just as an example, execute:
  HOST$ udp-receiver --nokbd --mcast-rdv-address 225.1.2.3 --ttl 1 \ --interface SimNet3 --portbase 22345 | vlc --file-caching 0 - 2>/dev/null

Notice that the option -mcast-rdv-address of this upd-receiver is different from the one used previously (-mcast-all-addr), because the versions of udpcast are different.

- Finally, execute udp-sender at server:
  server# udp-sender --file=./big_664.mpg --portbase 22345 --nopointopoint \
  --interface eth1 --ttl 3 --mcast-addr 232.43.211.234 --mcast-all-addr 225.1.2.3 \
  --nokbd --autostart 1 --blocksize 1400

If everything is properly configured, the file will be transmitted via multicast from server to the three receivers:
host2 and host4 receiving the file, and HOST playing in streaming the video.

The end of the overlay

We mentioned that the configuration of this scenario pretends to be similar to the one used for the gradual deployment of the multicast technology (MBONE). In fact, there are other examples in which an overlay was constructed to help during the deployment of a new technology, connecting islands by using tunneling and encapsulation (for instance, the slow deployment of the IPv6 protocol). As time goes by, there are more nodes into the overlay (because they are aware of the new technology, for instance multicast), and few out of it. Finally, it is expected that all nodes are able to understand the new technology, so the overlay may have no sense (nor the tunnels).

To try to test this, let us try to translate this same idea to our multicast scenario. At the beginning we mentioned that RC was not a multicast router, so it was not possible for this router to be part of the multicast tree. Instead, you used IP tunnels to construct an overlay of multicast routers, bypassing RC by means of the GRE tunnels. This overlay works well, but you know that it is not totally transparent, and in fact there are some performance problems. For instance, every time you send a message through the multicast tree, this message is sent twice in SimNet2 (one through tunnel0 and another one through tunnel1), wasting bandwidth. You should have noticed this, this design is not able to use the network infrastructure efficiently and to achieve one of the main requirements of multicast: “messages must be sent only once over each link of the network“. Anyway, the overlay works pretty well, and this is just one more of the side effects caused by the use of tunneling techniques (overhead caused by the tunnel headers, MTU problems, TTL, etc.).

Now consider that RC evolves technologically, and now it is able to be a multicast router. So, it should be included as part of the multicast tree. In fact, now all routers are multicast routers, and the overlay and the tunnels do not have any sense.
So, as a final exercise for this lab session, include RC in the multicast tree and send again the video from server to the two receivers: host2 and host4

- Delete the static multicast tree by killing smcroute in all the multicast routers. Instead of doing it manually, execute the following command in your HOST:

```
HOST$ simctl ipmulticast exec mcclean
```

- Delete the tunnels by executing the following command in your HOST:

```
HOST$ simctl ipmulticast exec deltun
```

Notice that deleting the tunnels means that you cannot access to the private networks (using unicast).

- Use smcroute to create a tree from SimNet3 to SimNet0 and SimNet5. Notice that the tunnels do not exist, and RC should be included in the multicast tree.
    1. How many commands did you execute in every router?
    2. Do you think that static multicast routing is worthy? What would happen in case of having much more routers to configure?
- Use udp-sender and udp-receiver to make the video transmission.

1. How many times the same flow is sent over each link?
2. What is now the maximum block size allowed in udp-sender?
3. And the TTL?
4. Can you use a bidireccional transmission (with feedback information)?
5. In terms of performance, do you think that this solution works better than the overlay?

As you can see, you have executed several commands to make this configuration possible. Multicast static routing is easy to configure and to understand in case of having small networks, but obviously is not an scalable option in case of having mid to large network topologies. In that case, dynamical multicast routing protocols like Protocol Independent Multicast (PIM) should be used. We are not going to deal with dynamical multicast routing in this lab session. We hope that all these experiments helped you to understand how complicated is to understand multicast, even in this simplistic scenario with only some few routers.