Practice 9 Memo of TCGI

0.5 MAC multicast addresses

After running on the server `ping -c1 224.0.0.1` we capture the traffic on SimNet3. The results where the following:

1. We can only see one ICMP packet on the net. It is a echo ping request.
2. The TTL of this packet is 1 so it won't leave SimNet3.
3. The source and destination @IP in the IP header are:
    - src: 172.16.1.3
    - dst: 224.0.0.1
4. For the ethernet header:
    - src: Ethernet II, fe:fd:00:00:03:01
    - dst: IPv4mcast_01, 01:00:5e:00:00:01
5. We can see the direct mapping of the @IP on the @MAC.
    - The first byte indicates multicast
    - The second and third are fixed and belog also to multicast
    - The fourth one begins with a 0 and if we add the 7 lsb of the second byte of the IP results to all 0.
    - From the fifth to the seventh, they match the IP fields.

Now we are going to also listen on SimNet3 but ping to 232.0.0.1. These are the results:

1. The @MAC of the frame are:
    - Src: fe:fd:00:00:03:01 (fe:fd:00:00:03:01)
    - Dst: IPv4mcast_01 (01:00:5e:00:00:01)
2. We can see that the destination @MAC is the same as before althow the @IP is different.
3. The sustem will resolve this ambiguity at the IP layer where it will retrieve the full @IP.

After setting up the server to not ignore broadcast and multicas ICMP packate we run the same test as in the fisrt exercice but now with `-c3`. These are the results:

1. The pings are now fully working. But there is no reply messages on SimNet3. I supose that the server is answering itself the requests.
2. As mentioned before, no reply messages where seen.

Now we disable the same flag on R2 and re-run the ping:

1. Both the server and R2 are answering the ping.
2. Now we can see reply messages on SimNet3 generated by R2
3. Those messages are sent in aunicast way. ARP gets triggered and also the @IP are the unicast ones

- Also mention that the ping app reprts duplicated packages.

0.6 Multicast transmission in a subnet

0.6.1 Multicast configuration

In order to retrieve information about the joined groups by the host, we can explore the file `/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
```

If we reverse the @IP in hex `010000E0` and turn it into bynary, we'll see 224.0.0.1. The `netstat -gn` command can also be used.

- `/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
- `/proc/net/ip_mr_cache`: this file shows the contents of the Multicast Forwarding Cache.

0.6.3 udp-sender

To set up the video server, we execute on the server machine:

```
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
```

- `--file`: The realive path to the file.
- `--min-clients`: Automatically start as soon as a minimal number of clients have connected.
- `--portbase`: The app port starting point.
- `--nopointopoint`: Don't use point-to-point, even if there is only one single receiver.
- `--mcast-all-addr`: Uses a non-standard multicast address for the control connection (which is used by the sender and receivers to "find" each other). This is not the address that is used to transfer the data.
- `--mcast-addr`: Uses the given address for multicasting the data. If not specified, the program will automatically derive a multicast address from its own IP (by keeping the last 27 bits of the IP and then prepending 232).

The other parameters are self explanatory.

After running the command these are the results:

1. We can only see one UDP packet and two IGMPv3 packets.
2. In the UDP packet:
    - Internet Protocol Version 4, Src: 172.16.1.3, Dst: 225.1.2.3
    - TTL=1
3. The port numbers Src Port: 22346, Dst Port: 22345. It complies with the goven command.
4. The two IGMPv3 packets are the same (redundacy) and they are 0x22 type so Membership report. The @IP are Src: 172.16.1.3, Dst: 224.0.0.22. Due to the fact that we are using v3 of this protocol, all `Menbership Reports/Join group` are sent to this specific @IP with TTL=1 (as specified by the app) and payload the multicast addr of the grup the host is willing to join.

- It is also worth mentioning that those IGMP mesages get no response because there is not a m/c capable router

Now if we check the other console:

```
Every 1.0s: cat /proc/net/igmp
Thu May  9 13:15:37 2019

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. We can see that now the iface eth1 has another entry for another group.

2. The new entry stands for the @IP 225.1.2.3 which is the specified @IP to send all the control data.

After closing the server:

1. We can see two unanswered `Memebership Report/leave group` containing the @IP of the previosly mentioned control group.
2. The added entry has now been removed.

0.6.4 upd-receiver

Now we set up the receiver. By running:

```
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. We can see that both eth interfaces and the loopback have joined the group.

In order to receive the video we execute `R2# udp-receiver --file=big_664.mpg --mcast-all-addr 225.1.2.3 --ttl 1 --portbase 22345`.

1. We do not see any packages on SimNet2 but two UDP and two IGMPv3 in SimNet3.
2. We do not see them on SimNet3 because R2 has not been set up to fw multicast traffic through eth1.
3. The UDP datagrams have Src: 198.51.100.2, Dst: 225.1.2.3 and a TTL of 1.
4. The port numbers are Src Port: 22345, Dst Port: 22346. These correspond with the ones specified on the command.
5. We can see two other IGMPv3 Menbership Report Join packets from the control group.

- The objective of the UDP frames is to contact the server.
- There is no reponse for the IGMPv3 frames.

If we run:

```
R2:~# cat /proc/net/igmp
Idx     Device    : Count Querier       Group
Users Timer    Reporter
1       lo        :     1      V3
                                010000E0     1
0:00000000               0
5       eth2      :     2      V3
                                030201E1     1
0:00000000               0
                                010000E0     1
0:00000000               0
6       eth1      :     1      V3
                                010000E0     1
0:00000000               0
```

1. We can see that a new entry has been added for the control group to eth2.

After closin the receiver:

1. We can see two new IGMP Report Messages of type leave.
2. The before created entry has been removed.

0.6.5 Sending the video file in the subnet

After setting up the transmitter and the receiver:

1. The trasmission was completed succesfuly.
2. We can see 12514 frames on wireshark.

If we focus on the four initial UDP packets of short-length.

1. The purpose of this frmes is to stablish the connection betweem the server and the client.
2. The destination addr of the first UDP frame is the control multicast IP
3. If we convert 232.43.211.234 to hex we get e8.2b.d3.ea (0xe82bd3ea). We can see that this IP has bee transmutted inside the data field.
4. The client R2 was not listenning to the multicast IP.
5. The purpose is to discover any servers.
6. Before sending the frames, R2 only knows the control multicast IP.
7. It ias an unicast package anouncing the data IP addr tha will be used for the transmission. We recognize the data transmission multicast IP.

If we take a look to the data frames:

1. The destinantion @IP os this frames is the data m/c IP.
2. The size of the vast majority is 1514, the full MTU.
3. The unicast frames can be ACK or control frames from the app.

0.7 Multicast and routing

0.7.1 Configuring GRE tunnels

After creating the tunnels with the script, we run a ping from the server to the host2 and to the host4.

1. Both pings are working.
2. We can see the ICMP paquets on wireshark. They use an outer IP header with IPs of the router within the tunnel, an intermidiate GRE header and an innner IP header with the private addresses of teh hosts.
3. The size of the GRE header is 4 bytes.

0.7.3 Multicast static routing: smcroute

If we take alook to the file `/proc/net/ip_mr_vif` we can see that is empty. The we start the `smcroute` daemon on R2 and we can now 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
R2:~#
```

This is a list of all the ifaces that support multicast and some statistics. We now add a new route to R2 by running `R2:~# smcroute -a eth1 0.0.0.0 232.43.211.234 tunnel0`. So now any frame entering through eth1 with any origin address and destination address the multicast group 232.43.211.234 will be routed through the tunnel0. Now we execute `smcroute -j eth1 232.43.211.234` to make R2 to join the iface eth1 to the multicast group `232.43.211.234`:

```
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
```

1. The iface has been set up correctly to the desire @IP.
2. We can see two IGMP join frames from eth1 to the multicast 224.0.0.22.

If we now send a ping to the multicast grop with a scope of three hops(`ping -t3 -c1 232.43.211.234`) form the server:

1. The ping is working. We can see ICMP packets on SimNets 1, 2 and 3
2. We can see GRE encapsualtion on both SimNets 1 and 2 with a GRE header size of 4bytes.

```
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            588       7         0       0
00000 010110AC 00000000
 2 tunnel0           0       0       588       7
00000 016EA8C0 00000000
 3 tunnel1           0       0         0       0
00000 026EA8C0 00000000
```

3. After sending several pings, the sattistics have changed.
4. After executing `ip mroute show` we can see:

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

- It shows the src @IP of the m/c ping messages and the m/c IP itself.
- It also presents the Input interface and the output one.
- It represents the rout we configured before.

5. R1 is not forwarding the packet because it has not been yet configured as a multicast router.

- Its is also worth mentionin that R2 has been configured as an uniderctional static mmulticast route.

Now, in order to enable multicast on R1 and to be able to reach SiNet0, we run the following commands on R1:

```
R1:~# echo 0 >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
R1:~# smcroute -d
R1:~# smcroute -a tunnel0 0.0.0.0 232.43.211.234
eth1
R1:~# smcroute -j tunnel0 232.43.211.234
```

If we test now this configuration by running `server:~# ping -t3 -c1 232.43.211.234` we can see that:

1. In SimNet0 there is an echo-ping reqeust frame.
2. We can't see thow any echo reply on SimNet0 because no on is listening to the multicast traffic. But on SimNet1, we can see that the tunnel is responding the ping.
3. The TTL value is 1, the desired one. In order to reach teh scope of SimNet0, we use a TTL of 3.
4. It won't reach SimNet0 because of the scope.

0.7.4 Testing tools

ssmping

It allows to verfy if a host can recieve packets from a specific source by sending unicast and multicast UDP frames. After setting up all the Wiresharks we run on the server the ssmping daemon:

```
server:~# ssmpingd
```

We can see that the server and hosts statst by running (the output befor transmissions where identical):

```
server:~# netstat -gn
IPv6/IPv4 Group Memberships
Interface       RefCnt Group
--------------- ------ ---------------------
lo              1      224.0.0.1
eth1            1      224.0.0.1
lo              1      ff02::1
tunl0           1      ff02::1
gre0            1      ff02::1
sit0            1      ff02::1
eth1            1      ff02::1:ff00:301
eth1            1      ff02::1
```

Then we run on host2 `ssmping 172.16.1.3` and get the following output for `netstat -gn`:

```
host2:~# netstat -gn
IPv6/IPv4 Group Memberships
Interface       RefCnt Group
--------------- ------ ---------------------
lo              1      224.0.0.1
eth1            1      232.43.211.234
eth1            1      224.0.0.1
lo              1      ff02::1
tunl0           1      ff02::1
gre0            1      ff02::1
sit0            1      ff02::1
eth1            1      ff02::1:ff00:501
eth1            1      ff02::1
```

1. On host2, we can see the addtion of one entry for `eth1` for the group `232.43.211.234` . The outputs for the app confirm that the packets sent form host2 reach the server and the multicast packets sent by the server reach the host2.
2. According to this results, we can stablish that the multicast routing from the server to the host2 works.

If we take a look to the UDP frames, we can see that:

1. The first packet has Src: 192.168.0.2 (host2), Dst: 172.16.1.3 (server). It is an unicast frame.
2. The second frame is also an unicast one with the @IP src and dst shifted.
3. The third packet is the true multicast one from the server to the host2. This frame is testing the unidirection communication from the server to the host.

If we take a loo to the IGMP frames, we can see that:

1. We observe one IGMP Join message and two block ones. The multicast group is the one use by the app `232.43.211.234`.
2. We can see via `netstat -gn` that host2 joins and exits the group
3. There are not other IGMP messages on the other nets.

If we switch the roles for the server and the host, we can see taht the host2 is not able to reach the server via multicat traffic due to the static route being missing. Only the unicast traffic makes it to the server.

mcsender and emcast

Another pair of app to test the multicast routinthe multicast routing.

- Whe run `mcsender -t3 232.43.211.234:1234` on the server to send packets to the m/c group `232.43.211.234` using the UPD port 1234 with a scope of three hops.
- On the host2 we run `emcast 232.43.211.234:1234` to join the group. We us `netstat -gn` to verify that it joins it correctly.

The results are the following:

1. On host2 we can see the masseges sent by the server:

```
host2:~# emcast 232.43.211.234:12345
this is the test message from mclab/mcsender
this is the test message from mclab/mcsender
this is the test message from mclab/mcsender
this is the test message from mclab/mcsender
this is the test message from mclab/mcsender
this is the test message from mclab/mcsender
this is the test message from mclab/mcsender
this is the test message from mclab/mcsender
```

2. The multicast routing from the server to the host2 it is working.
3. The packets sent form the server are UDP multicast frames with Src: 172.16.1.3, Dst: 232.43.211.234.
4. We can see the join and leave IGMPv3 join and leave messages on SimNet0 form host2. The group involved is 232.43.211.234.

0.7.5 Trees with multiple branches

We are going to create a tree to reach all the m/c islands. To do so, we ru on R2:

```
R2:~# smcroute -k
R2:~# smcroute -d
R2:~# smcroute -a eth1 0.0.0.0 232.43.211.234
tunnel0 tunnel1
R2:~# smcroute -j eth1 232.43.211.234
```

In order to test this configuration, we are going to execute the following commands:

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

1. We can see that the multicat traffing works as expected between host2 ans the server but not between host4 and the server. Host2 only reports unicast traffic.
2. The reason might be that the R3 does not have any m/c capabilites enabled.
3. On SimNet4 there is multicast traffic whereas on SimNet5 not. This confirms our therory taht R3 is blocking the multicast traffic.

We enable the m/c capablilities by running `echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts` of R3 and rerun the tests. Now everything works as expected after creating the apropiate route on R3:

```
R3:~# smcroute -d
R3:~# smcroute -a tunnel1 0.0.0.0 232.43.211.234
eth1
```

0.7.6 Sending the video

In order to send the video via multicast, we need to set up the control group since the data group has been setup before. The grup @IP is `255.1.2.3`:

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

We test the configuration using `mcsender` and `emcast` in order to be able to select the @IP and everything works just fine.

We start the transmission of the video by running:

```
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
```

We can see that the MTU of the tunnels is enough to support frames of 1400 bytes so we run on the server (also the ttl for the scope is 3):

```
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
```

We can see that it transfers all the files but not in a reliables way because the control channel is not being used. This is due to the fact that we have set an unidirectional route from the server to the hosts.

0.7.7 Bidirectional tree

In order to offer reliability, we are going to create the bidirectiona tree for both channels. To do so, we need to stablish the route from host2 and hot4 to the server by running:

```
R1:~# smcroute -a eth1 0.0.0.0 225.1.2.3 tunnel0
R1:~# smcroute -a eth1 0.0.0.0 232.43.211.234
tunnel0
R3:~# smcroute -a eth1 0.0.0.0 225.1.2.3 tunnel1
R3:~# smcroute -a eth1 0.0.0.0 232.43.211.234
tunnel1
```

In order to test the configuration, we run teh same commands as before for the hosts but we avoid using the async and max-bitrate options.

1. The UDP frames fill the full MTU
2. All the headers are the standard ones.
3. The maximum blocksize is 1424 since the MTU of the tunnel is 1476 -20 -4 -8 -20 = 1424.
4. The percentage of feedback frames is more or less 2%
5. They are unicast

0.7.8 Live Streaming

It works if i set up de default gw to R2.

0.7.9 The end of the overlay

We run the following commands to set up the scenario without tunnels:

```
R2:~# smcroute -d
R2:~# smcroute -a eth1 0.0.0.0 232.43.211.234 eth2
R2:~# smcroute -a eth1 0.0.0.0 225.1.2.3 eth2

RC:~# echo 0 >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
RC:~# smcroute -d
RC:~# smcroute -a eth2 0.0.0.0 232.43.211.234 eth1
eth3
RC:~# smcroute -a eth2 0.0.0.0 225.1.2.3 eth1 eth3
RC:~# smcroute -a eth1 0.0.0.0 225.1.2.3 eth2
RC:~# smcroute -a eth3 0.0.0.0 225.1.2.3 eth2
```