Telecommunicatiesystemen : IGMP
README

Timo Truyts,
Raphael Assa

Contents

1 Install & Usage

1.1 Requirements

In order to execute this project, you must have click modular router installed on your system. If that's not the case, get it at : http://read.cs.ucla.edu/click/download
Since this is a C++ project, you'll obviously need a C++ compiler to build the program. Be sure to pass the following parameters when executing the configure file during the install of click modular router itself : `--disable-linuxmodule --enable-local --enable-etherswitch`
Finally run `make` and if all goes well, the Click application should get built succesfully.

1.2 Install procedure

To install our project, start by copy-pasting the contents of local into /path-to-click/elements/local where "path-to-click" represents the directory where you installed Click earlier. Next, run "make elemlist" followed by "make" from within the folder where Click was installed.

1.3 Running Click

Run Click with ipnetwork.click as argument by using the following command:
`/path-to-click/userlevel/click [-p <port>] /path-to-extracted-folder/ipnetwork.click`
where "[-p <port>]" is the optional port parameter, "path-to-click" the directory name where Click was installed and "path-to-extracted-folder" being the folder that contains this readme.
Note that if you want to use handlers, you must specify a port for Telnet to connect to.

2 List of files/elements

What follows here are small summaries for each of the files or elements that were made for the project.
Some examples will also be provided on how to use the elements in Click and its handlers.

2.1 igmpv3.hh

This header file contains the structs that define the IGMP query & report headers so that they generating IGMP packets becomes easier and the code will be more readable. Furthermore there's also a struct representing a group record for the same reasons.

2.2 IGMPq

Takes 1 input and has 2 outputs. The IGMPq element can be used in a `.click` script to simulate most of the work being done by a multicast router. Among one of those things being done is periodically sending general queries according to the QQIC which can be provided to the element as a parameter amongst many others. In a `.click` script you can configure the IGMPq element as follows :
IGMPq(IP 192.168.1.1, QQIC 10, QRV 2, MRC 50, LMQI 10, MASK 225.225.225.0)

1. IP : The IP address of this router's interface, mandatory input.
2. QQIC : Querier Query Interval Code, decides how often general queries are sent. A bigger QQIC means general queries are sent less often. This value is also used as a parameter to determine how to set the expiration timers. Range = [1, 255], Default = 125.

3. QRV : Querier Robustness Variable, partly decides how long expiration timers are set. Also the default value for the Startup Query Count which is one among many other variables which also have the QRV as their default value. Range = [0, 7], Default = 2.

4. MRC : Max Response Code, used to calculate the max response time represented in 0.1 seconds. This max response time partly determines the time that is set on the expiration timers.
   Range = [0, 255], Default = 100 (10.0 sec).
   Note that the max response time in seconds may not be greater than the time represented by QQIC, thus for example MRC = 100 and QQIC = 5 is not allowed.

5. LMQI : Last Member Query Interval in 0.1 seconds, partly determines how the expiration timers are set when sending a group specific query. The same note applies for LMQI as seen in MRC.
   Range = [0, 255], Default = 10 (1 sec).

6. MASK : Subnet mask to be used which will, in comnbination with the IP, determine whether packets must be examined or not.

2.2.1 Handlers

1. **qqic** : read/write, adjusts or reads the current QQIC
   E.g.: `write router/i1.qqic 30`

2. **qrv** : read/write, adjusts or reads the current QRV
   E.g.: `write router/i1.qrv 3`

3. **mrc** : read/write, adjusts or reads the current MRC
   E.g.: `write router/i1.mrc 60`

4. **lmqi** : read/write, adjusts or reads the current LMQI
   E.g.: `write router/i1.lmqi 20`

5. **info** : read, prints all group records including its sources and timers
   E.g.: `read router/i1.info`

2.3 IGMPr

Takes 1 input and has 3 outputs. The IGMPr element can be used in a `.click` script to simulate most of the work being done by a multicast member. It can generate reports by using the handlers which will be processed by an IGMPq element, as well as respond to queries sent by an IGMPq element. In a `.click` script you can provide an optional boolean paremeter for "SILENT" which will tell if the IGMPr element should start without generating packets. This option can be used to test the expiration and behaviour of timers in the IGMPq element.

2.3.1 Handlers

1. **join** : write, requires 1 parameter "GROUP" & has 2 optional parameters "INCLUDE" and "SOURCES". The first parameter must be a multicast address, indicating the group that you wish to join. The second parameter is false by default, i.e. exclude mode. The last parameter indicates the sources that you want to either include or exclude. Consecutive join calls on the same group will result in overwriting the current state with the provided parameters. (join report is generated upon execution)
   E.g.: `write client21/igmp.join 225.1.1.1, 0, 192.168.1.1`

2. **leave** : write, requires 1 parameter "GROUP" which indicates the group that you wish to leave. The group address must be present in the current state, otherwise an error is returned. (leave report is generated upon execution)
   E.g.: `write client21/igmp.leave 225.1.1.1`

3. **sources** : write, takes 2 mandatory arguments "GROUP" and "ACTION" & 1 optional "SOURCES". The group address must also be present in the current state, otherwise an error is returned. The action can either be "add", "del" or "flush" to respectively add or delete sources to the current state, or flush the list of sources from the current state. Any other action will return an error. The last parameter indicates which sources should be added or deleted. Note that if the selected action is "flush", then the last parameter is ignored. (state-change report is generated upon execution)
   E.g.: `write client21/igmp.sources 225.1.1.1, add, 192.168.1.1 192.168.1.2`

4. **mode** : write, requires 2 parameters "GROUP" and "INCLUDE" & 1 optional "SOURCES".
   The group address must again be present in the current state to avoid an error. The second parameter indicates whether you wish to include or exclude the sources in the current state. If this handler is called and no change in the current state has occurred, no further action is taken. The last parameter indicates the sources that will be used to overwrite the current state. If no sources are provided, the current state's sources are used instead. Note that the last parameter is ignored if the second parameter did not trigger a change in the current state. (state-change report is generated upon execution)
   E.g.: `write client21/igmp.mode 225.1.1.1, 1, 192.168.1.1 192.168.1.3`

5. **silent** : read/write, takes 1 optional argument "SILENT" which is true by default. It will force the IGMPr element not to transmit any reports. This handler can be used to change the silent mode during execution of a `.click` script to demonstrate the expiration of the timers in IGMPq.
   E.g.: `write client21/igmp.silent false`

6. **groups** : read, returns all groups with their corresponding mode & sources.
   E.g.: `read client21/igmp.groups`

2.4 IGMPsq

Takes 1 input and has 1 output. This element will only change the destination MAC & IP address for group specific queries since they don't report to 224.0.0.1 like general queries do.

2.5 TeeIGMP

Takes 1 input and has 1 or more outputs. This element will pass on all traffic on output 0, except when it's a multicast packet. In that case it is copied over all outputs except 0.