Embedding it better...

µTasker Document

µTasker – Multicasting and Internet Group Management Protocol (IGMP)

Table of Contents

1. Introduction

The four main types of Ethernet addressing are

- Unicast – a single destination address should receive the frame
- Subnet broadcast – all destination addresses of the subnet should receive the frame (*net-directed broadcast address*)
- Broadcast – all destination addresses should receive the frame (limited IPv4 broadcast address is 255.255.255.255)
- Multicast – a certain group of destination addresses should receive the frame

This document is concerned with multicasting, the fourth of the main addressing types. Multicasting works only with UDP, and not TCP, due to the fact that it is not possible to have a single TCP connection with multiple peers. It is useful for sending a single frame to multiple destinations without any (or only minimum) disturbance to other nodes in the network that don't need to receive the frame. A good example is transmitting video or radio on a network whereby only a group of receivers on the network actually want to watch or hear it. These types of signal don't need to be sent over connection oriented links since it makes no sense to repeat lost packets – the result of such a packet loss means that there may be a slight disturbance in the sound or the picture but nothing serious. Initially no receiver is watching or listening but a group of such receivers could decide to do so at any point in time and then start receiving the transmissions. The receivers with no interest in watching are not disturbed.

Multicast frames are addressed based on multicast MAC addressing and also multicast IP addressing. This addressing is described in the next section.

The protocol used to manage joining groups is IGMP which is then described in subsequent sections.

This document doesn't intent to replace RFCs specifying the full details of IGMP but instead concentrates on making sense of IGMP as well describing the implementation and use in the µTasker project.

Hosts may be level 0, level1 or level 2 hosts.

- Level 0 hosts *don't support IP multicast activity*
- Level 1 hosts *support sending multicast IP datagrams but not receiving such*
- Level 2 hosts *have full support for IP multicasting. This type of host requires IGMP*

2. Multicast Addressing

Multicast IP group addresses are class D IP addresses. The first 4 bits of the address are '1110'. This means that all addresses in the range 224.0.0.0 and 239.255.255.255 are multicast group addresses. There are some addresses in the range assigned to well-known addresses by the IANA (Internet Assigned Numbers Authority) – for example 224.0.0.1 means "all systems on this subnet – the all-host group", 224.0.0.2 means "all routers on this subnet) and 224.0.1.1 is for NTP (Network Time Protocol).

When a multicast IP address is used a group of hosts can receive on it. This set of hosts is called a host group. How the hosts join the group is discussed later in the IGMP section.

A multicast MAC address is recognised by the 8th bit of the first byte being set ('1'). Unicast addresses have this bit always clear ('0'). The MAC broadcast address `ff-ff-ff-ff-ff-ff` is therefore a special type of multicast address due to the fact that it is unconditionally destined to every node in a network sector.

The IANA owns an Ethernet address block which is defined partly for multicast use. Its high-order 24 bits is `00:00:5e`, meaning that the multicast MAC address range is from `00:00:5e:00:00:00` to `00:00:5e:7f:ff:ff`. Half of the block is allocated for multicasting use and, respecting the MAC multicast bit the actual MAC address range is from **`01:00:5e:00:00:00`** to **`01:00:5e:7f:ff:ff`**.

A multicast IP address cannot be translated to a unique MAC address since there are 248'720'625 possible IPv4 addresses in the multicast address range but only 8'388'608 MAC address. About 30 IP addresses therefore can share a single MAC address and the MAC address matching at the receiver is not perfect and requires a higher layer filtering to take place in addition, based on the IP address, to ensure that unwanted multicast frame don't reach the higher layer receiver.

2.1. Multicast Ethernet Filtering

Ethernet receivers usually allow frames to be filtered based on whether they are multicast frame (the 8th bit of the first byte set '1') which is the minimum multicast filtering that is offered (multicast promiscuous). Additional filtering may be offered by allowing a certain number of specific multicast addresses to be added; this results in a 30:1 filter accuracy for each MAC address for each multicast reception address. More often, however, a multicast hash filter is offered which typically gives a 64:1 match accuracy for a multicast address. In any case some higher layer (mostly software) filtering is always required. If more multicast addresses need to be received than the hardware filter supports the multicast promiscuous mode will need to be set to be able to receive them all.

2.2. Multicasting in a Single Physical Network

If the transmitter and all hosts are located in a single physical network they need only know what multicast IP address is being used and the transmitter send UDP frames to the IP address with the corresponding multicast MAC address. All receivers listen on the multicast MAC address and perform a check of the multicast IP address on ach frame passing the initial filter. There can be multiple receive processes at the host that use the received frame. No host needs to be aware of whether other hosts are listening and also the transmitter doesn't need to know whether there are hosts listening or how many may be listening. Whether a process at a host is listening (that is, is a member of the multicast group) is managed locally and so is very simple.

By default, routers at the borders of the physical network will not send multicast frames to other networks or out into the Internet.

The situation gets more complicated when the multicast group can span multiple networks and that is where IGMP and routers kick in. This is the discussed in the next section.

3. IGMP

There are three versions of IGMP
- IGMPv1 is described in **RFC 1112**
- IGMPv2 is described in **RFC 2236**
- IGMPv3 is described in **RFC 3376**

The corresponding IGMP MIB that can be present in SNMP implementations is described in **RFC 2933**.

IGMP is required when multicast datagrams must pass through multicast routers. Hosts send IGMP messages (report) when they want to join a multicast group so that routers can then specifically bridge them for them. Hosts may also leave a group at any time and the router learns about this when it regularly queries the host as to whether it still has processes belonging to the group.

Each host may have multiple processes receiving datagrams on the multicast address. This means that the each datagram is received by multiple processes if these exist. The host only reports joining a multicast group once and can have any number of processes listening. The host only leaves the multicast group when there are no more processes listening.

A host may be a member of more than one multicast group at the same time.

Hosts with multiple interfaces may have multicast group spanning multiple interfaces or restricted to a single interface.

Even when multicast routers allow multicast datagrams to be bridged to further networks the transmitter can restrict datagrams to a local physical network by setting the TTL (Time to Live) of the IP datagrams to 1. This means that the TTL needs to be controllable by the application sending the multi-casted data.

Reports and queries used by IGMP are always use TTL of 1 since such are restricted to the local network. Multicast routers never generate ICMP "time exceeded" errors when the TTL reaches zero.

Although there may often be one host in a network that sends and all others listen, any host in a multicast group can send and the datagram be received by all others. If a host has further listening processes on the interface its transmissions are looped back to these processes by default, although the loop back can be disabled on a per datagram basis.

Every level 2 host automatically joins the “all-host” group (224.0.0.1) when initialised and remains a member as long as it is active.

3.1. IGMP v1

IGMP v0 is obsolete (originally described in **RFC 988**) so the minimum version is IGMP v1. IGMP v1 support two IGMP message:

- Host Membership Query
- Host Membership Report

Queries and reports use a fixed length message with a 20 byte IP header and 8 byte IGMP message field. Both types contain a 16-bit checksum and a 32-bit class D IP address field, whereby this is unused and zeroes are sent in a membership query message.

Hosts that join a group send a report for each of its group memberships (with one or more processes) on the interface and then repeat it once at a random time between 0 and 10s. *This is to increase the probability that it will be successfully delivered since UDP is not guaranteed*. Each report is sent to the host group’s multicast address with a TTL of 1.

Multicast routers use the membership query message on a regular basis to discover which hosts groups (still) have members on the particular network interface. Queries are sent to the all-hosts address (224.0.0.1) and have a TTL value of 1.

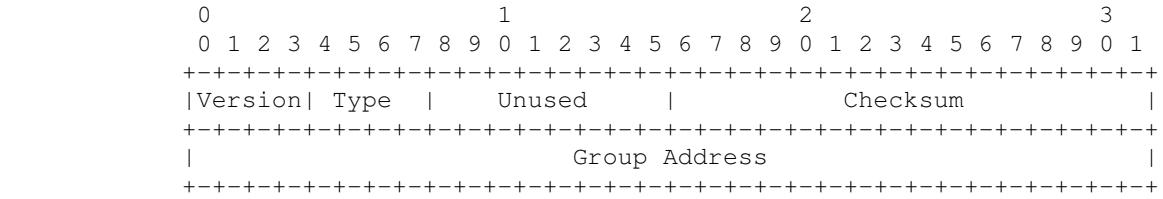
All hosts that still have multicast group membership(s) prepare a report in response for each group membership to a query. The response is sent to the multicast group address at a random time between 0 and 10s from the query reception. If a report is received from another multicast group member before the local report is sent out it is aborted since the multicast router only needs to know that there is still at least one host in the local network still in the group.

As a result, the only traffic on a single physical network without any multicast routers are IGMP reports issued when each host initially joins a group.

If a report delay timer is active when a query is received the query is ignored and the original timer left to timeout.

Group membership of the all-hosts group (224.0.0.1) is never reported.

IGMP v1 messages are encapsulated in IPv4 frames (using IP protocol number 2) and are always 8 bytes in length:

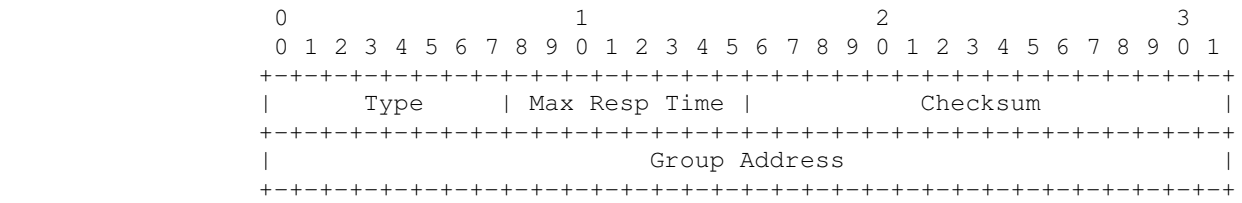


Version is 1 and two message types exist: 1 for host membership query and 2 for host membership report. The group address is set to 0.0.0.0 in queries and to the multicast group address in reports. The checksum is calculated over the 8 byte message, with the checksum field set to 0 for the calculation (*16-bit one's complement of the one's complement sum of the 8-octet IGMP message*).

3.2. IGMP v2

IGMP v1 allows hosts to join groups but leaving groups is a consequence of not reporting continued membership to IGMP queries. IGMP v2 add the capability of terminating membership.

The IP frame makes use of the IP Router Alert option as described in RFC 2113 in its IP header (this informs the router that it should examine the packet in more detail since it contains information relevant to the router) and makes use of an additional field in the 8 byte IGMP message (maximum response time), which is used in membership query messages. It gives the maximum allowed time to send a responding report in 1/10 second units (note that this value is fixed at 10s in IGMP v1).



The version has been removed from the type but is backward compatible with IGMP v1 due to its version 1 membership report being equivalent to that of IGMP v1.

- Version 1 Membership Report = 0x12
- Membership Query = 0x11 (general query or group-specific query)
- Version 2 Membership Report = 0x16
- Leave Group = 0x17

The group address is set to 0.0.0.0 when sending a general query and the queries multicast destination address set to the all-hosts address. The group address is set to the group address being queried when sending a group-specific query.

The IGMP v2 leave group message is sent when the last process of a host group leaves the group. However, if there are other host groups on the physical network this leave message is not sent if another host group caused the last report to be cancelled – that is, while the random report wait was active after the last query was received another host group send back its report, thus cancelling the local one.

When the host is operating with IGMP v2 it sends version 1 membership reports, rather than version 2 membership reports, in case there was an IGMP v1 router detected on the interface in question within the last 400 seconds. If there is an IGMP v1 router present the leave group message is not sent on the interface when a host leaves the group.

3.3. IGMP v3

IGMP v3 is designed to co-exist with IGMP v1 and IGMP v2.

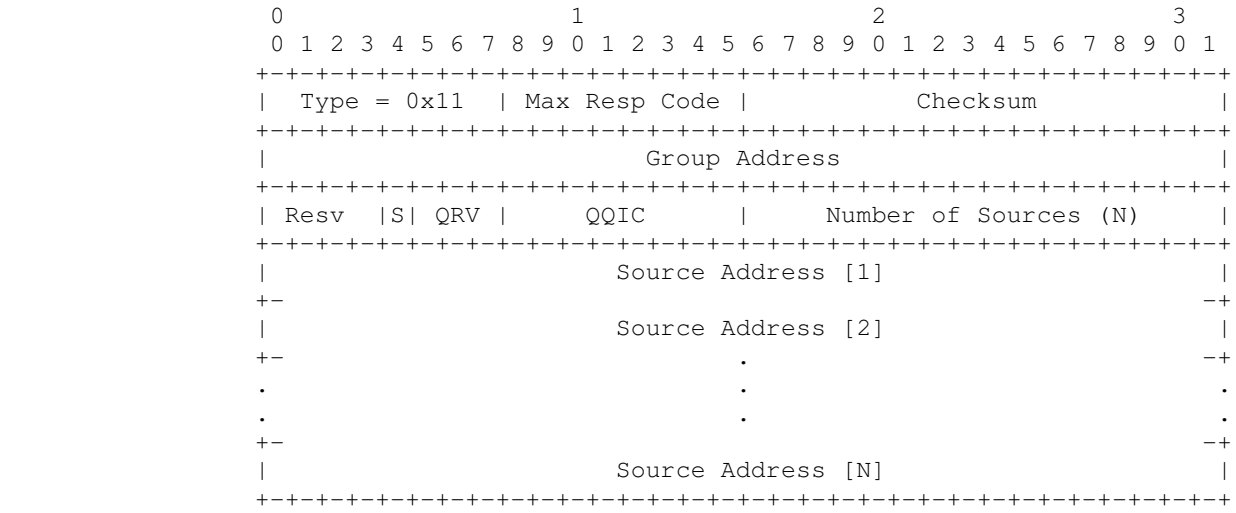
IGMP is considered to be implemented within the IP module (like ICMP). IGMP v3 is concerned with two message types:

- Membership Query 0x11
- Version 3 Membership Query = 0x22

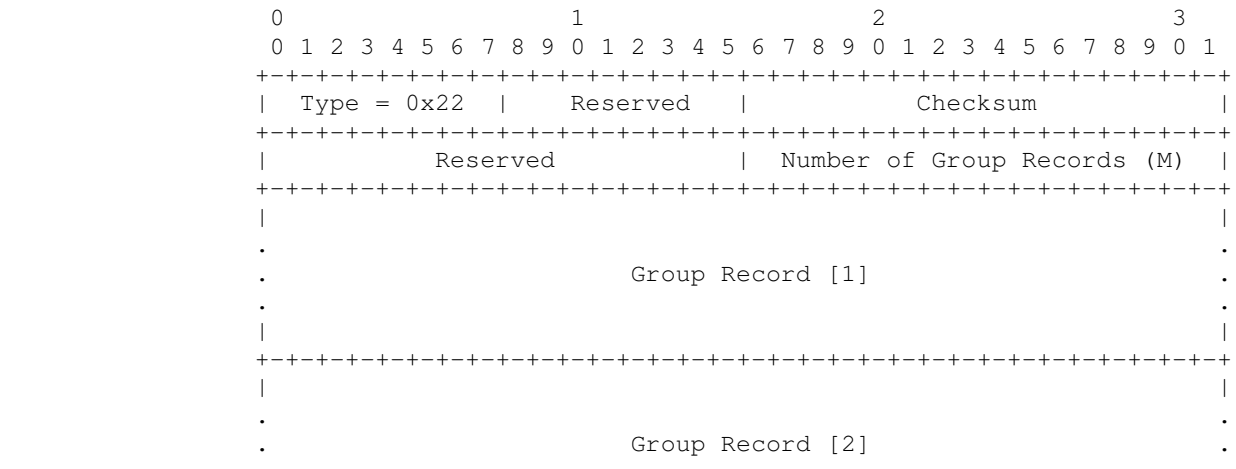
But also supports the IGMP v2 types 0x12, 0x16 and 0x17 for interoperability reasons

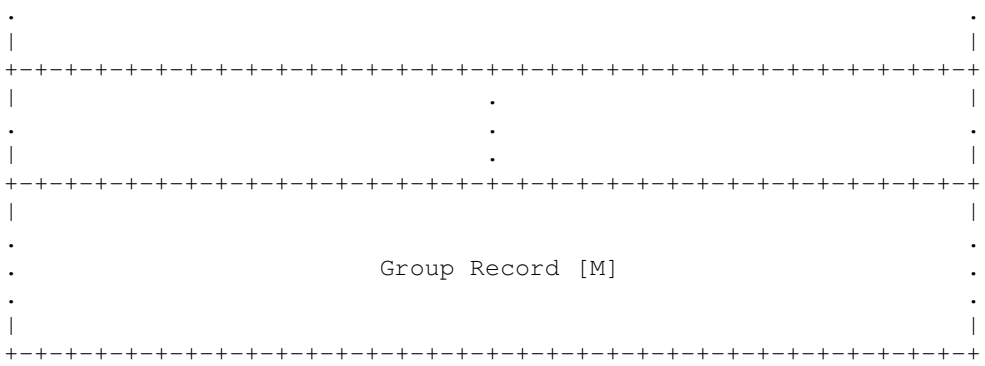
IGMP v3 messages are not fixed length as IGMP v1 and IGMP v2 messages but are variable length, depending on the message type:

Membership Query = 0x11



Membership Report = 0x22




Group Record [M]

4. Implementation and Use

IGMP is activated in the µTasker project by enabling the define `USE_IGMP`. By default IGMP v1 is supported. By enabling `USE_IGMP_V2` the operation respects IGMP v2 and by enabling `USE_IGMP_V3` the operation respects IGMP v3 – the highest enabled version is valid but with interoperability according to the IGMP specifications.

The maximum number of hosts that need to be able to exist at the same time and the maximum number of processes that each host can have are defined by

```
#define IGMP_MAX_HOSTS 2                  // host groups (in addition to all-hosts)
#define IGMP_MAX_PROCESSES 4  // the maximum number of processes in each host group
```

It is always possible to send a UDP multicast frame, even when no host multicast group membership exists. This is performed by using the function:

```
extern signed short fnSendUDP_multicast(USOCKET SocketHandle, int
iHostID, unsigned char *dest_IP, unsigned short usSourcePort,
unsigned short usRemotePort, unsigned char *ptrBuf, unsigned short
usDataLen, unsigned char ucOptions, unsigned char ucTTL);
```

Usually `ucTTL` is set to 1 for multicast UDP transmission so that the frame remains within the physical network. `ucOptions` can use `UDP_OPT_SEND_CS` to enable a checksum of the UDP data content and can also use `UDP_OPT_NO_LOOPBACK` if the multicast frame is not to be looped-back for reception by local host groups on this multicast address and this interface.

`iHostID` is set to 0 for transmissions that are not related to a host group process – *the use of this parameter is discussed later and can be used to avoid loop back of the frame to a particular local host group process.*

Should the destination IP address that is passed not be a multicast address the function returns `BAD_MULTICAST_ADDRESS`.

No multicast reception is possible when there are no multicast group members. The Ethernet multicast reception is disabled initially. To join a multicast group the following function is used:

```
extern int fnJoinMulticastGroup(unsigned char
ucMulticastGroupAddress[IPV4_LENGTH], USOCKET host_details, void
(*call_back)(int iHostID, unsigned short usSourcePort, unsigned
short usRemotePort, unsigned char *ptrBuf, unsigned short
usDataLen));
```

The return value is a unique reference to the host group and process that was added to the multicast group. This reference should be remembered so that a leave can be made at a later point in time. The first time a host group joins the Ethernet multicast reception is enabled on the all-hosts address as well as the group address (*usually this requires setting*

group multicast hash entries in the Ethernet hardware but the exact details may be dependent on the Ethernet controller used). If the host group belongs to multiple interfaces the multicast reception is enabled on all. The interfaces that the all-hosts group can receive on is defined by `IGMP_ALL_HOSTS_INTERFACES`. This is not needed when there is only one interface but needs to be defined form multiple-interface operation. The value `addInterface(DEFAULT_IP_INTERFACE)` would enable reception on only the default, but multiple interfaces can be used.

To leave the group the following function is used:

```
extern int fnLeaveMulticastGroup(int iHostID);
```

When a host group leaves membership (no more processes active) it also disables the multicast reception filter that it was using, if the same filter is not still used by another remaining group. If the removal of the host group leaves no further host groups active also the all-host multicast reception filter is disabled.

The call-back function passed with the `fnJoinMulticastGroup()` call is the call-back used to handle multicast UDP data reception for the process. If there are multiple processes belonging to a host group each of the process call-back functions are used to pass the data to each of the processes. *The order of the call-back handling depends on the order the processes were added but may change during use when processes are removed and added again (generally the order of the reception being passed to multiple processes should therefore not be relied on).*

The IGMP module sends reports when a host group is created and responds to queries as long as there is still at least one process in a host group. Random timer delays are respected, whereby the IGMP task's mono-stable timer is used for the function as long as no more than 1 host is defined. As soon as more than one host is defined global mono-stable timers are used instead.

An example of joining two multicast groups is given here, whereby two processes join the first host group and one the second:

```
static int iHostGroupID[3];
static unsigned char ucMulticastGroupAddress1[IPV4_LENGTH] = {224,10,10,10};
static unsigned char ucMulticastGroupAddress2[IPV4_LENGTH] = {224,0,0,251};

iHostGroupID[0] = fnJoinMulticastGroup(ucMulticastGroupAddress1,
(defineNetwork(DEFAULT_NETWORK) | defineInterface(DEFAULT_IP_INTERFACE)),
fnMulticastProcess1)                              // join a multicast group
iHostGroupID[1] = fnJoinMulticastGroup(ucMulticastGroupAddress1,
(defineNetwork(DEFAULT_NETWORK) | defineInterface(DEFAULT_IP_INTERFACE)),
fnMulticastProcess2);                             // join a multicast group
iHostGroupID[2] = fnJoinMulticastGroup(ucMulticastGroupAddress2,
(defineNetwork(SECOND_NETWORK) | defineInterface(DEFAULT_IP_INTERFACE)),
fnMulticastProcess3);                             // join a multicast group
```

The definitions of the network and interfaces are optional and 0 can be set when there is only one network and one interface in the system.

The following shows one process leaving the group later:

```
fnLeaveMulticastGroup(iHostGroupID[1]);
```

A simple example of a process call-back is:

```
static void fnMulticastProcess3(int iHostID, unsigned short usSourcePort, unsigned
short usRemotePort, unsigned char *ptrBuf, unsigned short usDataLen)
{
    typedef struct stMULTICAST_MESSAGE
    {
        UDP_HEADER     tUDP_Header;       // reserve header space
        unsigned char  data_content[10];  // reserve message space
    } MULTICAST_MESSAGE;
    MULTICAST_MESSAGE msg;
    uMemset(msg.data_content, 0xaa, sizeof(msg.data_content));
                                                    // test some message content
    fnSendUDP_multicast((defineNetwork(SECOND_NETWORK) |
defineInterface(DEFAULT_IP_INTERFACE)), iHostID, ucMulticastGroupAddress2, 0x6543,
0x3456, (unsigned char *)&msg, sizeof(msg.data_content), (UDP_OPT_SEND_CS), TTL_1);
}
```

This shows a multicast UDP message being sent on reception of a multicast UDP data packet (the reception data is not actually used). To be noted is the fact that the call-back process's host ID is passed together with the reception frame. This is then passed to `fnSendUDP_multicast()`, which is important since it stops the transmission from being looped back to the same process – if the option `UDP_OPT_NO_LOOPBACK` is not set the transmission is still looped back to all other processes in the local host group.

5. Testing IGMP and Multi-cast Transmission

The command line interface of the µTasker project (on UART, USB-CDC or Telnet) contains a simple IGMP and multicasting test interface in the LAN menu:

```
hosts               List multicast host groups
join                [multicast] join group
leave               [host ID] leave group
multi               <ip> send test frame
```

Initially there are no multicast host group members and the command "`hosts`" will inform of this fact.

It is possible to send a test multicast message using "`multi`" followed by a valid multicast IP address. This creates a small UDP packet and transmits it to this address.

Using the "`join`" command a process can be added to the multicast host members. Each process is assigned with a simple call-back that displays when it has received data (including its host ID). When a new host group is defines the IGMP process will add this to the group and send an initial report. Further processes can be added to the same IP multicast group up to the maximum processes that a single group can contain. Further host groups can be added (with different multicast IP address) up to the maximum host groups defined.

When there is a single processor, or multiple processes, in a host group each one will receive a copy of all received multicast UDP data for it.

The command "`leave`" is used, together with the ID of the host/process to free the process from the host group. If there are no further processes belonging to the host group, the host becomes free and a leave is sent when IGMP v2 is used.

The following shows a simple test session:

Test Interface commands and response	Notes
#hosts No host groups exist	Initially there are no host groups
#join 224.0.0.35 Joined as host/processor ID: 0x0002	A first process is added, which creates a host group on 224.0.0.35 *(a join report is sent)*
join 224.1.2.3 Joined as host/processor ID: 0x0001	As second host group joins on 224.1.2.3 *(a join report is sent)*
#hosts Host group with 1 Process(es): 0x0001 on multicast address 224.1.2.3 Host group with 1 Process(es): 0x0002 on multicast address 224.0.0.35	The host listing shows that there are two host group members and each has one process

`#join 224.1.2.3` `Joined as host/processor ID:  0x0101`	A second process is added to the host group on 224.1.2.3 (*no join report is sent*)
`#multi 224.0.0.35` `Mult-Rx: 0x0002 0x55`	When a multicast transmission is sent to 224.0.0.35 it is received by the since process on this host group

`#multi 224.1.2.3`
`Mult-Rx: 0x0001 0x55`
`Mult-Rx: 0x0101 0x55`
	A multicast transmission sent to 224.1.2.3 is received by all processes in this host group

`#hosts`
`Host group with 2 Process(es):`
`0x0001`
`0x0101`
`on multicast address 224.1.2.3`
`Host group with 1 Process(es):`
`0x0002`
`on multicast address 224.0.0.35`
	Here the 3 processes (and their IDs) are seen, in two host groups

`#leave 1` `Process freed`	One process is freed form the host group on 224.1.2.3 using its host/process ID 0x0001
`#multi 224.1.2.3` `Mult-Rx: 0x0101 0x55`	A reception on this multicast IP address I snow received by the remaining process only

`#hosts`
`Host group with 1 Process(es):`
`0x0101`
`on multicast address 224.1.2.3`
`Host group with 1 Process(es):`
`0x0002`
`on multicast address 224.0.0.35`	

`#leave 101` `Group left!`	When process ID 0x0101 is freed it causes the host group to leave the group (no more processes remaining) – *this causes an IGMP v2 leave to be sent if the conditions for it are fulfilled*
`#leave 2` `Group left!`	The final process leaves the final group – *again an IGMP v2 leave is sent*

#hosts No host groups exist	No host groups exist and so no reports are returned when an IGMP router sends queries.

6. Conclusion

IGMPv1 and IGMP v2 explained.

Implementation of IGMP v1 and v2 operation completed – supporting multiple networks and interfaces.

Work in progress-

- Add IGMP v3 plus implementation
- Add multiple network and multiple interface examples

Modifications:

V0.00 30.03.2014:
- Initial draft version

V0.01 14.04.2014:
- Second draft version