Network Working Group   M. Pullen
Request for Comments: 2490   George Mason University
Category: Informational   R. Malghan
                                              Hitachi Data Systems
                                                          L. Lavu
                                                      Bay Networks
                                                          G. Duan
                                                           Oracle
                                                             J. Ma
                                                        NewBridge
                                                           H. Nah
                                           George Mason University
                                                     January 1999

                 A Simulation Model for IP Multicast with RSVP

Status of this Memo

Copyright Notice

Abstract

   This document describes a detailed model of IPv4 multicast with RSVP
   that has been developed using the OPNET simulation package [4], with
   protocol procedures defined in the C language. The model was
   developed to allow investigation of performance constraints on
   routing but should have wide applicability in the Internet
   multicast/resource reservation community. We are making this model
   publicly available with the intention that it can be used to provide
   expanded studies of resource-reserved multicasting.

Table of Contents

1. Background

   The successful deployment of IP multicasting [1] and its availability
   in the Mbone has led to continuing increase in real-time multimedia
   Internet applications. Because the Internet has traditionally
   supported only a best-effort quality of service, there is
   considerable interest to create mechanisms that will allow adequate
   resources to be reserved in networks using the Internet protocol
   suite, such that the quality of real-time traffic such as video,
   voice, and distributed simulation can be sustained at specified
   levels. The RSVP protocol [2] has been developed for this purpose
   and is the subject of ongoing implementation efforts. Although the
   developers of RSVP have used simulation in their design process, no

simulation of IPmc with RSVP has been generally available for
analysis of the performance and prediction of the behavior of these
protocols. The simulation model described here was developed to fill
this gap, and is explicitly intended to be made available to the IETF
community.

2. The OPNET Simulation Environment

The Optimized Network Engineering Tools (OPNET) is a commercial
simulation product of the MIL3 company of Arlington, VA. It employs
a Discrete Event Simulation approach that allows large numbers of
closely-spaced events in a sizable network to be represented
accurately and efficiently. OPNET uses a modeling approach where
networks are built of components interconnected by perfect links that
can be degraded at will. Each component's behavior is modeled as a
state-transition diagram. The process that takes place in each state
is described by a program in the C language. We believe this makes
the OPNET-based models relatively easy to port to other modeling
environments. This family of models is compatible with OPNET 3.5.
The following sections describe the state-transition models and
process code for the IPmc and RSVP models we have created using
OPNET. Please note that an OPNET layer is not necessarily equivalent
to a layer in a network stack, but shares with a stack layer the
property that it is a highly modular software element with well
defined interfaces.

3. IP Multicast Model

The following processing takes place in the indicated modules. Each
subsection below describes in detail a layer in the host and the
router that can be simulated with the help of the corresponding OPNET
network layer or node layer or the process layer, starting from
physical layer.

3.1 Address format

The OPNET IP model has only one type of addressing denoted by "X.Y"
where X is 24 bits long and Y is 8 bits long, corresponding to an
IPv4 Class C network. The X indicates the destination or the source
network number and Y indicates the destination or the source node
number. In our model X = 500 is reserved for multicast traffic. For
multicast traffic the value of Y indicates the group to which the
packet belongs.

3.2 Network Layer

   Figure 1 describes an example network topology built using the OPNET
   network editor. This network consists of two backbone routers BBR1,
   BBR2, three area border routers ABR1, ABR2, ABR3 and six subnets F1,
   through F6. As OPNET has no full duplex link model, each connecting
   link is modeled as two simplex links enabling bidirectional traffic.
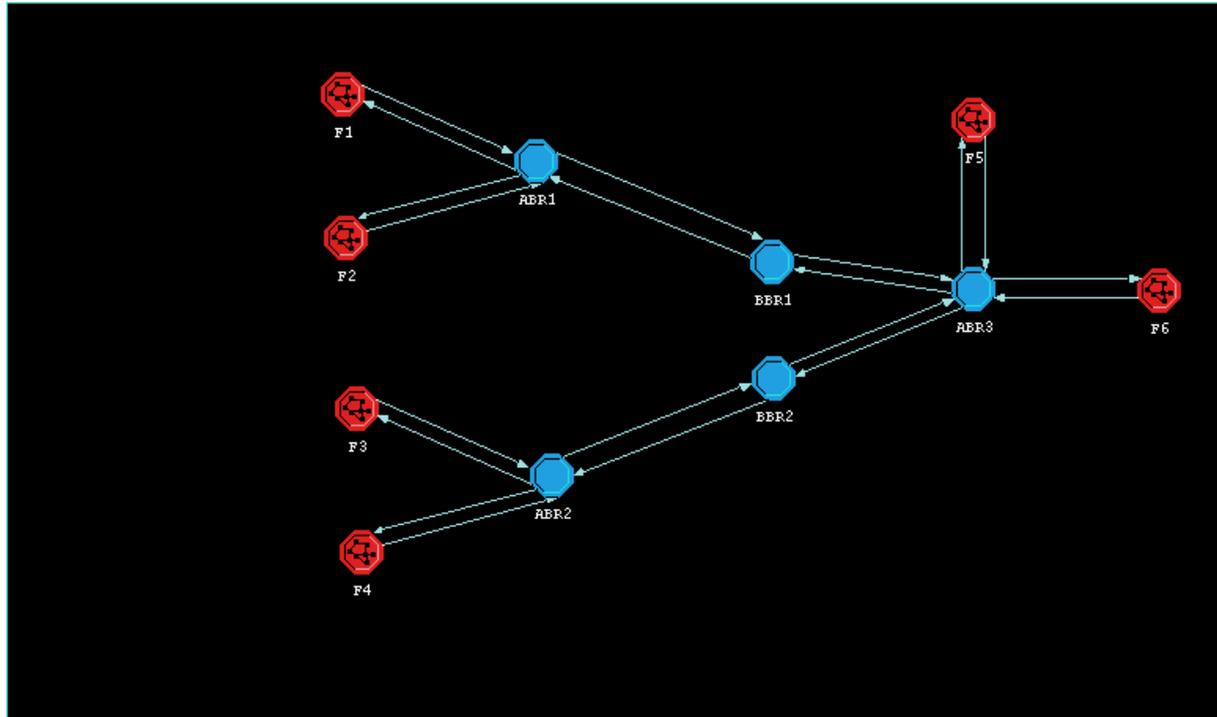

Figure 1: Network Layer of Debug Model

3.2.1 Attributes

   The attributes of the elements of the network layer are:

   a. Area Border Routers and Backbone Routers

      1. IP address of each active interface of each router
         (network_id.node_id)
      2. Service rate of the IP layer (packets/sec)
      3. Transmission speeds of each active interface (bits/sec)

   b. Subnets

      1. IP address of each active interface of the router in the subnet
      2. IP address of the hosts in each of the subnet.
      3. Service rate of the IP layer in the subnet router and the hosts.

   c. Simplex links

      1. Propagation delay in the links
      2. The process model to be used for simulating the simplex links
         (this means whether animation is included or not).

3.2.2 LAN Subnets

   Figure 2 shows the FDDI ring as used in a subnet. The subnet will
   have one router and one or more hosts. The router in the subnet is
   included to route the traffic between the FDDI ring or Ethernet in
   the corresponding subnet and the external network. The subnet router
   is connected on one end to Ethernet or FDDI ring and normally also is
   connected to an area border router on another interface (the area
   border routers may be connected to more than one backbone router). In
   the Ethernet all the hosts are connected to the bus, while in FDDI
   the hosts are interconnected in a ring as illustrated in Figure 2.

                    Figure 2: FDDI Ring Subnet Layer

   FDDI provides general purpose networking at 100 Mb/sec transmission
   rates for large numbers of communicating stations configured in a
   ring topology. Use of ring bandwidth is controlled through a timed
   token rotation protocol, wherein stations must receive a token and
   meet with a set of timing and priority criteria before transmitting
   frames. In order to accommodate network applications in which
   response times are critical, FDDI provides for deterministic
   availability of ring bandwidth by defining a synchronous transmission
   service. Asynchronous frame transmission requests dynamically share
   the remaining ring bandwidth.

   Ethernet is a bus-based local area network (LAN) technology. The
   operation of the LAN is managed by a media access protocol (MAC)
   following the IEEE 802.3 standard, providing Carrier Sense Multiple
   Access with Collision Detection (CSMA/CD) for the LAN channel.

3.3 Node layer

   This section discusses the internal structure of hosts and routers
   with the help of node level illustrations built using the Node editor
   of OPNET.

3.3.1 Basic OPNET elements

   The basic elements of a node level illustration are

   a. Processor nodes: Processor nodes are used for processing incoming
   packets and generating packets with a specified packet format.

   b. Queue node: Queue nodes are a superset of processor nodes. In
   addition to the capabilities of processor nodes, queue nodes also
   have capability to store packets in one or more queues.

   c. Transmitter and Receiver nodes: Transmitters simulate the link
   behavior effect of packet transmission and Receivers simulate the
   receiving effects of packet reception. The transmission rate is an
   attribute of the transmitter and receiving rate is an attribute of
   the receiver. These values together decide the transmission delay of
   a packet.

   d. Packet streams: Packet streams are used to interconnect the above
   described nodes.

   e. Statistic streams: Statistic streams are used to convey
   information between the different nodes: Processor, Queue,
   Transmitters and Receivers nodes respectively.

3.3.2 Host description

   The host model built using OPNET has a layered structure. Different
   from the OPNET layers (Network, Node and Process layer) that describe
   the network at different levels, protocol stack elements are
   implemented at OPNET nodes. Figure 3 shows the node level structure
   of a FDDI host.

a. MAC queue node: The MAC interfaces on one side to the physical
   layer through the transmitter (phy_tx) and receiver (phy_rx) and also
   provides services to the IP layer. Use of ring bandwidth is
   controlled through a timed token rotation protocol, wherein hosts
   must receive a token and meet with a set of timing and priority
   criteria before transmitting frames. When a frame arrives at the MAC
   node, the node performs one of the following actions:

      1. If the owner of the frame is this MAC, the MAC layer destroys
         the frame since the frame has finished circulating through the
         FDDI ring.
      2. if the frame is destined for this host, the MAC layer makes a
         copy of the frame, decapsulates the frame and sends the
         descapsulated frame (packet) to the IP layer. The original
         frame is transmitted to the next host in the FDDI ring.

  3. if the owner of the frame is any other host and the frame is not
     destined for this host, the frame is forwarded to the adjacent
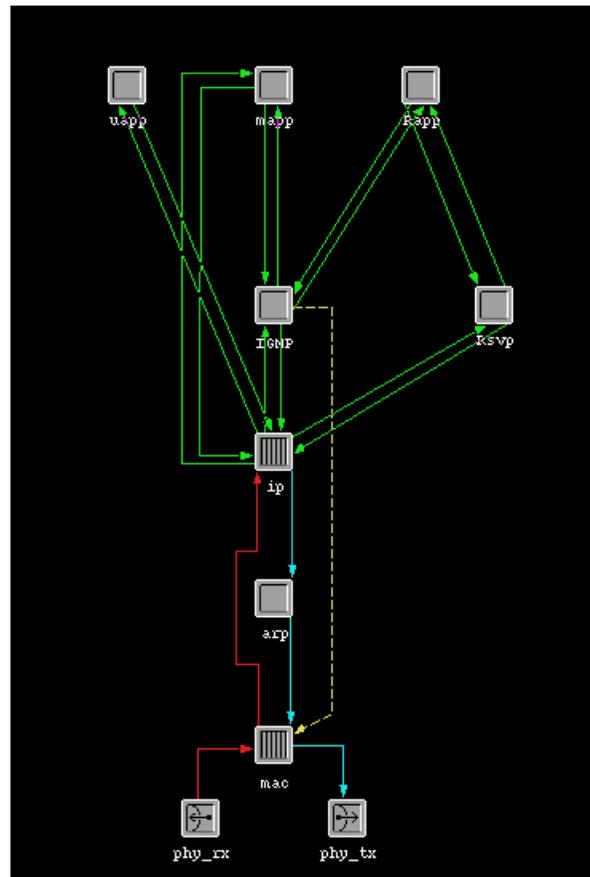   host.

Figure 3: Node Level of Host

b. ADDR_TRANS processor node: The next layer above the MAC layer is
   the addr_trans processor node. This layer provides service to the IP
   layer by carrying out the function of translating the IP address to
   physical interface address. This layer accepts packets from the IP
   layer with the next node information, maps the next node information
   to a physical address and forwards the packet for transmission. This
   service is required only in one direction from the IP layer to the
   MAC layer. Since queuing is not done at this level, a processor node
   is used to accomplish the address translation function, from IP to
   MAC address (ARP).

   c. IP queue node: Network routing/forwarding in the hierarchy is
   implemented here. IP layer provides service for the layers above
   which are the different higher level protocols by utilizing the

services provided by the MAC layer. For packets arriving from the
MAC layer, the IP layer decapsulates the packet and forwards the
information to an upper layer protocol based upon the value of the
protocol ID in the IP header. For packets arriving from upper layer
protocols, the IP layer obtains the destination address, calculates
the next node address from the routing table, encapsulates it with a
IP header and forwards the packet to the addr_trans node with the
next node information.

The IP node is a queue node. It is in this layer that packets incur
delay which simulates the processing capability of a host and
queueing for use of the outgoing link. A packet arrival to the IP
layer will be queued and experience delay when it finds another
packet already being transmitted, plus possibly other packets queued
for transmission. The packets arriving at the IP layer are queued
and operate with a first-in first-out (FIFO) discipline. The queue
size, service rate of the IP layer are both promoted attributes,
specified at the simulation run level by the environment file.

d. IGMP processor node: The models described above are standard
components available in OPNET libraries. We have added to these the
host multicast protocol model IGMP_host, the router multicast model
IGMP_gwy, and the unicast best-effort protocol model UBE.

The IGMP_host node (Figure 4) is a process node. Packets are not
queued in this layer. IGMP_host provides unique group management
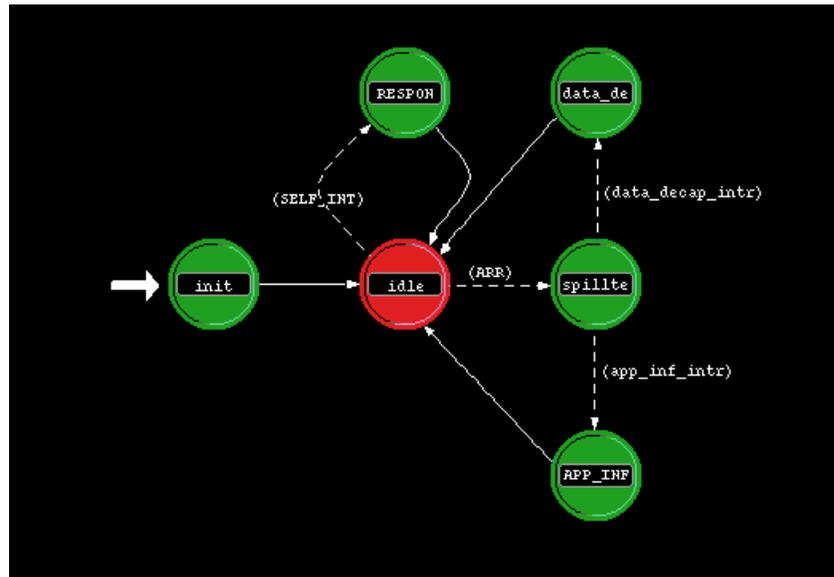services for the multicast applications utilizing the services

Figure 4: IGMP process on hosts

provided by the IP layer. IGMP_host maintains a single table which
consists of group membership information of the application above the
IGMP layer. The function performed by the IGMP_host layer depends
upon the type of the packet received and the source of the packet.

The IGMP_host layer expects certain type of packets from the
application layer and from the network:

1. Accept join group requests from the application layer (which can
   be one or more applications): IGMP_host maintains a table which
   consists of the membership information for each group. When a
   application sends a join request, it requests to join a specific
   group N. The membership information is updated. This new group
   membership information has to be conveyed to the nearest router
   and to the MAC layer. If the IGMP_host is already a member ofthis
   group (i.e. if another application above the IGMP_host is a member
   of the group N), the IGMP_host does not have to send a message to
   the router or indicate to the MAC layer. If the IGMP_host is not
   a member currently, the IGMP_host generates a join request for
   the group N (this is called a "response" in RFC 1112) and forwards
   it to the IP layer to be sent to the nearest router. In addition
   the IGMP_host also conveys this membership information to the MAC
   layer interfacing to the physical layer through the OPNET
   "statistic wire" connected from the IGMP_host to the MAC layer, so
   that the MAC layer knows the membership information immediately
   and begins to accept the frames destined for the group N. (An
   OPNET statistic wire is a virtual path to send information between
   OPNET models.)
2. Accept queries arriving from the nearest router and send responses
   based on the membership information in the multicast table at the
   IGMP_host layer: A query is a message from a router inquiring
   each host on the router's interface about group membership
   information. When the IGMP_host receives a query, it looks up the
   multicast group membership table, to determine if any of the
   host's applications are registered for any group. If any
   registration exists, the IGMP_host schedules an event to generate
   a response after a random amount of time corresponding to each
   active group. The Ethernet example in Figure 5 and the
   description in the following section describes the scenario.

```
         _____
         |        |         |          |
      ___|___   __|___    __|___    ____|___
     |       | |      |  |      |  |        |
     |  H1   | |  H2  |  |  H3  |  |   R    |
     |_____| |_____|  |_____|  |_____|
```
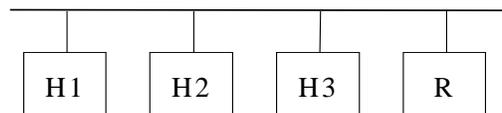
Figure 5: An Ethernet example of IGMP response schedule

The router R interfaces with the subnet on one interface I1 and to
reach the hosts. To illustrate this let us assume that hosts H1
and H3 are members of group N1 and H2 is a member of group N2.
When the router sends a query, all the hosts receive the query at
the same time t0. IGMP_host in H1 schedules an event to generate
a response at a randomly generated time t1 (t1 >= t0) which will
indicate the host H1 is a member of group N1. Similarly H2 will
schedule an event to generate a response at t2 (t2 >= t0)to
indicate membership in group N2 and H3 at t3 (t3 >= t0) to
indicate membership in group N3. When the responses are
generated, the responses are sent with destination address set to
the multicast group address. Thus all member hosts of a group
will receive the responses sent by the other hosts in the subnet
who are members of the same group.

In the above example if t1 < t3, IGMP_host in H1 will generate a
response to update the membership in group N1 before H3 does and
H3 will also receive this response in addition to the router. When
IGMP_host in H3 receives the response sent by H1, IGMP_host in H3
cancels the event scheduled at time t3, since a response for that
group has been sent to the router. To make this work, the events
to generate response to queries are scheduled randomly, and the
interval for scheduling the above described event is forced to be
less than the interval at which router sends the queries.

3. Accept responses sent by the other hosts in the subnet if any
   application layer is a member of the group to which the packet is
   destined.
4. Accept terminate group requests from the Application layer. These
   requests are generated by application layer when a application
   decides to leave a group. The IGMP_host updates the group
   information table and subsequently will not send any response
   corresponding to this group (unless another application is a
   member of this group). When a router does not receive any
   response for a group in certain amount of time on a specific
   interface, membership of that interface is canceled in that group.

e. Unicast best-effort (UBE) processor node: This node is used to
generate a best effort traffic in the Internet based on the User
Datagram Protocol (UDP). The objective of this node is to model the
background traffic in a network. This traffic does not use the
services provided by RSVP. UBE node aims to create the behaviors
observed in a network which has one type of application using the
services provided by RSVP to achieve specific levels of QoS and the
best effort traffic which uses the services provided by only the
underlying IP.

The UBE node generates traffic to a randomly generated IP address so
as to model competing traffic in the network from applications such
as FTP. The packets generated are sent to the IP layer which routes

the packet based upon the information in the routing table. The
attributes of the UBE node are:

1. Session InterArrival Time (IAT): is the variable used to schedule
   an event to begin a session. The UBE node generates an
   exponentially distributed random variable with mean Session IAT
   and begins to generate data traffic at that time.
2. Data IAT: When the UBE generates data traffic, the interarrival
   times between data packets is Data IAT. A decrease in the value of
   Data IAT increases the severity of congestion in the network.
3. Session-min and Session-max: When the UBE node starts generating
   data traffic it remains in that session for a random period which
   is uniformly distributed between Session-min and Session-max.

f. Multicast Application processor node: The application layer
consists of one or more application nodes which are process nodes.
These nodes use the services provided by lower layer protocols IGMP,
RSVP and IP. The Application layer models the requests and traffic
generated by Application layer programs. Attributes of the
application layer are:

1. Session IAT: is the variable used to schedule an event to begin a
   session. The Application node generates an exponentially
   distributed random variable with mean Session IAT and begins to
   generate information for a specific group at that time and also
   accept packets belonging to that group.
2. Data IAT: When Application node generates data traffic, the inter
   arrival time between the packets uses Data IAT variable as the
   argument. The distribution can be any of the available
   distribution functions in OPNET.
3. Session-min and Session-max: When an application joins a session
   the duration for which the application stays in that session is
   bounded by Session-min and Session-max. A uniformly distributed
   random variable between Session-min and Session-max is generated
   for this purpose. At any given time each node will have zero or
   one flow(s) of data.
4. NGRPS: This variable is used by the application generating
   multicast traffic to bound the value of the group to which an
   application requests the IGMP to join. The group is selected at
   random from the range [0,NGRPS-1].

3.3.3 Router description

   There are two types of routers in the model, a router serving a
   subnet and a backbone router. A subnet router has all the
   functions of a backbone router and in addition also has a
   interface to the underlying subnet which can be either a FDDI
   network or a Ethernet subnet. In the following section the subnet
   router will be discussed in detail.

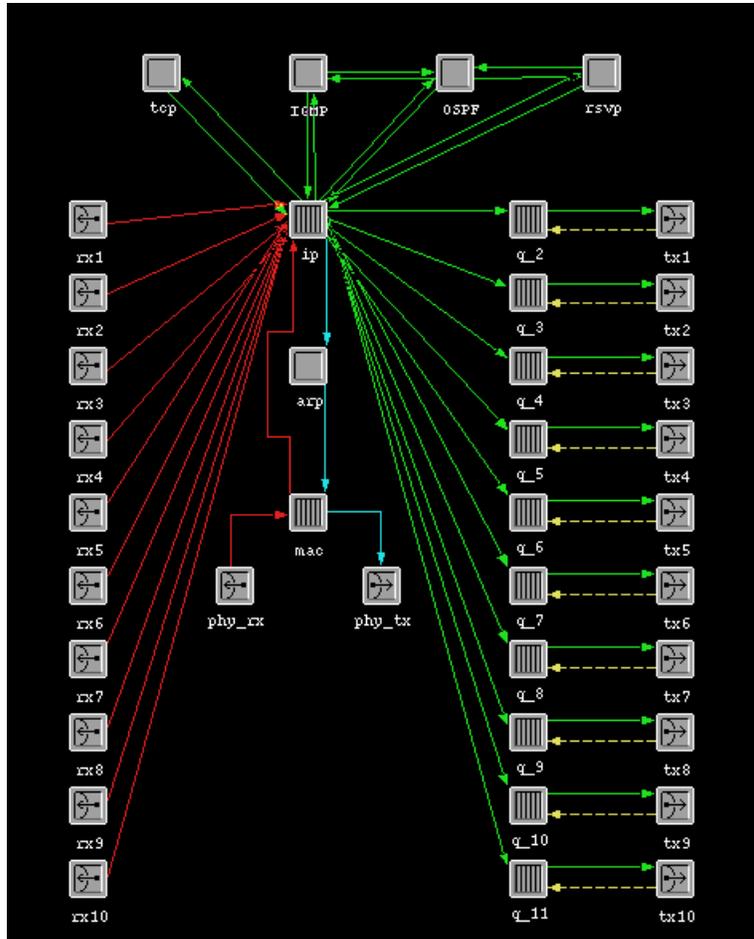Figure 6 shows the node level model of a subnet router.

Figure 6: Node Level of Gateway

a. The queueing technique implemented in the router is a
combination of input and output queueing. The nodes rx1 to rx10
are the receivers connected to incoming links. The router in
Figure 6 has a physical interface to the FDDI ring or Ethernet,
which consists of the queue node MAC, transmitter phy_tx, and the
receiver phy_rx. The backbone routers will not have a MAC layer.
The services provided and the functions of the MAC layer are the
same as the MAC layer in the host discussed above.

There is one major difference between the MAC node in a subnet
router and that in a host. The MAC node in a subnet router
accepts all arriving multicast packets unlike the MAC in a host

which accepts only the multicast packets for groups of which the
host is a member. For this reason the statistic wire from the IGMP
to MAC layer does not exist in a router (also because a subnet
router does not have an application layer).

b. Addr_trans: The link layer in the router hierarchy is the
addr_trans processor node which provides the service of
translating the IP address to a physical address. The addr_trans
node was described above under the host model.

c. IP layer: The router IP layer which provides services to the
upper layer transport protocols and also performs routing based
upon the information in the routing table. The IP layer maintains
two routing tables and one group membership table.

The tables used by the router model are:

1. Unicast routing table: This table is an single array of one
   dimension, which is used to route packets generated by the UDP
   process node in the hosts. If no route is known to a particular
   IP address, the corresponding entry is set to a default route.
2. Multicast routing table: This table is a N by I array where N is
   the maximum number of multicast groups in the model and I is the
   number of interfaces in the router. This table is used to route
   multicast packets. The routing table in a router is set by an
   upper layer routing protocol (see section 4 below). When the IP
   layer receives a multicast packet with a session_id corresponding
   to a session which is utilizing the MOSFP, it looks up the
   multicast routing table to obtain the next hop.
3. Group membership table: This table is used to maintain group
   membership information of all the interfaces of the router. This
   table which is also an N by I array is set by the IGMP layer
   protocol. The routing protocols use this information in the group
   membership table to calculate and set the routes in the Multicast
   routing table.

Sub-queues: The IP node has three subqueues, which implement queuing
based upon the priority of arriving packets from the neighboring
routers or the underlying subnet. The queue with index 0 has the
highest priority. When a packet arrives at the IP node, the packets
are inserted into the appropriate sub-queue based on the priority of
their traffic category: control traffic, resource- reserved traffic,
or best effort traffic. A non-preemptive priority is used in
servicing the packets. After the servicing, packets are sent to the
one of the output queues or the MAC. The packets progress through
these queues until the transmitter becomes available.

Attributes of the IP node are:

1. Unique IP address for each interface (a set of transmitter and
   receiver constitute an interface).
2. Service rate: the rate with which packets are serviced at the
   router.
3. Queue size: size of each of the sub queues used to store incoming
   packets based on the priority can be specified individually

d. Output queues: The output queues perform the function of queueing
the packets received by the IP layer when the transmitter is busy. A
significant amount of queuing takes place in the output queues only
if the throughput of the IP node approaches the transmission capacity
of the links. The only attribute of the queue node is:

Queue size: size of the queue in each queue node. If the queue is
full when a packet is received, that packet is dropped.

e. IGMP Node: Also modeled in the router is the IGMP for implementing
multicasting, the routing protocol, and RSVP for providing specific
QoS setup.

The IGMP node implements the IGMP protocol as defined in RFC 1112.
The IGMP node at a router (Figure 7) is different from the one at a
host. The functions of the IGMP node at a router are:

1. IGMP node at a router sends queries at regular intervals on all
   its interfaces.
2. When IGMP receives a response to the queries sent, IGMP updates
   the multicast Group membership table in the IP node and triggers
   on MOSPF LSA update.
3. Every time the IGMP sends a query, it also updates the multicast
   group membership table in the IP node if no response has been
   received on for the group on any interface, indicating that a
   interface is no longer a member of that group. This update is
   done only on entries which indicate an active membership for a
   group on a interface where the router has not received a response
   for the last query sent.
4. The routing protocol (see ection 4 below) uses the information in
   the group membership table to calculate the routes and update the
   multicast routing table.
5. When the IGMP receives a query (an IGMP at router can receive a
   query from a directly connected neighboring router), the IGMP node
   creates a response for each of the groups it is a member of on all
   the interfaces except the one through which the query was
   received.
6. The IGMP node on a backbone router is disabled, because IGMP is
   only used when a router has hosts on its subnet.
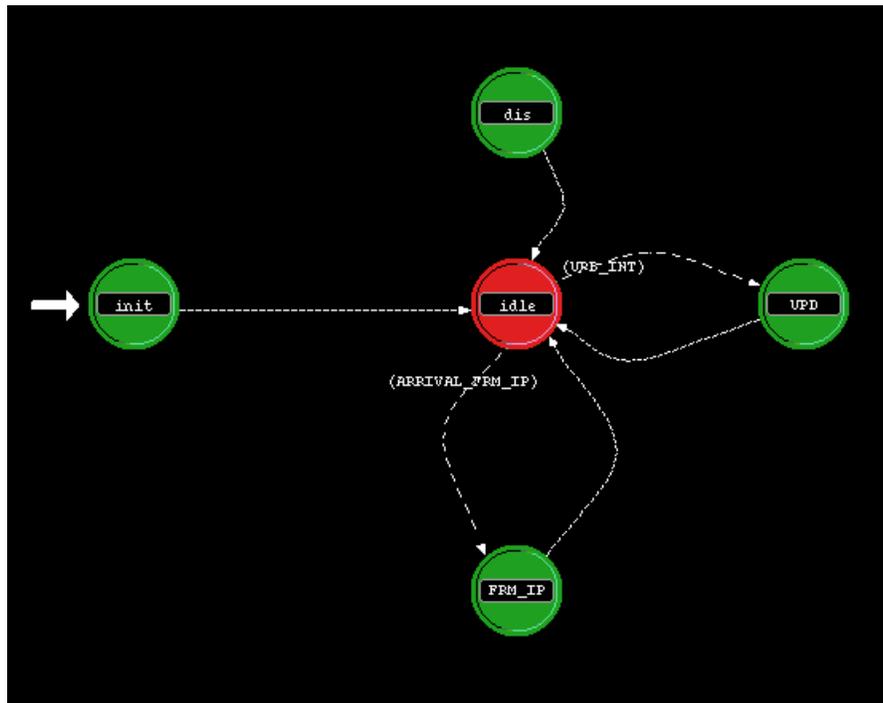
Figure 7: IGMP process on routers

4. RSVP model

   The current version of the RSVP model supports only fixed-filter
   reservation style. The following processing takes place in the
   indicated modules. The model is current with [2].

4.1 RSVP Application

4.1.1 Init

   Initializes all variables and loads the distribution functions for
   Multicast Group IDs, Data, termination of the session. Transit to
   Idle state after completing all the initializations.

4.1.2 Idle

   This state has transitions to two states, Join and Data_Send. It
   transit to Join state at the time that the application is scheduled
   to join a session or terminate the current session, transit to
   Data_Send state when the application is going to send data.

4.1.3 Join

   The Application will send a session call to local RSVP daemon. In
   response it receives the session Id from the Local daemon. This makes
   a sender or receiver call. The multicast group id is selected
   randomly from a uniform distribution. While doing a sender call the
   application will write all its sender information in a global session
   directory.

   If the application is acting as a receiver it will check for the
   sender information in the session directory for the multicast group
   that it wants to join to and make a receive call to the local RSVP
   daemon. Along with the session and receive calls, it makes an IGMP
   join call.

   If the application chooses to terminate the session to which it was
   registered, it will send a release call to the local RSVP daemon and
   a terminate call to IGMP daemon. After completing these functions it
   will return to the idle state.

4.1.4 Data_Send

   Creates a data packet and sends it to a multicast destination that it
   selects. It update a counter to keep track of how many packets that
   it has sent. This state on default returns to Idle state.

4.2 RSVP on Routers

   Figure 8 shows the process model of RSVP on routers.

4.2.1 Init

   This state calls a function called RouterInitialize which will
   initialize all the router variables. This state will go to Idle state
   after completing these functions.

4.2.2 Idle

   Idle state transit to Arr state upon receiving a packet.

4.2.3 Arr

   This state checks for the type of the packet arrived and calls the
   appropriate function depending on the type of message received.

a. PathMsgPro: This function was invoked by the Arr state when a path
   message is received. Before it was called, OSPF routing had been
   recomputed to get the latest routing table for forwarding the Path
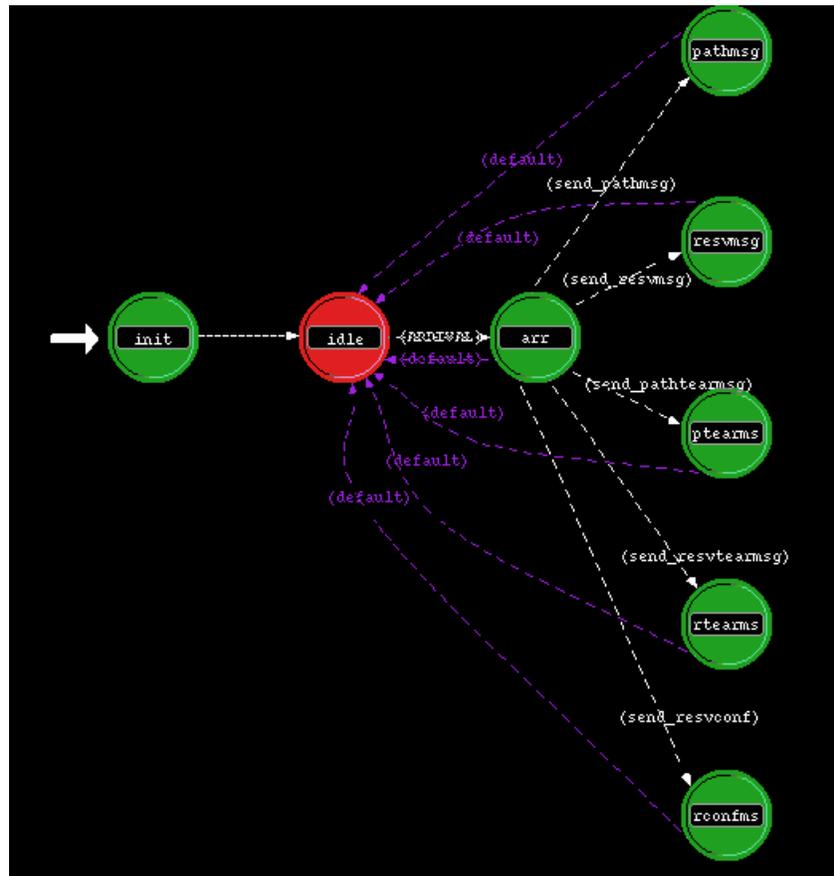   Message.

Figure 8: RSVP process on routers

1. It first checks for a Path state block which has a matching
   destination address and if the sender port or sender address or
   destination port does not match the values of the Session object
   of the Path state block, it sends an path error message and
   returns. (At present the application does not send any error
   messages, we print this error message on the console.)
2. If a PSB is found whose Session Object and Sender Template Object
   matches with that of the path message received, the current PSB
   becomes the forwarding PSB.
3. Search for the PSB whose session and sender template matches the
   corresponding objects in the path message and whose incoming
   interface matches the IncInterface. If such a PSB is found and the
   if the Previous Hop Address, Next Hop Address, and SenderTspec
   Object doesn't match that of path message then the values of path
   message is copied into the path state block and Path Refresh
   Needed flag is turned on. If the Previous Hop Address, Next Hop
   Address of PSB differs from the path message then the Resv Refresh

Needed flag is also turned on, and the Current PSB is made equal
to this PSB.

4. If a matching PSB is not found then a new PSB is created and and
   Path Refresh Needed Flag is turned on, and the Current PSB is made
   equal to this PSB.
5. If Path Refresh Needed Flag is on, Current PSB is copied into
   forwarding PSB and Path Refresh Sequence is executed. To execute
   this function called PathRefresh is used. Path Refresh is sent to
   every interface that is in the outgoing interfaces list of
   forwarding path state block.
6. Search for a Reservation State Block whose filter spec object
   matches with the Sender Template Object of the forwarding PSB and
   whose Outgoing Interface matches one of the entry in the
   forwarding PSB's outgoing interface list. If found then a Resv
   Refresh message to the Previous Hop Address in the forwarding PSB
   and execute the Update Traffic Control sequence.

b. PathRefresh: This function is called from PathMsgPro. It creates
the Path message sends the message through the outgoing interface
that is specified by the PathMsgPro.

c. ResvMsgPro: This function was invoked by the Arr state when a Resv
message is received.

1. Determine the outgoing interface and check for the PSB whose
   Source Address and Session Objects match the ones in the Resv
   message.
2. If such a PSB is not found then send a ResvErr message saying that
   No Path Information is available. (We have not implemented this
   message, we only print an error message on the console.)
3. Check for incompatible styles and process the flow descriptor list
   to make reservations, checking the PSB list for the sender
   information. If no sender information is available through the PSB
   list then send an Error message saying that No Sender information.
   For all the matching PSBs found, if the Refresh PHOP list doesn't
   have the Previous Hop Address of the PSB then add the Previous Hop
   Address to the Refresh PHOP list.
4. Check for matching Reservation State Block (RSB) whose Session and
   Filter Spec Object matches that of Resv message. If no such RSB is
   found then create a new RSB from the Resv Message and set the
   NeworMod flag On. Call this RSB as activeRSB. Turn on the Resv
   Refresh Needed Flag.
5. If a matching RSB is found, call this as activeRSB and if the
   FlowSpec and Scope objects of this RSB differ from that of Resv
   Message copy the Resv message Flowspec and Scope objects to the
   ActiveRSB and set the NeworMod flag On.
6. Call the Update Traffic Control Sequence. This is done by calling
   the function UpdateTrafficControl

7. If Resv Refresh Needed Flag is On then send a ResvRefresh message
   for each Previous Hop in the Refresh PHOP List. This is done by
   calling the ResvRefresh function for every Previous Hop in the
   Refresh PHOP List.

d. ResvRefresh: this function is called by both PathMsgPro and
ResvMsgPro with RSB and Previous Hop as input. The function
constructs the Resv Message from the RSB and sends the message to the
Previous Hop.

e. PathTearPro: This function is invoked by the Arr state when a
PathTear message is received.

1. Search for PSB whose Session Object and Sender Template Object
   matches that of the arrived PathTear message.
2. If such a PSB is not found do nothing and return.
3. If a matching PSB is found, a PathTear message is sent to all the
   outgoing interfaces that are listed in the Outgoing Interface list
   of the PSB.
4. Search for all the RSB whose Filter Spec Object matches the Sender
   Template Object of the PSB and if the Outgoing Interface of this
   RSB is listed in the PSB's Outgoing interface list delete the RSB.
5. Delete the PSB and return.

f. ResvTearPro: This function is invoked by the Arr state when a
ResvTear message is received.
1. Determine the Outgoing Interface.
2. Process the flow descriptor list of the arrived ResvTear message.
3. Check for the RSB whose Session Object, Filter Spec Object matches
   that of ResvTear message and if there is no such RSB return.
4. If such an RSB is found and Resv Refresh Needed Flag is on send
   ResvTear message to all the Previous Hops that are in Refresh PHOP
   List.
5. Finally delete the RSB.

g. ResvConfPro: This function is invoked by the Arr state when a
ResvConf message is received. The Resv Confirm is forwarded to the IP
address that was in the Resv Confirm Object of the received ResvConf
message.

h. UpdateTrafficControl: This function is called by PathMsgPro and
ResvMsgPro and input to this function is RSB.

1. The RSB list is searched for a matching RSB that matches the
   Session Object, and Filter Spec Object with the input RSB.
2. Effective Kernel TC_Flowspec are computed for all these RSB's.
3. If the Filter Spec Object of the RSB doesn't match the one of the
   Filter Spec Object in the TC Filter Spec List then add the Filter
   Spec Object to the TC Filter Spec List.

   4. If the FlowSpec Object of the input RSB is greater than the
      TC_Flowspec then turn on the Is_Biggest flag.
   5. Search for the matching Traffic Control State Block(TCSB) whose
      Session Object, Outgoing Interface, and Filter Spec Object matches
      with those of the Input RSB.
   6. If such a TCSB is not found create a new TCSB.
   7. If matching TCSB is found modify the reservations.
   8. If Is_Biggest flag is on turn on the Resv Refresh Needed Flag
      flag, else send a ResvConf Message to the IP address in the
      ResvConfirm Object of the input RSB.

4.2.4 pathmsg: The functions to be done by this state are done through
   the function call PathMsgPro described above.

4.2.5 resvmsg: The functions that would be done by this state are done
   through the function call ResvMsgPro described above.

4.2.6 ptearmsg: The functions that would be done by this state are done
   through the function call PathTearPro described above.

4.2.7 rtearmsg: The functions that would be done by this state are done
   through the function call ResvTearPro described above.

4.2.8 rconfmsg: The functions that would be done by this state are done
   through the function call ResvConfPro described above.

4.3 RSVP on Hosts

   Figure 9 shows the process of RSVP on hosts.

4.3.1 Init

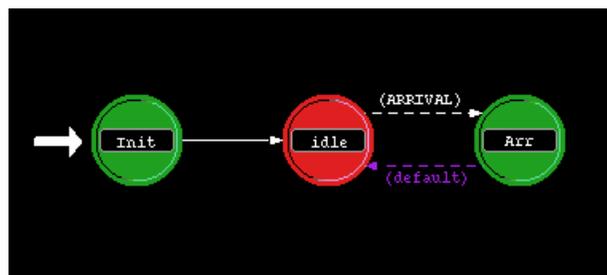   Initializes all the variables. Default transition to idle state.

Figure 9: RSVP process on hosts

4.3.2 idle

   This state transit to the Arr state on packet arrival.

4.3.3 Arr

   This state calls the appropriate functions depending on the type of
   message received. Default transition to idle state.

   a. MakeSessionCall: This function is called from the Arr state
   whenever a Session call is received from the local application.

   1. Search for the Session Information.
   2. If one is found return the corresponding Session Id.
   3. If the session information is not found assign a new session Id to
      the session to the corresponding session.
   4. Make an UpCall to the local application with this Session Id.

   b. MakeSenderCall: This function is called from the Arr state
   whenever a Sender call is received from the local application.

   1. Get the information corresponding to the Session Id and create a
      Path message corresponding to this session.
   2. A copy of the packet is buffered and used by the host to send the
      PATH message periodically.
   3. This packet is sent to the IP layer.

   c. MakeReserveCall: This function is called from the Arr state
   whenever a Reserve call is received from the local application. This
   function will create and send a Resv message. Also, the packet is
   buffered for later use.

   d. MakeReleaseCall: This function is called from the Arr state
   whenever a Release call is received from the local application. This
   function will generate a PathTear message if the local application is
   sender or generates a ResvTear message if the local application is
   receiver.

4.3.4 Session

   This state's function is performed by the MakeSessionCall function.

4.3.5 Sender

   This state's function is han by the MakeSenderCall function.

4.3.6 Reserve

   This state's function is performed by the MakeReserveCall function.

4.3.7 Release

   This state's function is performed by the MakeReleaseCall function.

5. Multicast Routing Model Interface

   Because this set of models was intended particularly to enable
   evaluation by simulation of various multicast routing protocols, we
   give particular attention in this section to the steps necessary to
   interface a routing protocol model to the other models. We have
   available implementations of DVMRP and OSPF, which we will describe
   below. Instructions for invoking these models are contained in a
   separate User's Guide for the models.

5.1 Creation of multicast routing processor node

   Interfacing a multicast routing protocol using the OPNET Simulation
   package requires the creation of a new routing processor node in the
   node editor and linking it via packet streams. Packet streams are
   unidirectional links used to interconnect processor nodes, queue
   nodes, transmitters and receiver nodes. A duplex connection between
   two nodes is represented by using two unidirectional links to connect
   the two nodes to and from each other.

   A multicast routing processor node is created in the node editor and
   links are created to and from the processors(duplex connection) that
   interact with this module, the IGMP processor node and the IP
   processor node. Within the node editor, a new processor node can be
   created by selecting the button for processor creation (plain gray
   node on the node editor control panel) and by clicking on the desired
   location in the node editor to place the node. Upon creation of the
   processor node, the name of the processor can be specified by right
   clicking on the mouse button and entering the name value on the
   attribute box presented. Links to and from this node are generated
   by selecting the packet stream button (represented by two gray nodes
   connected with a solid green arrow on the node editor control panel),
   left clicking on the mouse button to specify the source of the link
   and right clicking on the mouse button to mark the destination of the
   link.

5.2 Interfacing processor nodes

   The multicast routing processor node is linked to the IP processor
   node and the IGMP processor node each with a duplex connection. A
   duplex connection between two nodes is represented by two uni-
   directional links interconnecting them providing a bidirectional flow
   of information or interrupts, as shown in Figure 6. The IP processor
   node (in the subnet router) interfaces with the multicast routing
   processor node, the unicast routing processor node, the Resource
   Reservation processor node(RSVP), the ARP processor node(only on
   subnet routers and hosts), the IGMP processor node, and finally the
   MAC processor node (only on subnet routers and hosts) each with a
   duplex connection with exceptions for ARP and MAC nodes.

5.2.1 Interfacing ARP and MAC processor nodes

   The service of the ARP node is required only in the direction from
   the IP layer to the MAC layer(requiring only a unidirectional link
   from IP processor node to ARP processor node). The MAC processor
   node on the subnet router receives multicast packets destined for all
   multicast groups in the subnet, in contrast to the MAC node on subnet
   hosts which only receives multicast packets destined specifically for
   its multicast group. The MAC node connects to the IP processor node
   with a single uni-directional link from it to the IP node.

5.2.2 Interfacing IGMP, IP, and multicast routing processor nodes

   The IGMP processor node interacts with the multicast routing
   processor node, unicast routing processor node, and the IP processor
   node. Because the IGMP node is linked to the IP node, it is thus able
   to update the group membership table(in this model, the group
   membership table is represented by the local interface(interface 0)
   of the multicast routing table data structure) within the IP node.
   This update triggers a signal to the multicast routing processor node
   from the IGMP node causing it to reassess the multicast routing table
   within the IP node. If the change in the group membership table
   warrants a modification of the multicast routing table, the multicast
   routing processor node interacts with the IP node to modify the
   current multicast routing table according to the new group membership
   information updated by IGMP.

5.2.2.1 Modification of group membership table

   The change in the group membership occurs with the decision at a host
   to leave or join a particular multicast group. The IGMP process on
   the gateway periodically sends out queries to the IGMP processes on
   hosts within the subnet in an attempt to determine which hosts
   currently are receiving packets from particular groups. Not
   receiving a response for a pending IGMP host query specific to a
   group indicates to the gateway IGMP that no host belonging to the
   particular group exists in the subnet. This occurs when the last
   remaining member of a multicast group in the subnet leaves. In this
   case the IGMP processor node updates the group membership able and
   triggers a modification of the multicast routing table by alerting
   the multicast routing processor node. A prune message specific to
   the group is initiated and propagated upward establishing a prune
   state for the interface leading to the present subnet, effectively
   removing this subnet from the group-specific multicast spanning tree
   and potentially leading to additional pruning of spanning tree edges
   as the prune message travels higher up the tree. Joining a multicast
   group is also managed by the IGMP process which updates the group
   membership table leading to a possible modification of the multicast
   routing table.

5.2.2.2 Dependency on unicast routing protocol

   The multicast routing protocol is dependent on a unicast routing
   protocol (RIP or OSPF) to handle multicast routing. The next hop
   interface to the source of the packet received, or the upstream
   interface, is determined using the unicast routing protocol to trace
   the reverse path back to the source of the packet. If the packet
   received arrived on this upstream interface, then the packet can be
   propagated downstream through its downstream interfaces (excluding
   the interface in which the packet was received). Otherwise, the
   packet is deemed to be a duplicate and dropped, halting the
   propagation of the packet downstream. This repeated reverse path
   checking and broadcasting eventually generates the spanning tree for
   multicast routing of packets. To determine the reverse path forward
   interface of a received multicast packet propagated up from the IP
   layer, the multicast routing processor node retrieves a copy of the
   unicast routing table from the IP processor node and uses it to
   recalculate the multicast routing table in the IP processor node.

5.3 Interrupt Generation

   Using the OPNET tools, interrupts to the multicast routing processor
   node are generated in several ways. One is the arrival of a
   multicast packet along a packet stream (at the multicast routing
   processor node) when the packet is received by the MAC node and
   propagated up the IP node where upon discarding the IP header
   determination is made as to which upper layer protocol to send the
   packet. A second type of interrupt generation occurs by remote
   interrupts from the IGMP process alerting the multicast routing
   process of an update in the group membership table. A third occurs
   when the specific source/group (S,G) entry for a multicast packet
   received at the IP node does not exist in the current multicast
   routing table and a new entry needs to be created. The IP node
   generates an interrupt to the multicast routing processor node
   informing it to create a new source/group entry on the multicast
   routing table.

5.3.1 Types of interrupts

   The process interrupts generated within the OPNET model can be
   handled by specifying the types of interrupts and the conditions for
   the interrupts using the interrupt code, integer number representing
   the condition for a specific interrupt. The conditions for
   interrupts are specified on the interrupt stream linking the
   interrupt generating state and the state resulting from the
   interrupt. For self-interrupts (interrupts occurring among states
   within the same process), interrupts of type OPC_INTRPT_SELF are
   used. For remote interrupts (interprocess interrupts), the
   conditions for specific interrupts are specified from the idle state

to the state resulting from the interrupt within the remote process.
The remote interrupts are of type, OPC_INTRPT_REMOTE. A third type
of interrupt is the OPC_INTRPT_STRM, which is triggered when packets
arrive via a packet stream, indicating its arrival. The condition of
this interrupt is also specified from the idle state to the resultant
state by the interrupt condition stream defined by a unique interrupt
code. For all of these interrupts, the interrupt code is provided
within the header block (written in C language) of the interrupted
process. When the condition for the interrupt becomes true, a
transition is made to the resultant state specified by the interrupt
stream.

5.3.2 Conditions for interrupts

Several interrupt connections exist to interface the IGMP processor
node, IP processor node , and the multicast routing processor node
with each other in the present OPNET Simulation Model. Also, the IP
processor node interfaces with the unicast routing protocol which
interfaces with the IGMP processor node. An OPC_INTRPT_STRM
interrupt is generated when a multicast packet arrives via a packet
stream from the IP processor node to the multicast routing processor
node. A remote interrupt of type, OPC_INTRPT_REMOTE, is generated
from the IGMP process to the IP process when a member of a group
relinquishes membership from a particular group or a new member is
added to a group. This new membership is updated in the group
membership table located in the IP node by the IGMP process which
also generates a remote interrupt to the multicast routing protocol
process, causing a recalculation of the multicast routing table in
the IP module.

5.4 Modifications of modules in the process model

Modifications of routing protocol modules (in fact all of the modules
in the process model) are made transparently throughout the network
using the OPNET Simulation tools. An addition or modification of a
routing module in any subnet will reflect on all the subnets.

6. OSPF and MOSPF Models

OSPF and MOSPF models [5] are implemented in the OSPF model
containing fourteen states. They only exist on routers. Figure 10
shows the process model. The following processing takes place in the
indicated modules.

6.1 init

This state initializes all the router variables. Default transition
to idle state.

6.2 idle

   This state has several transitions. If a packet arrives it transits
   to arr state. Depending on interrupts received it will transit to
   BCOspfLsa, BCMospfLsa, hello_pks state. In future versions, links
   coming up or down will also cause a transition.

6.3 BCOspfLsa

   Transition to this state from idle state is executed whenever the
   condition send_ospf_lsa is true, which happens when the network is
   being initialized, and when ospf_lsa_refresh_timout occurs. This
   state will create Router, Network, Summary Link State Advertisements
   and pack all of them into an Link State Update packet. The Link State
   Update Packet is sent to the IP layer with a destination address of
   AllSPFRouters.

Figure 10: OSPF and MOSPF process model on routers

6.4 BCMospfLsa

   Transition to this state from idle state is executed whenever the
   condition send_mospf_lsa is true. This state will create Group
   Membership Link State Advertisement and pack them into Mospf Link

State Update Packet. This Mospf Link State Update Packet is sent to
IP layer with a destination address of AllSPFRouters.

6.5 arr

The arr state checks the type of packet that is received upon a
packet arrival. It calls the following functions depending on the
protocol Id of the packet received.

a. OspfPkPro: Depending on the type of OSPF/MOSPF packet received the
function calls the following functions.
1. HelloPk_pro: This function is called whenever a hello packet is
   received. This function updates the router's neighbor information,
   which is later used while sending the different LSAs.
2. OspfLsUpdatePk_pro: This function is called when an OSPF LSA
   update packet is received (router LSA, network LSA, or summary
   LSA). If the Router is an Area Border Router or if the LSA belongs
   to the Area whose Area Id is the Routers Area Id, then it is
   searched to determine whether this LSA already exists in the Link
   State database. If it exists and if the existing LSA's LS Sequence
   Number is less than the received LSA's LS Sequence Number the
   existing LSA was replaced with the received one. The function
   processes the Network LSA only if it is a designated router or
   Area Border Router. It processes the Summary LSA only if the
   router is a Area Border Router. The function also turns on the
   trigger ospfspfcalc which is the condition for the transition from
   arr state to ospfspfcalc.
3. MospfLsUpdatePk_pro: This function is called when a MOSPF LSA
   update packet is received. It updates the group membership link
   state database of the router.

6.6 hello_pks

Hello packets are created and sent with destination address of
AllSPFRouters. Default transition to idle state.

6.7 mospfspfcalc

The following functions are used to calculate the shortest path tree
and routing table. This state transit to upstr_node upon detupstrnode
condition.

a. CandListInit: Depending upon the SourceNet of the datagram, the
candidate lists are initialized.

b. MospfCandAddPro: The vertex link is examined and if the other end
of the link is not a stub network and is not already in the candidate
list it is added to the candidate list after calculating the cost to
that vertex. If this other end of the link is already on the shortest

path tree and the calculated cost is less than the one that shows in
the shortest path tree entry update the shortest path tree to show
the calculated cost.

c. MospfSPFTreeCalc: The vertex that is closest to the root that is
in the candidate list is added to the shortest path tree and its link
is considered for possible inclusions in the candidate list.

d. MCRoutetableCalc: Multicast routing table is calculated using the
information of the MOSPF shortest Path tree.

6.8 ospfspfcalc

The following functions are used in this state to calculate the
shortest path tree and using this information the routing table.
Transition to ospfspfcalc state on ospfcalc condition. This is set to
one after processing all functions in the state.

a. OspfCandidateAddPro: This function initializes the candidate list
by examining the link state advertisement of the Router. For each
link in this advertisement, if the other end of the link is a router
or transit network and if it is not already in the shortest-path tree
then calculate the distance between these vertices. If the other end
of this link is not already on the candidate list or if the distance
calculated is less than the value that appears for this other end add
the other end of the link to candidate list.

b. OspfSPTreeBuild: This function pulls each vertex from the
candidate list that is closest to the root and adds it to the
shortest path tree. In doing so it deletes the vertex from the
candidate list. This function continues to do this until the
candidate list is empty.

c. OspfStubLinkPro: In this procedure the stub networks are added to
shortest path tree.

d. OspfSummaryLinkPro: If the router is an Area Border Router the
summary links that it has received is examined. The route to the Area
border router advertising this summary LSA is examined in the routing
table. If one is found a routing table update is done by adding the
route to the network specified in the summary LSA and the cost to
this route is sum of the cost to area border router advertising this
and the cost to reach this network from that area border router.

e. RoutingTableCalc: This function updates the routing table by
examining the shortest path tree data structure.

6.9 upstr_node

   This state does not do anything in the present model. It transitions
   to DABRA state.

6.10 DABRA

   If the router is an Area Border Router and the area is the source
   area then a DABRA message is constructed and send to all the
   downstream areas. Default transition to idle state.

7. DVMRP Model

   The DVMRP model is implemented based on reference [6], DVMRP version
   3. There are nine states. The DVMRP process only exists on Routers.
   Figure 11 shows the states of the DVMRP process.

7.1 Init

   Initialize all variables, routing table and forwarding table and load
   the simulation parameters. It will transit to the Idle state after
   completing all the initializations.

7.2 Idle

   The simulation waits for the next scheduled event or remotely invoked
   event in the Idle State and transit to the state accordingly. In the
   DVMRP model, Idle State has transitions to Probe_Send, Report_Send,
   Prune_Send, Graft_Send, Arr_Pkt, Route_Calc and Timer states.

7.3 Probe_Send State

   A DVMRP router sends Probe messages periodically to inform other
   DVMRP routers that it is operational. A DVMRP router lists all its
   known neighbors' addresses in the Probe message and sends it to All-
   DVMRP-Routers address. The routers will not process any message that
   comes from an unknown neighbor.

7.4 Report_Send

   To avoid sending Report at the same time for all DVMRP routers, the
   interval between two Report messages is uniformly distributed with
   average 60 seconds. The router lists source router's address,
   upstream router's address and metric of all sources into the Report
   message and sends it to All-DVMRP-Routers address.
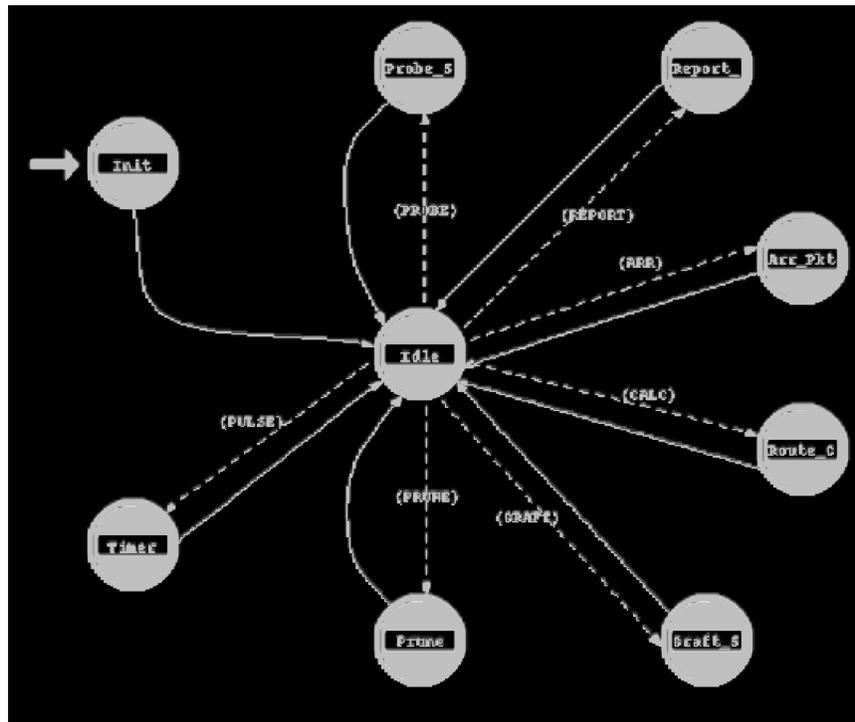
Figure 11. DVMRP process on routers

7.5 Prune_Send

   The transition to this state is triggered by the local IGMP process.
   When a host on the subnetwork drops from a group, the IGMP process
   asks DVMRP to see if the branch should be pruned.

   The router obtains the group number from IGMP and checks the IP
   Multicast membership table to find out if there is any group member
   that is still in the group. If the router determines that the last
   host has resigned, it goes through the entire forwarding table to
   locate all sources for that group. The router sends Prune message,
   containing source address, group address and prune lifetime,
   separately for each (source, group) pair and records the row as
   pruned in the forwarding table.

7.6 Graft_Send

   The transition to this state is triggered by the local IGMP process.
   Once a multicast delivery has been pruned, Graft messages are
   necessary when a host in the local subnetwork joins into the group. A
   Graft message sent to the upstream router should be acknowledged hop
   by hop to the root of the tree guaranteeing end-to-end delivery.

The router obtains the group number from IGMP and go through the
forwarding table to locate all traffic sources for that group. A
Graft message will be sent to the upstream router with the source
address and group address for each (source, group) pair. The router
also setups a timer for each Graft message waiting for an
acknowledgement.

7.7 Arr_Pkt

All DVMRP control messages will be sent up to DVMRP layer by IP. The
function performed by the DVMRP layer depends upon the type of the
message received.

a. Probe message: The router checks the neighbors' list in Probe
message, update its their status to indicate the availability of its
neighbors.

b. Report message: Based on exchanging report messages, the routers
can build the Multicast delivery tree rooted at each source. A
function called ReportPkPro will be called to handle all possible
situations when receiving a report message. If the message is a
poison reverse report and not coming from one of the dependent
downstreams, the incoming interface should be added to the router's
downstream list. If the message is not a poison reverse report but it
came from one of the downstreams, this interface should be deleted
from the downstreams list. And then, the router compared the metric
got from the message with the metric of the current upstream, if the
new metric is less than the older one, the router's upstream
interface should be updated.

c. Prune message: The router extracts the source address, group
address and prune lifetime, marks the incoming interface as pruned in
the dependent downstream list of the (source, group) pair. If all
downstream interfaces have been pruned, the router will send a prune
message to its upstream.

d. Graft message: The router extracts the source and group address,
active the incoming interface in the dependent downstream list of the
(source, group) pair. If the (source, group) pair has been pruned,
the router will reconnect the branch by sending a graft message to
its upstream interface.

e. Graft Acknowledge message: The router extracts the source and
group address, clear the graft message timer of the (source, group)
pair in the forwarding table.

7.8 Route_Calc

   The transition to this state is triggered by the local IP process.
   Once the IP receives a packet, it will fire a remote interrupt to the
   DVMRP and ask the DVMRP to prepare the outgoing interfaces for the
   packet. The DVMRP process obtains the packet's source address and
   group address from the IP and checks the (source, group) pairs in the
   forwarding table to decide the branches that have the group members
   on the Multicast delivery tree. The Group Membership Table on IP will
   be updated based on this knowledge.

7.9 Timer

   This state is activated once every second. It checks the forwarding
   table, if the Graft message acknowledgment timer is expired, The
   router will retransmit the Graft message to the upstream. If the
   prune state lifetime timer is expired, the router will graft this
   interface so that the downstream router can receive the packets to
   the group again. The router also checks if the (source, group) pair
   is pruned by the upstream router, if so, it will send a graft message
   to the upstream interface.

8. Simulation performance

   Our simulations of three network models with MOSPF routing have
   showed good Scalability of the protocol. The running platform we used
   is a SGI Octane Station with 512 MB main memory and MIPS R10000 CPU,
   Rev 2.7. Here we list the real running time of each model along with
   its major elements and the packet inter-arrival times for the streams
   generated in the hosts.

Simulated time	Debug Model 11 Routers, 12 Hosts Reserve Data 0.01s Best-effort Data 0.01s	Intermediate Model 42 routers, 48 hosts Reserve Data 0.02s Best-effort Data 0.025s	Large Model 86 routers, 96 hosts Reserve Data 0.02s Best-effort Data 0.025s
100 s	3 hours	14 hours	30 hours
200 s	7 hours	30 hours	- - -

9. Future work

   We hope to receive assistance from the IPmc/RSVP development
   community within the IETF in validating and refining this model. We
   believe it will be a useful tool for predicting the behavior of
   RSVP-capable systems.

10. Security Considerations

   This RFC raises no security considerations.

11. References

   [1] Deering, S., "Host Requirements for IP Multicasting", STD 5,
       RFC 1112, August 1989.

   [2] Braden, R., Zhang, L., Berson, S., Herzog, S. and S. Jamin,
       "Resource Reservation Protocol (RSVP) -- Version 1 Functional
       Specification", RFC 2205, September 1997.

   [3] Wroclawski, J., "The Use of RSVP with IETF Integrated Services",
       RFC 2210, September 1997.

   [4] MIL3 Inc., "OPNET Modeler Tutorial Version 3", Washington, DC,
       1997

   [5] Moy, J., "Multicast Extensions to OSPF", RFC 1584, March 1994.

   [6] Pusateri, T., "Distance Vector Multicast Routing Protocol", Work
       in Progress.

Authors' Addresses

    J. Mark Pullen
    C3I Center/Computer Science
    Mail Stop 4A5
    George Mason University
    Fairfax, VA 22032

    EMail: mpullen@gmu.edu

    Ravi Malghan
    3141 Fairview Park Drive, Suite 700
    Falls Church VA 22042

    EMail: rmalghan@bacon.gmu.edu

    Lava K. Lavu
    Bay Networks
    600 Technology Park Dr.
    Billerica, MA 01821

    EMail: llavu@bacon.gmu.edu

    Gang Duan
    Oracle Co.
    Redwood Shores, CA 94065

    EMail: gduan@us.oracle.com

    Jiemei Ma
    Newbridge Networks Inc.
    593 Herndon Parkway
    Herndon, VA 20170

    EMail: jma@newbridge.com

    Hoon Nah
    C3I Center
    Mail Stop 4B5
    George Mason University
    Fairfax, VA 22030

    EMail: hnah@bacon.gmu.edu

Full Copyright Statement