# Contents

# Figures

## Introduction

The goal of this project is to model and simulate Internet Protocol multicast (IPmc) network to enable performance analysis using various protocols. Since the primary concern is to study the performance of multicast applications, the IP model which was modeled models the Internet Gateway Management Protocol (IGMP) extensively. Multimedia applications and real time traffic are sensitive to delay compared to unicast applications. The delays involved for a packet are of the following three kinds of decreasing importance in this context.

1. queuing delays at each intermediate routers
2. transmission delay
3. propagation delay

The propagation delay is a delay which cannot be avoided and is fixed. This delay depends on the physical distance and type of link. The transmission link is also fixed which depends on the link speeds even though this can be different for different routers. Queuing delays depend upon the traffic coming at each router, the processing capabilities of the routers and the link capacities. Transmission delay and the propagation delays are negligible compared to queuing delays in most scenarios. This happens when a router gets traffic for processing, more than it can handle for a long period of time. In mathematical terms when the average packet arrival rate is greater than the service rate of the router for a long period of time, the queue in the router builds up and hence a longer waiting time for the packets before being served and routed to the next destination. This causes considerable delay and affects delay sensitive traffic the most.

The solutions for minimizing the delay are

a) controlling the inflow of traffic each sender delivers to the network for transportation.
b) knowing the shortest route to the receiver from the source

ReSerVation Protocol (RSVP) protocol explained in section 3 aims to use to first solution and avoid congestion. QOSPF protocol explained in the section 2 aims to use the second solution to avoid congestion in a network.

In this project IP multicast model is built using OPNET with the QOSPF and RSVP functioning above the IP layer. RFC 1112[1] is used to model of the IP multicast. RFC 1112 deals with definition of the host behavior in a multicast environment. IGMP explained in [3] is used to model the routers.

This work was done in conjunction with Dr. Mark Pullen's project to provide a simulation for the QOSPF routing protocol. Lava K. Lavu provided the RSVP and QOPSF models. The author developed the IGMP model and modified the OPNET IP model and the Medium Access Control (MAC) layer to support multicast. Accordingly this report will focus on the function of the IGMP and the IPmc.....???

## Network architecture

Fig 2.1 illustrates an example of a network used to develop the model. This figure corresponds to the network layer configuration built in OPNET.

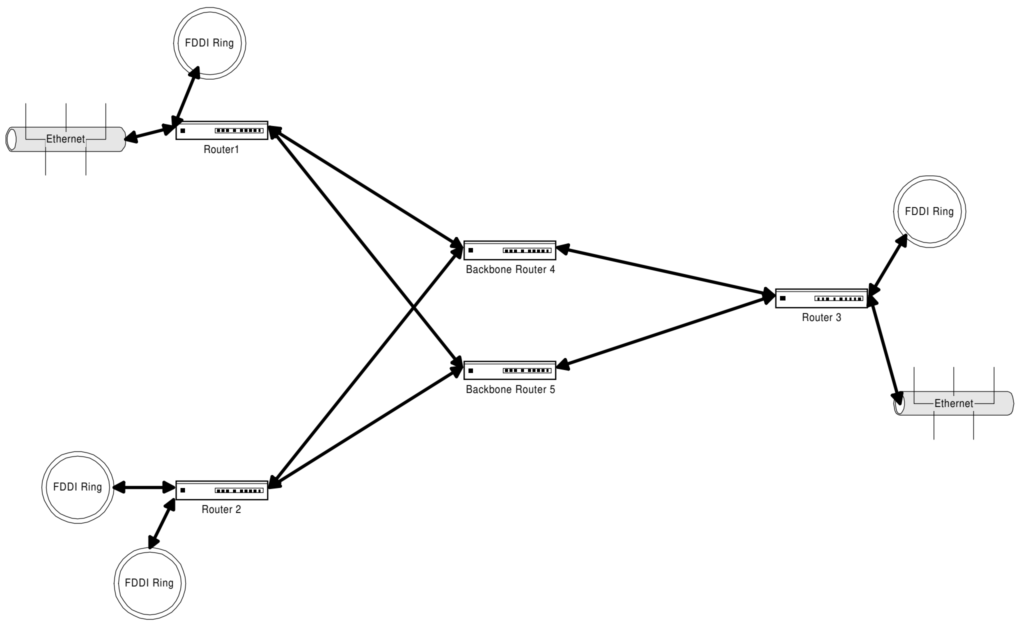


Fig 2.1 Network layer topology

The network consists of two backbone routers namely Backbone Router 4, Backbone Router 5 and three other normal routers . The normal routers are connected to either a Fiber Distributed Data Interface (FDDI) network or to a Ethernet based network.

Figure 2.2 describes a structure of a FDDI ring. It consists of 5 hosts and a router : *Subnet router*. The 5 hosts and the subnet router are interconnected as shown. The subnet router is the router to route the traffic between the hosts in the FDDI and the external networks. The structure of the subnet router is explained in section 2.2. The structure of a host is explained in section 2.3.

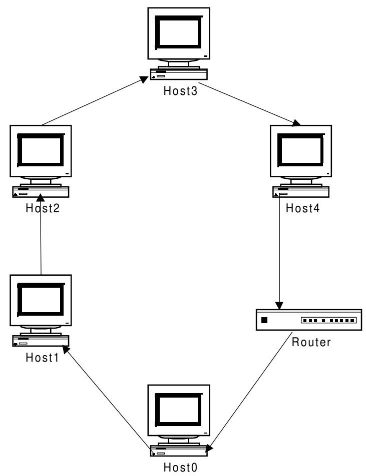


Fig 2.2 An example FDDI ring

Figure 2.3 describes the structure of an Ethernet used in building the model. Here again there are 5 hosts and one subnet router.

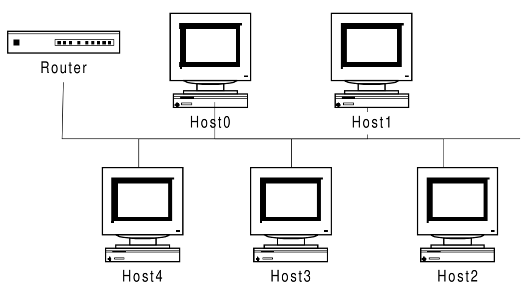


Fig 2.3 An example Ethernet subnet

The following section describes the concept of layering to implement network protocols. The layered architecture of router is described in the Section 2.1.

### Layering

Networking protocols are normally developed in layers, with each layer responsible for a different facet of the communications. A protocol suite such as IGMP/IP or TCP/IP is the combination of different protocols at various layers. The model implemented for this project can be considered to be a 4-layer system as shown in Fig 2.4.

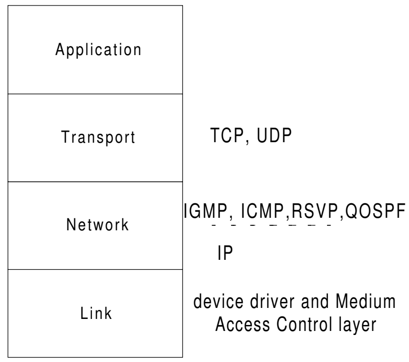


Fig 2.4 Four layer of Internet protocol suite

Each layer has a different responsibility.

1. The *link* layer normally includes the device driver in the operating system and the corresponding network interface card in the computer. Together they handle all the hardware details of physically interfacing with the cable.
2. The *network* layer handles the movement of the packets around the network. This layer handles the routing of packets. The layers IGMP, ICMP, RSVP, QOSPF even though is shown with the IP layer, perform certain functions with the services of the IP layer sometimes inter acting with the application layer. This will become clear in Section 5 when the model built for this project is described.
3. The *transport* layer provides a flow of data between two hosts, for the application layer above. TCP provides a reliable flow of data between hosts. UDP, on the other hand provides a much simpler and unreliable service to the application layer.
4. The *application* layer handles the details of the particular application.

### Independent routers

There are two types of routers in the whole network. All the routers which are used to route traffic between subnet routers or routers are called independent routers. The other type of routers are the ones inside the subnets used to route the traffic between the hosts in a subnet and the independent routers connected to it. The architecture inside an Independent router is illustrated in Fig 2.2.

Both types of routers have the five entities IP (Internet Protocol), TCP (Transmission Control Protocol), IGMP, QOSPF and RSVP.

The IP layer handles communication from one machine to another. It can accept a packet from the upper layer protocols TCP, IGMP, QOSPF, RSVP, or IP entity in neighboring routers and deliver the packet to the corresponding protocol in the destination machine. It accepts a request to send a packet from these layers along with an identification of the destination machine, encapsulates in an IP datagram, and routes the packet using the information in it’s routing table.

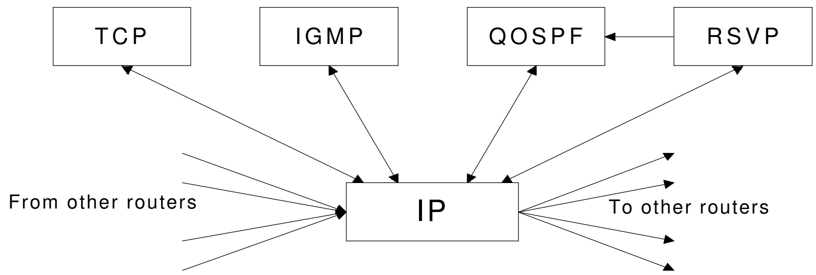


Fig 2.2 Independent router architecture

TCP layer processes the unicast packets, IGMP layer processes the multicast packets, QOSPF layer processes the QOSPF protocol packets and the RSVP processes the RSVP packets. The processing in these layers are explained in detail in Section 6.

### Subnet routers

Fig 2.3 Illustrates the architecture of a router in a subnet. These routers are superset of the independent routers except that they have any only a single connection with the external network. In addition to the entities mentioned for the Independent router, these routers also have a MAC Layer through which it accepts the traffic from the hosts in it's subnet to be routed to other subnets.

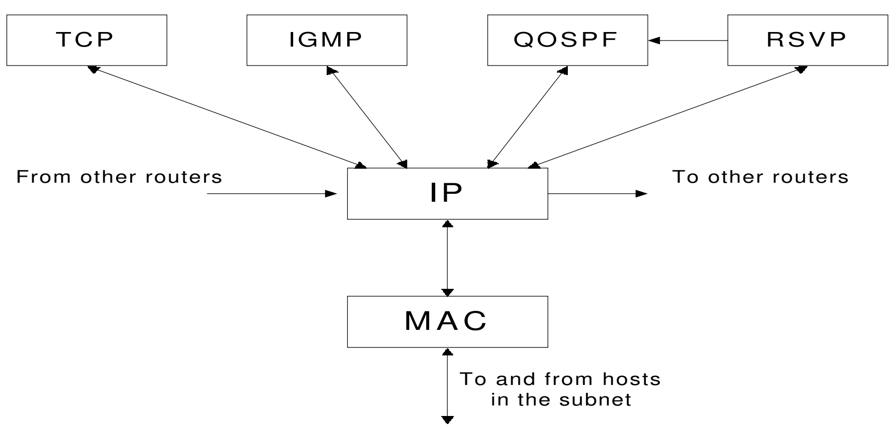


Fig 2.3 Subnet router architecture

The MAC layer in a subnet layer will either be a FDDI MAC or a Ethernet MAC depending upon on type of the underlying network. When a packet is sent from the IP layer to the MAC layer the destination IP address is to be mapped on to a corresponding physical layer address before sending the packet onto to the underlying network which in this project is either a FDDI ring or a Ethernet.

### Hosts

Hosts are the models which generate the multicast and the TCP traffic. TCP traffic is a background traffic used to model the unicast traffic between hosts encountered in the internet, in addition to multicast traffic we are modelling. Fig 2.4 illustrates the layered model of a host modeled in this project.

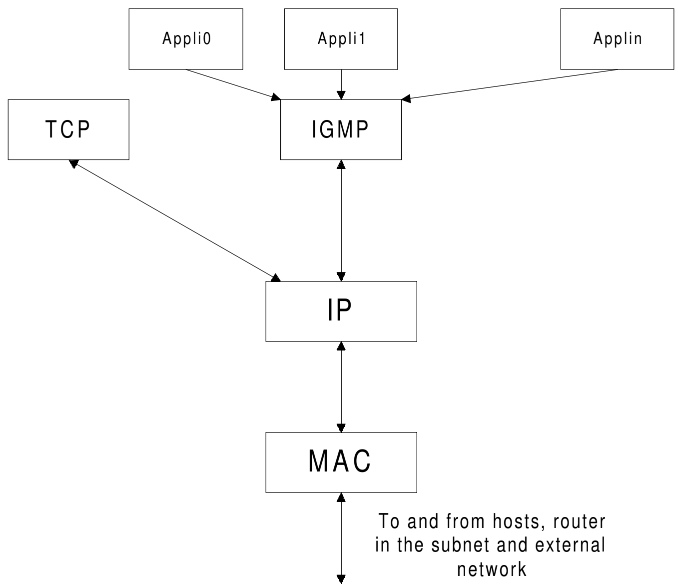


Fig 2.4 Host architecture

The application layers called 'Appli*n*' generate a request to become a member of a particular group. The requests are sent to the IGMP layer in a host which are different from the IGMP layers in the routers in their functions. Once the application layer has sent the message to join a group, it starts transmitting data packets for that group if it has any.

The next layers in the hierarchy are TCP and IGMP. The IGMP is defined in [2]. It accepts requests from the application layers and maps them onto a IGMP message called "response". This message is delivered to the IP layer for routing. On the other side the IGMP layer accepts packets from the network for applications above it or messages from the nearest router through the IP layer beneath it. The messages from IP layer are directed to the appropriate application if they are data packets. If they are IGMP messages appropriate tables are updated.

TCP layer is used only to generate a back ground traffic to emulate the internet. TCP module generates a TCP connection request at certain intervals and starts sending data for that connection, if it gets a response. The TCP module establishes connection at random intervals for a non-deterministic amount of time with a random destination.

The next layer in the hierarchy is the IP (Internet Protocol) layer. IP layer is concerned with maintaining the routing tables, routing the packets coming to it. For this project since the hosts are connected only to a single network the route used is always a default route towards the nearest router or to a host in the same subnet. When IP layer gets a packet from the network, if the packet's destination address matches its own address it sends the packet to either TCP or IGMP depending upon the type of packet. When the IP layer receives the packet from the upper layer protocols, it looks up the routing table and forwards the packet to the lower layer protocol with the next node information.

In addition to routing, the IP layer also takes care of fragmentation based upon the value of the Maximum Transmission Unit (MTU) of the underlying physical link. More detailed description of the functions of the IP layer are described in Sec 6.

## QOSPF

QOSPF is an extension of OSPF[11] and MOSPF[12]that is used to provide Quality of Service (QoS) routing. Advertisements indicating the resources available and the resources used are advertised to the routing domain and paths are computed based on topology information, link resource information, and the quality required for the flow in consideration.

QOSPF can work in conjunction with a QoS signaling protocol such as RSVP to provide QoS routing. RSVP which is discussed in next section allows the instantiation of network state to provide a specific service level to a data stream. QOSPF is a layer which works above the IP layer in the OSI protocol structure. QOSPF is active only in the routers and most of its interaction is with IP layer. When QOSPF works in conjunction with RSVP above the IP layer, a route calculation action in QOSPF is triggered by a request for a route from the RSVP deamon.

When QOSPF deamon has done the calculation of the optimal path, it sets the appropriate value in the routing table of the IP layer which will affect the path the corresponding data stream will take. Thus the routing table in IP layer for the data streams which use RSVP follow the optimal path.

The difference in QOSPF compared to other route calculation protocols is that, in addition to using the shortest path for calculating the route to a destination, the QOSPF deamon also uses the information about the available resources along the route. This method makes it unique in finding out the most optimal route for a particular data stream. This difference is explained in the following figure.

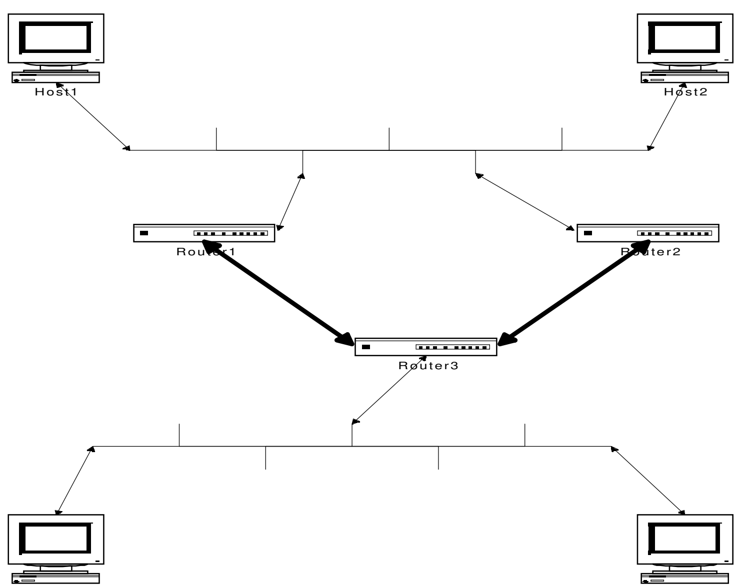


Fig 2.1 Network topology

Suppose host H1 is sending data to host H3 at rate R. Routing protocols other than QOSPF will give the shortest path H1 -> R1 -> R3 -> H3. But the processing capabilities of R1 may be limited compared to R2 and so the data stream will experience higher delays compared to the route H1 -> R2 -> R3 -> H3. This problem is encountered due to lack of taking into account the processing capabilities of an intermediate router i.e. in other words the routing protocol always uses the shortest path.

QOSPF uses both topology information and resource information to calculate "QoS routes" that can provide the resources needed for a flow even though the route may not strictly be the shortest path.

## Protocol description

QOSPF protocol involves a number QOSPF messages exchanged between the QOSPF deamons in each router. Routers advertise network resource information and topology information. A route for a specific data flow is calculated based on topology, network resource information, and QoS requirements for that flow. The QoS requirements for a specific flow is obtained from *resv* message which is sent by an application requiring a specific QoS which is defined in the RSVP[3].

The network resource information includes available link resources on a router as well as existing link resource reservations on the router. The resource information is advertised in Link Resource Advertisement (RES-LSAs) and Resource Reservation Advertisements (RRAs).

Input to the QOSPF Dijkstra calculation includes the source and destinations address and the QoS requirements for the flow. The QOSPF Dijkstra calculation for an area is performed by processing the different LSAs.

## Internet Group Management Protocol (IGMP)

IGMP is a protocol used by multicasting hosts and routers. Multicasting provides the service of delivering information to multiple destinations. Some of the applications which use multicasting are interactive conferencing, dissemination of mail or news to multiple recipients.

### Protocol description

IGMP lets all the systems on a physical network know which hosts currently belong to which multicast groups. This information is required by all the multicast routers, so they know which multicast datagrams to forward onto which interfaces. IGMP is defined in [2].

IGMP messages are transmitted in IP datagrams. IGMP message has a fixed-size message with no optional data. Fig 4.1 shows the format of an IGMP message.

| IGMP ver | IGMP type | Checksum |
|---|---|---|
| 32 bit group address | | |

Fig 4.1 IGMP message format

An IGMP 'type' of 1 is a query sent by a multicast router and 2 is response sent by a host. In a query the group address is set to 0 and in a report it contains the group address being reported. IGMP uses class D addressing. The first 4 bits is always 1110 and the next 28 bits stand for the multicast group ID.

The concept of multicasting is described below.
A multicast group is identified by an unique group number ‘N’. A multicast group consists of one or more hosts. A process in a host joins a group by sending a join packet. The process can then stay in the group as long as it wants. When a process is a member of a group it can receive or transmit destined for that group. A part of the destination address of the packet will be the unique multicast group number. So a multicast packet sent out by the host who is a member of a particular group will be received by all the members registered for that group.

Routers connected to such hosts generating multicast traffic must be capable implementing the IGMP protocol. Multicast routers maintain a table having information about all the groups which the routers’s interfaces are members of. Fig 4.2 illustrates the concept of multicasting where an Ethernet subnet and a FDDI ring are interconnected by a router.

I1 and I2 stand for interfaces of the router. I1 connected to the Ethernet and I2 connected to FDDI ring. Two hosts on the Ethernet are members of group A, and one host member of group B. On the FDDI ring there are two hosts both members of group B.

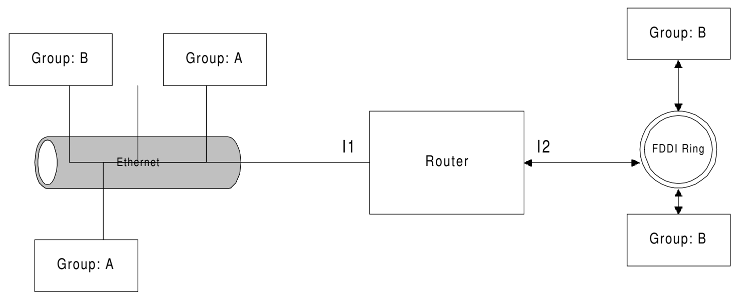


Fig 4.2 Illustration of multicasting

The router maintains a multicast routing table similar to one shown below.

| Group | Interface |
|---|---|
| A | I1 |
| B | I1, I2 |

The table indicates to the router that interface I1 is a member of group A and group B. Interface I2 is a member of group B. If the router gets a packet destined for group A on interface I1, router does not forward the packet since I2 is not a member of group A (i.e. No hosts on the FDDI ring is a member of group A). If the router receives a packet destined for group B on either of the interfaces, it will forward it on the other interface since there are hosts who are members of group B on both the subnets.

The IGMP layer at the routers sent IGMP packets called queries at intermediate intervals on all the interfaces the router is connected to. The router is asking all the hosts connected directly to the router if they are members of any multicast group. If a host is a member of a group N, the host sends a *join* packet (which is defined as *response* in [1]) for group N. The host sends a join packet for all the groups the host is a member of.

All the IGMP packets (*join* and *query* packets) have a "time to live" (ttl) value in the IP header set to 1. This restrains the IGMP packets not to be forwarded to external networks.

When a host receives a query from the nearest router it schedules to send the response which is a *join* packet after a random interval of time. When a host H1 sends out a *join* packet for a group N either when it is joining the group the first time, or as a response to the query sent out from the router, this *join* packet is received the router and by all the hosts who are members of group N in the subnet which the host H1 belongs to. So if another host H2 (in the same subnet)a member of group N has scheduled to send a *join* packet, receives the *join* packet sent by host H1, H2 does not send a *join* packet since a *join* packet has already been sent on that interface for the group N.

When a host wants to leave a group it stops responding to the queries sent by router to which it is connected. Eventually the router will cancel the membership since it does not receive an responses from any host on that interface. This is how the group membership maintained at a router.

## OPNET Model Description

This chapter describes the OPNET model built for this project. Each section in this chapter describes in detail each layer in the host and the router with the help of the corresponding OPNET network layer or node layer or the process layer starting from highest layer in the hierarchy.

*Address format*: The model built using the OPNET has only one type of addressing denoted by "X.Y" where X is 24 bits long and Y is 8 bits long. The X indicates the destination or the source network number and Y indicates the destination or the source node number. In this project X = 500 is reserved for multicast traffic. For multicast traffic the value of Y indicates the group to which the packet belongs to.

### Network Layer

Fig 6.1 describes the network topology built using the OPNET network editor. This network topology is the same as the example discussed in Chapter 2. It consists of two backbone routers BR2, BR4, three area border routers BR0, BR1, BR3 and six subnets F0, F1, F2, F3, F4, and F5. The links interconnecting each of the different elements of the network are pairs of simplex links which enables bi-directional traffic.

Fig 6.1 Network layer elements

The attributes of the elements of the network layer are:

*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)

*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.

*Simplex links*

1. Propagation delay in the links
2. the process model to be used for simulating the simplex links (with or without animation).

Fig 6.2 shows the FDDI ring in a subnet and a Ethernet in a subnet. Both the subnets will have one router and one or more hosts. The routers in the subnet are included to route the traffic between the FDDI ring or Ethernet in the corresponding subnet and the external network. Router is connected on one end to Ethernet or FDDI ring and generally connected to an area border router on another interface (ie. Routers are multihomed). In the Ethernet all the hosts are connected to the bus and in FDDI ring the hosts are interconnected as illustrated in the figure 6.2.

Fig 6.2 Example FDDI ring and Ethernet

The 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 control protocol (MAC) which has been standardized by the IEEE under the name 802.3. The role of this MAC protocol is to provide efficient and fair sharing of the communication channel connecting the stations of the LAN.

## Node layer

This section discusses the internal structure of the hosts and routers with the help of node level illustrations built using the Node editor of the OPNET. The basic elements of a node level illustration are

1. Processor nodes: Processor nodes are basically used for processing incoming packets and generating packets with a specified packet format.
2. Queue node: Queue nodes are a super set of processor nodes. In addition to the capabilities of processor nodes, queue nodes also have capability to store packets in one or more queues.
3. Transmitter and Receiver nodes: Transmitters simulate the transmission effect of packet transmission and Receivers simulate the receiving effects of packet reception. The transmission speed is an attribute of the transmitter and receiving speed is an attribute of the receiver. These values together decide the transmission delay of a packet.
4. Packet streams: Packet streams are used to interconnect the above described nodes.
5. Statistic streams: Statistic streams are used to convey information between the different nodes Processor, Queue, Transmitters and Receivers nodes respectively.

### FDDI Host description

The host model built using OPNET is very similar to the TCP/IP protocol suite. The node level figure of a host or router illustrates the layered structure implementation which is similar to the TCP/IP protocol explained in Section 2.1. Fig 6.3 shows the node level structure of a FDDI host.

#### MAC queue node

The process model for the FDDI MAC is discussed in Section 5.3.1. 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, it 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.

Fig 6.3 Node level architecture of a FDDI host

A detailed description of the protocol implemented is described when describing the State Transition Diagram (STD) of process model *fddi_mac* in Section 5.3.1.

* the term frame is used in the context of flow of information at the physical layer.

** the term packet is used in context of flow of information at and above the IP layer.

**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.

**IP queue node**

The next layer in the hierarchy is the IP layer. 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. In this layer packets incur delay due to limited throughput capability. Packets experience delay when the average number of packet arrivals to the IP layer exceeds the average service rate of the IP layer. The packets arriving at the IP layer are queued and operates 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. The process model of the IP layer "ip_rte" is discussed in Section 5.3.2. The layers above the IP are the transport layer protocol UDP and muticast protocol IGMP_host*[1]. The functions of the IGMP_host layer at the host and router are different and hence use different process models. The functions of the IGMP_host and UDP node are discussed below.

*[1] IGMP_host denotes the IGMP model at the host and IGMP denotes the IGMP model at a router.

**IGMP processor node**

IGMP_host node: IGMP_host node is a process node. Packets are not queued in this layer. IGMP_host provides services for the multicast applications utilizing the services provided by the IP layer. The IGMP_host maintains a single table which consists of group membership information of the application above the IGMP layer. The functions performed by the IGMP_host layer depends upon the type of the packet recieved and the source of the packet.

The IGMP_host layer expects certain type of packets from the application layer and from the network. The following are the set of four packets the IGMP_host expects and the functions performed.

**1.** Accept *join group* requests from the application layer (it 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 of this group (ie. 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* which 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 statistic wire connected from the IGMP_host to the MAC layer so that the MAC layer begins to accept the frames destined for the group *N*.

**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, if any of the 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 Fig 6.4 and the description in the following section describes the scenario.

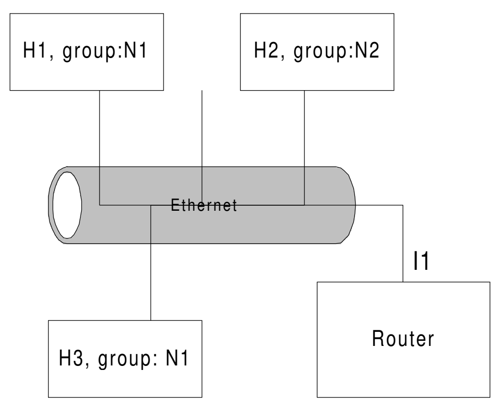


Fig 6.4 An example Ethernet architecture

The router *R* interfaces with the Ethernet on one interface *I1* and *H1*, *H2*, *H3* are the hosts on the Ethernet. Also let us assume that the 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 muticast group address. So all the member hosts of a group will receive the responses sent by the other hosts in the Ethernet 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. This is the reason why the events to generate response to queries are scheduled randomly. Also the interval for scheduling the above described event has 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. The Ethernet example in Fig 6.4 described the function to be performed when a IGMP_host receives the response sent by other hosts in the subnet.

**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 from now on will not send any response corresponding to this group (unless any other 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, the membership is canceled for that group on that interface.

#### UDP processor node

This node is used model a best effort traffic in the Internet using *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 QOSPF or RSVP. UDP node aims to create the scenarios observed in a network which has one type of application using the services provided by QOSPF, RSVP to acheive specific levels of QoS and the best effort traffic which uses the services provided by only the underlying IP.

The UDP node generates traffic to a randomly generated IP address so as to create congestion in the network. 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 UDP node are

a) Session Inter Arrival Time (IAT) : is the variable used to schedule an event to begin a session. The UDP node generates a exponentially distributed random variable with mean Session IAT and begins to generate data traffic at that time.

b) Data IAT : When the UDP generates data traffic, the inter arrival times between data packets is Data IAT. A decrease in the value of Data IAT increases the severity of congestion in the network.
c) Session-min and Session-max : When the UDP node starts generating data traffic it remains in that session for a period which is uniformly distributed between Session-min and Session-max.

#### 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.

The attributes of the application layer are

a) Session IAT: is the variable used to schedule an event to begin a session. The Application node generates a 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.
b) 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.
c) 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 Sess-max. A uniformly distributed random variable between Session-min and Session-max is generated for this purpose.
d) NGRP: 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.

### Router description

There are two types of routers. A router in 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.

The queueing technique implemented in the router is a combination of input and output queueing. The nodes rx1, rx2, rx3 and rx4 are the receivers connected to incoming T1 links. The router in Fig 6.5 has a physical interface to the FDDI ring, which consists of the queue node *mac*, transmitter *phy_tx*, and the receiver *phy_rx*. The backbone routers will not have the *mac* layer. The services provided and the functions of the *mac* layer is same as in a host discussed in Section 5.2.1.1.

Fig 6.5 shows the node level model of a subnet router.

Fig 6.5 Node level architecture of a subnet router

There is one major difference between the *mac* node in an subnet router and one in an host. The *mac* node in a subnet router accepts all the multicast packets unlike the *mac* in a host which accepts only the multicast packets, which the IGMP_host is a member of. This is the reason why the statistic wire from the IGMP to *mac* layer does not exist in a router.

The next layer in the hierarchy is the addr_trans processor node which provides the service of translating the IP address to physical address. The *addr_trans* node was discussed in Section 5.2.1.2.

**IP layer**

The next layer in the hierarchy is the 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 two routing tables are

a) *Unicast routing table*: This table is a single dimensional array, which is used to route packets generated by the UDP process node in the hosts. If no route is available to a particular IP address, the corresponding entry is set to a default route in the array.

b) *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 a upper layer routing protocol which for this project is the QOSPF layer. When the IP layer receives a multicast packet with a session_id corresponding to a session which is utilizing the QOSPF and RSVP, it looks up the multicast routing table to obtain the next hop.

c) *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 QOSPF (can be any routing protocol) uses this information in the group membership table to calculate and set the routes in the *Multicast routing table*.

The IP layer has K number of sub queues built in it, which implements queuing based upon the priority of arriving packets from the neighboring routers or the underlying subnet. The architecture of the router queues is illustrated in Fig 6.6.

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. A non-preemptive priority is used in servicing the packets. After the servicing, packets are sent to the one of the output queues q_1, q_2, q_3, ... q_I or the mac. The packets are queued in these queues until the transmitter becomes available for transmission.

Fig 6.6 Queue architecture in a router

Attributes of the IP node are:

a) Unique IP address for each interface (a set of transmitter and receiver constitute an interface).
b) Service rate: the rate with which packets are serviced at the router.

c) Queue size: size of each of the sub queues used to store incoming packets based on the priority can be specified individually.

#### Output queues

The output queues perform the function of queuing the packets served by the IP layer when the transmitter is busy. Significant amount of queuing takes place in the output queues only if the throughput of the IP node is very much greater than the transmission capacity of the links.

Attribute of the queue node:

a) Queue size: size of the queue in each queue nodes.

#### Transport and routing layer protocols

The next layers in the hierarchy level are the IGMP for implementing the multicasting, QOSPF for implementing the routing protocol, and RSVP for providing specific QoS.

IGMP Node

The IGMP node implements the IGMP protocol as defined in RFC 1112. The IGMP node at a router is different from the one at a host. The functions of the IGMP node at a router is described in detail in the following section. Section 5.3.4 describes in the detail the implementation.

a) IGMP node at a router sends queries at regular intervals on all its interfaces.
b) When IGMP receives a response to the queries sent, IGMP updates the multicast Group membership table (Section 5.2.2.1) in the IP node.
c) 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 any group on any interface, indicating that a interface is no longer a member of a specific group. This update is done only on entries which indicate an active membership for a group on a interface and the router has not received a *response* for the last query sent. Routing protocol (which is QOSPF in this project) uses the information in the Group membership table to calculate the routes and update the *Multicast routing table*.
d) 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.

### Process models

This section describes the process models specified by a State Transition Diagram (STD) for all the nodes discussed in the previous section. The previous section 5.2 described the different functions performed by all the nodes used to form the network. This section describes the implementation details.

The following sections discuss the implementation of each process model which executes the functions described in the previous sections. The aim here is only to provide an overview of the source code.

### FDDI MAC process model : *fddi_mac*

The fddi_mac process must execute within a queue module and be connected to a transmitter, receiver and to the addr_trans processor. fddi_mac uses a single sub-queue of the queue module in which it is located to hold frames that are waiting to become eligible for transmission. Fig 6.7 shows the process model STD of the fddi_mac.

Fig 6.7 FDDI mac STD

<u>General Operation:</u>

*CLAIM:*

This is the initial state of the process model and is entered upon before the simulation starts. Its primary purpose to emulate the negotiation that takes place in an FDDI ring with regard to the value of Target Token Rotation Time (TTRT). The minimum of the TTRT requested by all the stations in a ring takes into effect. This is used to regulate the use of bandwidth in the ring. After exiting the CLAIM state, the fddi_mac process model always transfers control to the INIT state upon receiving its next interrupt.

*INIT:*

Most of the actions to be performed in this state have to do with assignment of values to state and global variables. These include the station latency, inter-station propagation delay, the combined inter-station delay and priority table. The process also defines a timer object in the header block and a series of procedures for manipulating and querying timers. In INIT, fddi_mac also acquires knowledge of its own station address (physical address) and places this value in state variable. The fddi_mac also registers itself in a global table that maps station addresses to process object ID's. This table is used by an acceleration

mechanism that bypasses idle periods to avoid letting the token freely circulate and generate a large number of events. The state variable, which represents the synchronous bandwidth usable on each token capture is also initialized here.
The final statements of the INIT have to do with the initial generation of a token for the ring. The state variable which will keep track of the total amount of bandwidth scheduled for transmission since a token was captured was, is also initialized here.

*ISSUE_TK*:
This states primary action is to forward a packet that represents the token with the proper delay. In addition ISSUE_TK checks for the special condition whereby the station is releasing the token without having performed any frame transmissions, and the station has no data to transmit. If any station has data to transmit the token will be forwarded normally, otherwise the token will blocked until a station registers its intent to use it.

*IDLE:*
This state is the state the process model stays when there are no events corresponding to the fddi_mac. A transition from this state can take place due to four conditions.

1. arrival of a packet (service data units(SDU)) from the upper layers (which is IP in this model)
2. arrival of a frame or a token from the physical layer
3. self interrupts representing expiration of timers
4. remote interrupts requesting that a token be generated and inserted into the ring.

*TRT_EXP:*
This state is entered when fddi_mac receives a self interrupt due expiration of certain timer. After setting certain variables the control returns to IDLE state.

*ENCAP:*
This state is entered when a SDU arrives from the IP layer. The actions performed here, acquires the arriving SDU and associated control information to use these to create a MAC frame suitable for transmission to a peer MAC entity via the transmitter. The resulting MAC frame is enqueued until a later time when it becomes eligible for transmission in the TX_DATA state.

*RCV_TK:*
This state is entered by fddi_mac upon receipt of a token from the physical layer. Upon reciept of the token this state tests for certain conditions in order to determine if the token is usable. Based upon the results of the tests if the token is usable control is transferred to the TX_DATA state otherwise the token is forwarded to the next station and control transferred to the IDLE state.

*TX_DATA:*
This state is entered by the fddi_mac when the token is captured and a determination is made that it is usable. The role of the TX_DATA state is to dequeue and send frames into the ring until the token is no longer usable by n, at which time it is forwarded down-stream.

*FR_RCV:*
This state is entered when a information frame is received from the physical frame. The action performed in this state is to extract the source address of the packet so that a determination can be made with regard to stripping the frame from the ring. The control then transfers either to the FR_REPEAT or the FR_STRIP state.

*FR_REPEAT:*
In this state, a frame has been received that was originated by a station other than this one. If the IGMP is a member of the group address this frame belongs to, a copy of it is forwarded to the IP layer. The original frame is always transmitted back onto the ring until it reaches the source from which the frame originated.

*FR_STRIP:*
This state is entered when a frame is no longer to be put on the ring. The frame is destroyed and control transferred to the IDLE state.

### ADDR_TRANS process model :*addr_trans*

Fig 6.7 shows the STD of the addr_trans process model. The addr_trans process model has only one input packet stream from the IP layer and one outgoing packet stream to the *mac* layer. The function of this process model is to provide address mapping from the IP address to physical address. This *addr_trans* performs this function by looking up a file which is an simulation level attribute.

Fig 6.8 addr_trans process model STD

General Operation: The addr_trans process model performs initializations in the INIT state. This involves the loading of the address translation table from the file specified when the simulation is run. Control then passes directly to the WAIT state and waits for an interrupt. The only interrupt that is possible for this process model is the packet arrival from the IP layer. In addition, each arriving packet is expected to be accompanied by an ICI of the format *ip_arp_req*. The ICI indicates the node number of the next node to be reached on the local physical network. In the ARRIVAL state, this address in the ICI is used as a key into the address mapping table by calling the externally defined procedure *arp_sup_phys_addr_get()*. This procedure returns the physical address of the node of interest and the packet is forwarded to the Data Link Layer. With an ICI of format *ip_mac_req* specifying the next node physical address. If no match is found, the packet is discarded.

Address Mapping Table Definition: Address mapping tables are used to specify the association between IP addresses and physical network addresses so that routing and packet filtering can be performed properly within arbitrary physical networks. Each *addr_trans* process is assigned an address mapping file before the simulation begins which the process model uses to build the address mapping table.
The address mapping file is a text file and consists of as many number of lines as the number of physical addresses possible in a subnet. The first entry in each line is the IP node number and the second entry the corresponding physical address. A sample address mapping file used for this project is shown below.

| #IP Node number | Physical address |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 501 | 1001 |
| 502 | 1002 |
| 510 | 1010 |

Fig 6.9 A sample address mapping file

Lines preceded by the pound sign are comment lines. In the above example file the IP addresses 501, 502 and 510 are multicast addresses and correspond to physical addresses 1001, 1002, 1010.

### Internet Protocol (IP) layer: *ip_gwy*

The IP is a connectionless network level protocol involved in interconnecting multiple networks of different types. The two types of networks modeled in this project are FDDI and Ethernet.

The packets that are created and forwarded by IP modules are called *datagrams*. IP datagrams carry headers that hold control information such as source and destination addresses, priority levels, protocol identifiers, fragment identifiers and so on. IP also has the task of performing reassembly of the fragments before it can deliver data to the higher level protocols.

Fig 6.10 shows the STD process model of the IP node (*ip_gwy*: stands for the IP at the gateway). The *ip_gwy* can have one or more direct connections to the physical layer where the link layer model does not require address mapping or through the *addr_trans* and the *MAC* layer or to one or more upper layer protocols. The main task of the *ip_gwy* is to enque datagrams based on the priorities, serve them at a specified rate and route them to the proper destination.

Fig 6.10 STD of the IP node : ip_gwy

<u>General Operation:</u> This section describes the control flow in the STD.

INIT: The process performs initialization in this state. Before the simulation starts the attributes are obtained in this state. The process model obtains the number of active interfaces, IP address for each set of active interfaces, service rate, initializes the Multicast Group membership table, Multicast routing table and Unicast routing table. After the above actions control transfers to the *arrival* state if there is a datagram arrival or else to the *idle* state.

*IDLE:*

The process model remains in this state when there is no packet arrival or when a datagram is being served by the IP layer. Two types of interrupts can transfer the state from *idle.* A) packet arrival b)service completion. These two interrupts are called **ARRIVAL** and **SVC_COMPLETION** which are defined in the header block of the process model. When there is a packet arrival the control transfers to the *arrival* state and to the *svc_completion* state when a IP completes servicing a datagram.

*ARRIVAL:*

This is the state control transfers to whenever there is packet arrival to the IP layer. The actions performed in this state are

1. number of arrivals to the IP layer variable is updated to calculate the throughput.
2. The priority of the datagram is obtained from the information in the IP header.
3. An attempt is made to queue the datagram at the tail of the $n^{th}$ sub queue, where the value of *n* indicates the priority of the datagram. The function *get_prio* ( ) obtains the priority of the datagram. g*et_prio()* is defined in the function block of the process model. If the queue is full the datagram is destroyed. If the datagram insertion is successful and no datagram is being

served at that instant the control transfers to SVC_START state otherwise control transfers back to IDLE state.

*SVC_START*:
Control can be transferred to this state from either ARRIVAL state or the SVC_COMPL state. In this state the size of the datagram at the head of the queue with the highest priority obtained and the corresponding service time is calculated. An interrupt is scheduled for this time to indicate service completion of a datagram. Control transfers to the IDLE state.

*SVC_COMPL:*
When a datagram completes service it undergoes its final handling in the SVC_COMPL state. Control is transferred to this state due to a **SVC_COMPLETION** interrupt indicating of completion of service. In the SVC_COMPL state, handling is dependent upon the destination address of the datagram and/or the protocol ID in the header. The possible values and the corresponding actions performed are discussed below. Before routing the packet, in this state the information protocol value, destination address, source address, time to live counter value, and fragment information are collected. If the packet is destined for the upper layer protocols the packet is forwarded to the appropriate node. In case of packets with a group address, copies of packets are made based on the information in the Multicast Routing table and forwarded to the next nodes. In case of unicast packets the next node is obtained from the Unicast Routing table. If no route exists the packet is routed to the default route or destroyed depending upon the type of the router.

If the maximum transmission unit of the underlying physical layer is less than the size of the packet, the packet is fragmented, appropriate fragment id in the IP header set and forwarded. If the queue is empty, control is transferred to the SVC_START state for servicing of the next packet. If no packets exist in any of the sub-queues, control is transferred to the IDLE state.

## IGMP_host

Fig 6.11 shows the process model STD of the IGMP_host. The IGMP_host will execute from within a process module since queuing is not done at this layer.
The main functions of IGMP_host involve in maintaining group membership information of the application layers above the IGMP_host and responding to queries sent by the nearest router.

General Operation:

*INIT:*
The initializations are performed in this state. The attributes obtained in this state are IP address of the host in which the IGMP_host resides, initializes the multicast group membership table, loads the appropriate distribution functions. After the initializations are done, control is transferred to the IDLE state.

*IDLE:*
This is the state where the STD remains when there are no events for the current simulation time. Control can be transferred from the IDLE state due to either a self interrupt or a packet arrival from the IP layer or a packet arrival from the application layer. When there is a packet arrival from any layer, control is transferred to the SPLIT state. If the interrupt is a self interrupt control is transferred to the RESP state. The self interrupt is an interrupt scheduled for generating responses to queries received from the nearest router. This is the only possible self interrupt.

*SPLI:*
Control is transferred to this state when there is a packet arrival. The lines in this state determine whether the packet came from IP or from upper application layer. If the packet came from the IP layer control is transferred to IP_PACK state and to the APP_INF if the arrival was from the application layer.

Fig 6.11 IGMP_host process model STD

*IP_PACK:*
Transition to IP_PACK takes place when there is a packet arrival from the network. There can be two types of packets arriving from the network.

1. Data packet from the network with the destination address set to some group address. The lines in this state obtain the destination address from the packet, lookup the multicast membership table to obtain the application nodes which are members of this group and forward the data part after decapsulating the packet to all the members.
2. IGMP packets which will either be quereies sent from the router or responses sent by other hosts on the subnet. It should also be noted that IGMP packets are never forwarded since they have their ttl value set to 1.

If the packet is a query, the IGMP_host looks up the multicast group membership table for any active groups. It schedules an event for each active group for sending a response after random intervals of time. If the packet received is a response sent by another host, the IGMP_host checks for an event it had scheduled corresponding to that group and cancels the event.

After completion of this state transfer is controlled to IDLE state where it will wait for the next interrupt.

*APP_INF:*
Transition to APP_INF takes place when there is a packet arrival from the application layer. There can be three types of packets arriving from the application layer.

1. Data SDU for a specific group. The IGMP_host encapsulates the SDU into a IP packet, sets the destination address and forwards it to the IP layer for routing it through the network.
2. Join group packet: This packet is a indication to the IGMP_host that the a application node intends to join a group. The IGMP_host updates the multicast group membership table and sends a response to the router with destination address set to the group address the application intends to join.
3. Terminate group packet: This packet is an indication to the IGMP_host that the application intends to leave the group which had joined earlier. The IGMP_host updates the group membership table and stops sending responses to queries for that group.

Whenever the IGMP_host gets a request to leave or join a group it also has to pass the information to the MAC layer through the statistic wire by writing onto a outstat.

The control is transferred to the IDLE state.

*RESP:*
The process model enters this state after a self interrupt. The function of this state is to send a response for the groups which are active and no other host has sent a response for that group. The lines in the state create a response packet, set the appropriate values in the packet and forward it to the IP layer to be sent to the router.
After all the responses are sent, control is transferred back to the IDLE state.

## Conclusions

The IPmc model built is currently being used to simulate the QOSPF routing protocol by Lava K Lavu. IGMP functions were implemented very extensively and found to be performing as defined in [1]. The IPmc model built was tested with multicast applications joining and leaving random groups, generating traffic at different rates and various distributions. The queue sizes at the routers were set to reasonable values such that packets were not lost at the routers due to queue size limitation. This was done so as to verify the correctness of the model operation and the functions implemented. In addition the OPNET animation capabilities helped to verify the correctness of the functions by visually providing the action of hosts joining and leaving multicast sessions in the network.

## References


1. Deering, "Host Requirements for IP Multicasting", RFC 1112, August 1989.
2. MIL3 Inc., "OPNET: Optimized Network Engineering Tools, Simulation Kernel Manual", November 1991.
3. Braden et. al., "Resource Reservation Protocol Version 1 Functional Specification", November 1996
4. W. Richard Stevens, “TCP/IP Illustrated Volume 1”.
5. Mischa Scwartz, “Telecommunication Networks: Protocols, Modeling and Analysis”.
6. Averril M. Law, W. David Kelton, “Simulation Modeling and Analysis”.
7. William Stallings, “Data and Computer Communications”.
8. Donald Gross, Carl M. Harris, “Fundamentals of Queuing theory”.
9. Donald E. Comer, “Internetworking with TCP/IP (Volume 1), Principles, Protocols, and Architecture.
10. Z. Zhang, “Quality of Service Extensions to OSPF”, June 1996
11. J. Moy, “OSPF Version 2”, RFC 1583
12. J. Moy, “Multicast Extensions to OSPF”, RFC 1584, March 1994.

## Source code

The following section includes the source code written for modeling the IPmc. This includes only the source codes which were completely written by the author. The fddi_mac source code includes only changes made to the process model provided by MIL3 Inc. in the standard model directories to incorporate multicast functions at the MAC layer.