Chapter 4 System Design

In the previous chapter, an analysis of IGMP and PIM-DM has been described. In this chapter, design aspects of PIM-DM and IGMP are discussed. Here, both protocols are concerned in terms of system architecture and object classes design so that both protocols could be implemented in UMJaNetSim. The first section of this chapter focuses on IGMP while the second section concentrates on PIM-DM. In addition, the design of simulation applications for IP Multicast simulation is also discussed. The final section is the summary for this chapter.

4.1 Overall System Design

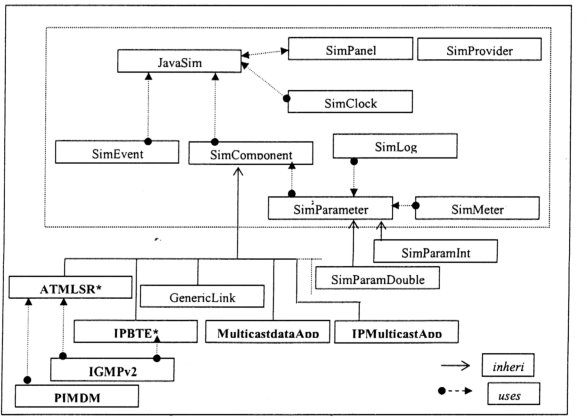

Figure 4.1 New UMJaNetSim Objects with IP Multicast

To provide IP Multicast through PIM-DM simulation in UMJanetSim, IGMP module and PIM-DM module are added. Hence, *IGMPv2* class and *PIMDM* class are developed and adapted into the existing system architecture of UMJaNetSim. Since IGMP is implemented in hosts and routers, *IPBTE* class and *ATMLSR* class must include *IGMP* object. However, PIMDM class is only added in *ATMLSR* class. Figure 4.1 shows the UMJaNetSim objects for PIM-DM simulation. *IPBTE* class and *ATMLSR* class are adapted with a new function. Discussion on both classes could be found in the following sections. In addition, two application classes for multicast simulation are also designed. There are *MulticastdataApp* class and *IPMulticastApp* class. Details of the design for each new class are presented in the following sections.

4.2 IGMP Design

4.2.1 System Architecture Design

Based on the analysis on IGMP, a core that supports IGMP Group Table and IGMP Timers is required. The core consists of four main modules, namely Input, Output, Join and Leave. These modules are used to interact with other components in a host or router.

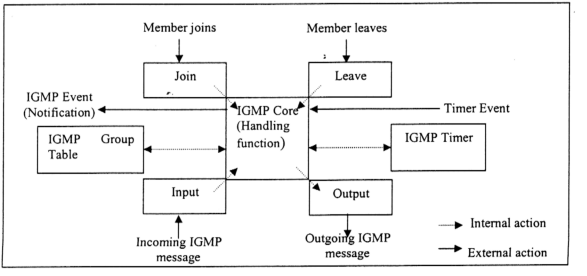

Figure 4.2 Basic System Architecture for IGMP

Although the IGMP implemented in routers and hosts are slightly different, it is possible to design one IGMP that can be implemented in both routers and hosts. The basic system architecture for the IGMP is shown in Figure 4.2.

4.2.1.1 IGMP Core Design

The IGMP core supports several functions. First, the IGMP Core must determine whether it is running in a host or router. Different actions would be taken for different implementation.

The IGMP Core is the main control center for four main modules. Input, Join and Leave modules will pass its output to IGMP Core for processing. All IGMP messages are sent through the Output module.

The IGMP Core supports IGMP messages handling function. It is responsible for processing the received IGMP messages and sending appropriate IGMP messages. Besides, it is able to handle timer event. Appropriate action is taken upon receiving a timer event.

At the same time, it also provides services to manage group membership entries. These services include new entry adding, entry removing, and entry information manipulating. Entry information manipulating includes the entry field value obtain, flag set, flag reset, entry status determination, reference count increasing and decreasing.

Furthermore, the IGMP Core also provides the notification of joining and leaving event to the multicast routing protocol.

The Input module and the Output module are used as an interface to receive and send IGMP message from and to lower layers. IGMP messages are passed through these two modules before going into or out from the IGMP Core.

The Join module and the Leave module are used to handle member join and leave for a host. When a host desires to join or leave a multicast group, an event is sent to the Join module or the Leave module. In these two modules, membership joining and leaving is processed respectively with cooperation from the IGMP Core. Table 4.1 shows the brief description of the functions for each module.

Table 4.1 Descriptions of Functions for Each Module

IGMP Module	Functions
IGMP Core	Processes and handles group membership entries, IGMP message (receive and send), IGMP timer, IGMP event (notification), determines status of IGMP whether it is used in host or router and other assistant function.
Input	Receives IGMP message and passes it to IGMP Core
Output	Sends IGMP from IGMP Core
Join	Handles members join event
Leave	Handles members leave event

The IGMP Core uses IGMP Group Table in managing the IGMP entry and the IGMP Timers for every timer in IGMP.

4.2.1.2 IGMP Group Table Design

The IGMP group table defines the information needed in an event of joining a multicast group.

4.2.1.3 IGMP Timers Design

An IGMP timer is designed to provide timing services. All timers in IGMP are created and destroyed in the IGMP timer element. Other than that, it provides a service to operate a timer, such as timer start and timer stop. Each timer includes information that is used to differentiate one from another.

4.2.2 Object-oriented Design

The system architecture design for IGMP in section 4.1.1 is translated into objects. Figure 4.3 shows the object diagram for IGMP. An *IGMPv2* class is created as an object to provide functions of IGMP. It is implemented in both *IPBTE* class and *ATMLSR* class. *IGMPv2* class consists of a *Group* class and a *IGMPTimer* class. *SimParamGroupTable* is used to provide user interface for IGMP membership table displays. Besides, *IGMPv2* class uses other existing classes for testing and simulation purposes.

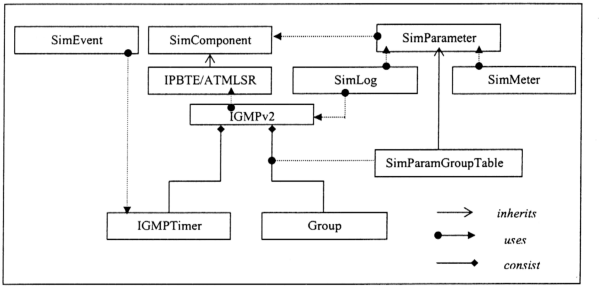

Figure 4.3 IGMP Objects in UMJaNetSim

4.2.3 Class Design

Four new classes are created for the implementation of IGMP. Besides, modification is done in the existing *IPBTE* class and *ATMLSR* class. In this section, the important attributes and methods for each class are presented.

4.2.3.1 *IGMPv2* Class

IGMPv2 class is a class that provides all functions of IGMP. This class is called by the *IPBTE* class and the *ATMLSR* class. The *IGMPv2* class includes the attributes and methods below.

- Important Attributes

  -A list of membership entries

  -A list of IGMP timers

  -A status indicating whether a host or router (querier and non-querier)

- Important Methods

  -Initialization

    -Initialize *IGMPv2*

    -Set status of implementation of *IGMPv2*

  -Main interfacing

    -Input IGMP

    -Output IGMP

    -Join group

    -Leave group

  -Message handling

    -Send report

    -Send query

    -Send leave

    -Receive report

    -Receive query

    -Receive leave

  -IGMP notification

    -Notify add

    -Notify remove

  -Group membership entry handling

    -Add entry

    -Remove entry

    -Set state of an entry

    -Increase reference count

    -Decrease reference count

    -Set last report flag

    -Clear last report flag

-Check if entry exists

-IGMP timer handling

-Handle startup query timer timeout

-Handle unsolicited report timer timeout

-Handle query timer timeout

-Handle query response timer timeout

-Handle group membership timer timeout

-Handle other querier present timer timeout

-Handle last membership query timer timeout

4.2.3.2 *Group* class

Group class is a private class in the *IGMPv2* class. It is used to define the elements of group membership entry. The followings are the attributes of the *Group* class.

- Important Attributes

  -A state indicates the status for an entry

  -A group address that defines the joined multicast group

  -An interface where the group member exists

  -A reference count that indicates the number of process still interested in the group (host only)

  -A last report flag that indicates the last party that send IGMP report (host only)

4.2.3.3 *IGMPTimer* class

IGMPTimer class is another private class in the *IGMPv2* class. It defines the timers for IGMP and includes some basic function of timer manipulating. It cooperates with the functions of timer handling in the *IGMPv2* class. The following are the attributes and methods of the *IGMPTimer* class.

- Important Attributes

  -An event type of timer

  -An expiring time of a timer

- Important Methods

  -Start timer

-Stop timer

-Clear timer

4.2.3.4 *SimParamGroupTable* Class

SimParamGroupTable class is used to provide GUI view in displaying group membership entries for both host and router. The main purpose is to show and validate that the IGMP protocol performs correctly by checking the current entries. It is an assisting simulation tools rather than a working component required by IGMP. It is included in the *IGMPv2* class.

4.2.3.5 Modification in *IPBTE* Class

IPBTE class can be simulated as an interface card for a node, or an encapsulated sub network attached to a router. An IGMP module resides in the *IPBTE* and a few methods have to be modified. Stated below are important modifications that must be done in the IPBTE class.

- Important Modifications

  -Defines *IGMPv2* object (in constructor)

  -Sets the state of *IGMPv2* to HOST

  -Includes initialization of *IGMPv2*

  -Calls *SimParamGroupTable* object from *IGMPv2* object

  -Includes IGMP messages, timer and event listener

4.2.3.6 Modification in *ATMLSR* Class

In this study, the *ATMLSR* class is used to simulate functionality of a router. Hence, it needs to be modified so that IGMP is supported in the router. Stated below are important modifications that must be done in the *ATMLSR* class.

- Important Modifications

  -Defines *IGMPv2* object (in constructor)

  -Sets the state of *IGMPv2* to QUERIER

  -Includes IGMP initialization

  -Add *SimParamGroupTable* object from *IGMPv2* object

-Include IGMP messages, timer and event listener

-Include IGMP messages send and receive handler to deal with *IGMPv2* object

4.3 PIM-DM Design

4.3.1 System Architecture Design

From the analysis on PIM-DM in the previous chapter, a system architecture for PIM-DM has been designed and is shown in Figure 4.4. Basically, PIM-DM supports PIM-DM timers and three entry tables, namely PIM-DM SG (source and group) Table, PIM-DM Neighbor Router Table and PIM-DM Membership Table. PIM-DM Core is the main control part in managing and providing PIM-DM functions.

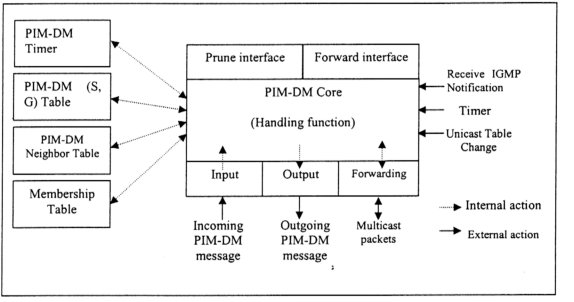

Figure 4.4 Basic System Architecture for PIM-DM

4.3.1.1 PIM-DM Core Design

In the main control part of PIM-DM, three important interfacing modules are Input module, Output module and Forwarding module. The Input modules and the Output modules are used for receiving and sending PIM-DM control messages respectively, while the Forwarding module handles the forwarding of multicast packets. The received PIM-DM control messages are passed to PIM-DM core, where processing of PIM-DM control messages are carried out. Any outgoing PIM-DM control messages are then

passed out through Output modules. Forwarding module is used to handle the forwarding of multicast packets.

In the PIM-DM Core, Prune Interface and Forward Interface are two other important modules. The Prune Interface module is used to prune an interface while the Forward Interface module is used to turn an interface into forwarding state.

There are other functions in the PIM-DM Core, including entries handling, timer event handling, IGMP notification handling, and so on. The PIM-DM Core uses the three tables to keep PIM-DM entries.

4.3.1.2 PIM-DM SG Table Design

It contains all elements in an SG entry.

4.3.1.3 PIM-DM Neighbor Table Design

It contains information of the PIM-DM neighbor router.

4.3.1.4 PIM-DM Membership Table Design

It contains information of membership join, which is received from the IGMP Notification.

4.3.1.5 PIM-DM Timers Design

It is responsible for creating and destroying all the timers in PIM-DM. Additionally, it provides service in managing timers, such as start timer and stop timer. Each timer includes information to differentiate one from another.

4.3.2 Object-oriented Design

The system architecture design for PIM-DM in section 4.1.1 is translated into objects. Figure 4.5 shows the objects diagram for PIM-DM. A *PIMDM* class is created as an object to provide functions of PIM-DM. It is implemented in the *ATMLSR* class. The *PIMDM* class consists of a *SG* class, *PIMDMNeighbor* class, *PIMDMMembership*

class, and *PIMDMTimer* class. The *SimParamPIMDMTable* and the *SimParamPIMDMNeighbor* class are used to provide user interface for PIM-DM SG entry displays and PIM-DM neighbor information displays respectively. Besides, the *PIMDM* class uses other existing class for testing and simulation purposes.

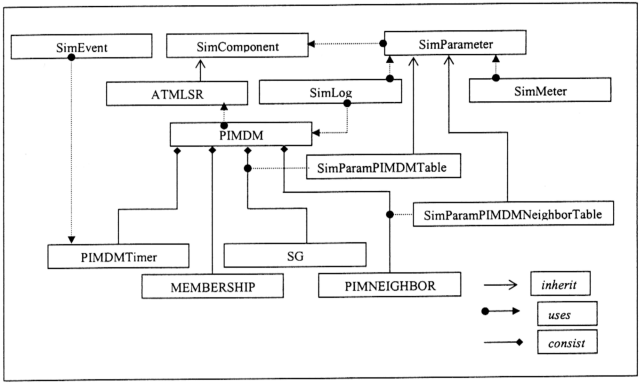

Figure 4.5 PIM-DM Objects in UMJaNetSim

4.3.3 Class Design

Seven new classes are created for implementation of PIM-DM. The following are some important attributes and methods of each class.

4.3.3.1 *PIMDM* Class

PIMDM class is a class that provides all functions of PIM-DM. This class is called by the *ATMLSR* class. The *PIMDM* class includes the attributes and methods below.

- Important Attributes
  - -A list of SG entries
  - -A list of PIM-DM neighbor router entries

-A list of PIM-DM membership entries

-A list of PIM-DM timers

- Important Methods

-Initialization

-Initialize PIM-DM

-Main interface

-Input PIM-DM

-Output PIM-DM

-Get forwarding list

-Handle with not RPF multicast packets

-Interface handling

-Prune interface

-Turn into forward interface

-Neighbor handling

-Process generation ID

-Handle new generation ID

-PIM-DM event handling

-Handle notification add from IGMP

-Handle notification remove from IGMP

-Handle unicast table change

-Message handling

-Send Hello message

-Send Prune message

-Send Graft message

-Send Graft-Ack message

-Send Assert message

-Receive Hello message

-Receive Prune message

-Receive Graft message

-Receive Graft-Ack message

-Receive Assert message

-SG entry handling

-Add entry

-Remove entry

-PIM-DM neighbor entry handling

-Add entry

-Remove entry

-PIM-DM membership handling

-Add entry

-Remove entry

-PIM-DM timer event handling

-Handle Hello timer timeout

-Handle Neighbor timer timeout

-Handle Prune timer timeout

-Handle data timeout timer timeout

-Handle Graft-Ack timer timeout

-Handle Assert timer timeout

4.3.3.2 *SG* Class

SG class is a private class in the *PIMDM* class. It is used to define the elements of PIM-DM (S, G) entry. The following are the attributes of the *SG* class.

- Important Attributes

  -A state indicates the status for an entry

  -A source address that indicates the source of the multicast packets

  -A group address that defines the multicast group

  -An incoming interface for the received multicast packets

  -A list of outgoing interface for the group member

4.3.3.3 *SGOF* Class

SGOF class is a private class in the *SG* class. It is used to define the elements of outgoing interface in PIM-DM (S, G) entry. The followings are the attributes of the *SGOF* class.

- Important Attributes
  -An outgoing interface for the group member

4.3.3.4 *PIMNeigbor* Class

PIMNeighbor class is a private class in the *PIMDM* class. It is used to define the information of a PIM-DM neighbor router. The followings are the attributes of the *PIMNeighbor* class.

- Important Attributes
  -A PIM-DM neighbor address
  -A generation ID
  -An interface where the PIM-DM neighbor exist

4.3.3.5 *PIMDMTimer* class

PIMDMTimer class is another private class in the *PIMDMTimer* class. It defines the timers for PIM-DM and includes some basic function for manipulating timers. It cooperates with the functions of timer handling in *PIMDM* class. The following are the attributes and methods of the *PIMDMTimer* class.

- Important Attributes
  -An event type of timer
  -An expiring time of a timer
- Important Methods
  -Start timer
  -Stop timer
  -Clear timer

4.3.3.6 *SimParamPIMDMTable* Class

SimParamPIMDMTable class is used to provide GUI view for PIM-DM (S, G) entries display. It is an assistant simulation tools rather than a working component required by PIM-DM. It is included in the *PIMDM* class.

4.3.3.7 *SimParamPIMDMNeighbor* Class

SimParamPIMDMNeighbor class is used to provide GUI view for displaying PIM-DM neighbor information. It is an assistant simulation tools rather than a working component required by PIM-DM. It is included in the *PIMDM* class.

4.3.3.8 Modification in *ATMLSR* Class

ATMLSR class is used to simulate functionality of a router. It needs to be modified so that PIM-DM is supported in the router. Stated below are important modifications that must be done in the *ATMLSR* class.

- Important Modifications
  -Define *PIMDM* object (in constructor)
  -Include initialization of *PIMDM*
  -Add *SimParamPIMDMTable* object from *PIMDM* object
  -Add *SimParamPIMDMNeighbor* object from *PIMDM* object
  -Include PIM-DM messages, timer and event listener
  -Include PIM-DM message send and receive handler to deal with *PIMDM* object

4.4 Multicast Applications Design

In order to simulate and evaluate IP Multicast using PIM-DM, a multicast application is needed. Hence, two simulation applications are created. There are *MulticastdataApp* class and *IPMulticastApp* class. The *MulticastdataApp* class simulates a sender or a source of a multicast application, while the *IPMulticastApp* class simulates a receiver or a member a of multicast application

4.4.1 *MulticastdataApp* Class

This class is used to create an application to send multicast packets to a multicast group. It provides a simulation on a multicast application. It is designed for the purpose of simulating, therefore the contents of the multicast packets are not important and it is a

empty packet. Some values need to be specified before simulation starts. These values are stated below.

- Important Attributes
  -Multicast group address
  -Start sending time
  -Stop sending time
  -Data rate
- Important Methods
  -Start sending
  -Stop sending
  -Send multicast packets

4.4.2 *IPMulticastApp* Class

This class is used to simulate the multicast group joining, group leaving, and packet receiving of a multicast application. Once a multicast group is joined, it could receive multicast packets from the particular multicast group.

To simulate the joining and leaving process, some values need to be specified before simulation. These values are stated below.

- Important Attributes
  -Multicast group address
  -Join time
  -Join duration
- Important Methods
  -Join
  -Leave
  -Receive multicast packets

4.5 Summary

In this chapter, the designs of IGMP, PIM-DM and the simulation application of multicast are discussed. In each section, the important attributes and methods, which are required for respective design, are listed. The implementations of the system depend on these designs.

In the next chapter, the implementations of the design are discussed.