Chapter 2 Literature Review

In this chapter, a review on IP Multicast, computer simulation and UMJaNetSim is presented. The first part of this chapter concentrates on IP Multicast. The second part of this chapter concentrates on computer simulation. The last part analyzes UMJaNetSim.

In section 2.1, an introduction of IP Multicast is presented. Advantages and applications of IP Multicast are the main topic to be included. This is followed by a description on multicast address, IGMP, multicast algorithms and multicast protocols.

In section 2.3, a brief description on computer simulation is presented. This begins by an introduction of the computer simulation. The following sections describe two different programming approaches in developing a simulator, namely procedural approach and object-oriented approach. Then, a brief description on Java is presented. Next is an overview of computer network simulator, concentrating on UMJaNetSim network simulators.

Section 2.4 is an analysis on UMJaNetSim. An overview of the architecture and Application Programming Interface (API) of UMJaNetSim are presented in two separate sections. These two sections will briefly describe the function and process involved in three fundamental architectures and the main components in UMJaNetSim.

The final section in this chapter gives a summary of the chapter.

2.1 IP Multicast

2.1.1 Introduction

Today, Internet involves applications of one-to-one, one-to-many, many-to-many, and many-to-one communication patterns. Unicast is one-to-one communication pattern while many-to-one can be done by either unicast or multicast. However, no standards exist for alternate and equivalent many-to-one application designs yet. [7]

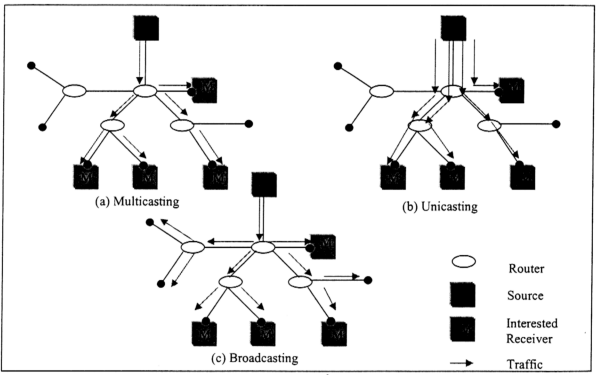

Figure 2.1 Three Approaches in One- to- Many Applications

For one-to-many and many-to-many communication patterns, packet forwarding can be achieved in three different ways, which are multiple unicasting, broadcasting, and multicasting. The first approach is a method where multiple unicast traffic is sent to multiple destinations whereas the second approach broadcasts the traffic to entire networks so that it could be received by the desired destination. However, these two approaches are not efficient. [8] The first solution consumes a large amount of bandwidth, while the second solution not only increases the traffic load but also creates unnecessary traffic. By applying multicasting, only one packet is sent in order to reach

all interested receivers. In other words, IP Multicast is a mechanism that sends a single copy of an IP packet to all members in a multicast group, rather than sending out multiple copies of an IP packet to all members. [9] It provides many of advantages and improves efficiency compared to the other approaches.

2.1.1.1 Advantages of IP Multicast

One of the advantages of using IP Multicast is it can decrease network load. [10] IP Multicast transmits only one packet by the source and replicates the packet only when it is necessary. Hence, it utilizes less bandwidth compared to unicast or broadcast approaches. The difference in bandwidth utilization is significant especially multimedia traffic, such as movie applications, is involved.

Multicast can also be used in resource discovery. There are certain applications, such as Bootstrap Protocol and Open Shortest Path First (OSPF), that uses multicast to find available services. [10]

Multicast is also very important in distributing multimedia data. Nowadays, multimedia application has become important in IP networks. Applications like audio-cast and video-cast have become popular in the Internet. Instead of establishing point-to-point connections to multiple participant hosts, multicasting is employed to send the multimedia data. Using IP Multicast not only decreases bandwidth utilization, but also provides flexibility in joining and leaving a group. [10]

2.1.1.2 Multicast Applications

Multicast applications can be either a sending source, a receiver, or both. A receiver must express an interest to become a member of a multicast group before it could receive any multicast from the multicast group. A source is not necessarily a member of a multicast group.

Basically, multicast applications can be divided into two types, i.e. one-to-many and many-to-many. One-to-many is a communication pattern, which normally has a source

as the sender and multiple members as the receiver. Many-to-many applications involve multiple senders and multiple receivers, in which a sender is also a receiver. Examples for one-to-many are the transmission of corporate messages to employees, communication of stock quotes to brokers, live transmission of news coverage, web TV, distributed databases, distance learning, and so on. Examples for many-to-many applications are video and audio conferencing, telecommuting, replicating databases, share whiteboard applications, multi-player games and so on.

2.1.2 Model of IP Multicast

IP Multicast is the transmission of an IP packet to group of zero or more hosts identified by a single IP multicast group address. A multicast packet will be delivered to all members of the multicast group. The membership of a multicast group is dynamic, in which hosts may join and leave multicast groups at any time and anywhere. There is no limitation to the total number of members. A host may join as a member in more than one group at a time. [11]

IP Multicast is an extension of the implementation of Internet Protocol. [9] By adding multicast to the IP networks, some extra protocols and modifications need to be defined and implemented.

Firstly, multicast group addresses are required. A multicast group address is designed to enable a sending source to specify the destination group. It also enables a host to express its interest in joining and leaving a multicast group.

If a multicast source and a multicast receiver are in the same LAN, the multicast receiver needs only to listen to all multicast packets that are sent to that particular multicast group. However, when the multicast receivers are located some where in a WAN, the multicast receivers must employ a protocol to inform a router so that the interested multicast packets can be forwarded to the multicast receivers. [10] Hence, IGMP is needed in both host and router to handle the joining and leaving of a multicast

group in a subnet. IGMP will notify multicast routing protocol in a router regarding the multicast membership condition.

Multicast routing protocol is needed to construct the multicast packet delivery trees and to perform efficient multicast packet forwarding. [9] There are various multicast forwarding algorithms available today. Multicast routing protocol employs multicast forwarding algorithms to establish a multicast tree and carries out multicast packet forwarding.

Multicast applications for creating and managing the multicast data are also needed. Figure 2.2 shows the basic model for IP Multicast.

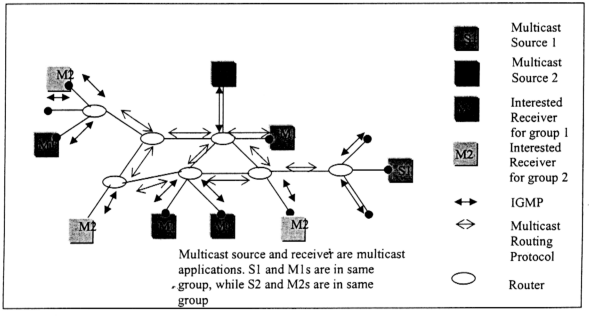

Figure 2.2 Basic IP Multicast Model

2.1.3 Multicast Group Address

Multicast is based on the concept of a group. [12] Each multicast group is assigned an identification address, which is called multicast group address. Multicast receivers express its interest in joining a group by informing the multicast group address to the

IGMP. Meanwhile, multicast group address is included as the destination address in every multicast packet when it is sent out.

Class D IPv4 address is used for multicast group address, and the addresses range from 224.0.0.0 to 239.255.255.255. [11] However, Internet Assigned Numbers Authority (IANA) maintains a list of registered IP multicast group. [4] Table 2.1 lists the assignment of multicast group address from IANA.

Table 2.1 Assignment of Multicast Address from IANA

Multicast Addresses	Name	Descriptions
244.0.0.0	Reserved	Reserved and not assigned
224.0.0.1- 224.0.0.255	Reserved Link Local Addresses	Reserved for routing protocols, other low level topology discovery or maintenance protocol
224.0.1.0- 238.255.255.255	Globally Scoped Address	Assigned to multicast applications
239.0.0.0- 239.255.255.255	Limited Scope Addresses	Reserved for site-local administratively scoped applications

2.1.4 Internet Group Management Protocol (IGMP)

IGMP is a protocol that handles multicast group. It is implemented between hosts and neighboring routers. IGMP helps routers to identify members of a multicast group in a subnet. It allows a host to express interest to join and leave a certain multicast group.

IGMP is a companion to the IP protocol and is located at network layer. The IGMP message is encapsulated in an IP packet with the protocol value of two. Figure 2.3 shows the position of the IGMP protocol in relation to other protocols in the network layer, while Figure 2.4 shows the encapsulation of IGMP message.

Basically, operation of IGMP can be divided into 4 parts. There are joining, monitoring, continuing and leaving. At first, a host must join a multicast group by informing

neighbouring router through IGMP Report message. The router is responsible to monitor the membership condition for the multicast group by sending Query message through IGMP. Hosts that are still interested in joining a multicast group should inform router when a Query message is received. When a host decides to leave, it should also inform the neighbouring router. [3]

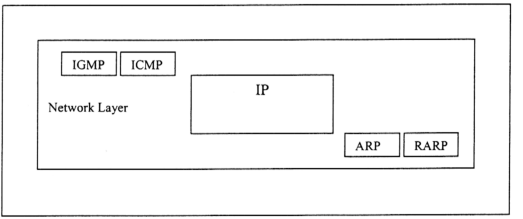

Figure 2.3 Position of IGMP in Network Layer

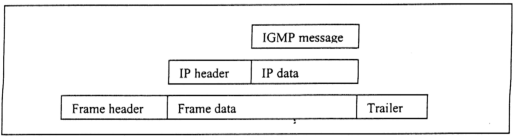

Figure 2.4 Encapsulation of IGMP Packet

Currently, three versions of IGMP have been defined. There are IGMP version 1[11], IGMP version 2 [13], and IGMP version 3 [14]. IGMP version 3 is a draft specification and is not deployed yet.

2.1.4.1 IGMP Version 1

In IGMP version 1, only two types of IGMP message have been defined, which are Host Membership Query message and Host Membership Report message (hereafter it is

known as IGMP Query and IGMP Report respectively). [11] When a host wants to join a multicast group, it sends an IGMP Report to join the multicast group to a neighbouring router. However, in IGMP version 1, a host will not acknowledge a router when it decides to leave a multicast group.

Meanwhile, a router sends an IGMP Query periodically to learn about the membership condition in its subnet. A host responds to the query if it still wants to continue the multicast group membership. An IGMP Reports is sent to inform the router the multicast groups that it still interested in. To avoid simultaneous responses from hosts, which may create extra traffics, random response timers are set in every host that would like to send a reply. At the same time, the hosts will listen to the communication in the subnet. If a report has been made for the multicast group that the host is interested in, a duplicate Report will be suppressed. By doing so, unnecessary traffic is avoided. [11]

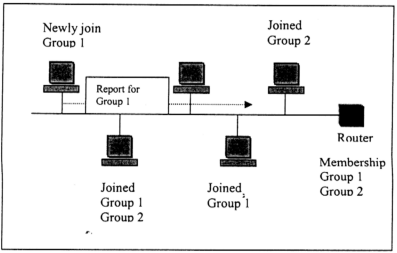

Figure 2.5 Reporting Membership

If the router does not receive any confirmation for a previously joined multicast group from the hosts in the subnet, the router assumes that no member exists in that particular multicast group. [11] The membership condition in a subnet is passed to the multicast routing protocol, so that multicast tree and forwarding decision could be made. Figure 2.5, Figure 2.6 and Figure 2.7 illustrate the mechanism in IGMP version 1.

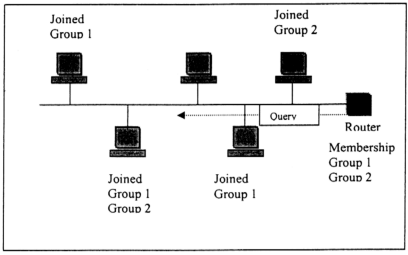

Figure 2.6 Monitoring Membership

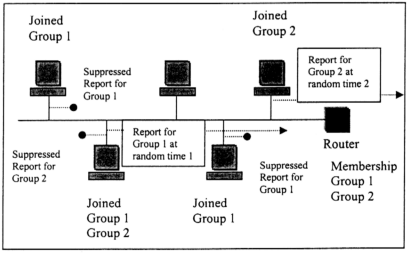

Figure 2.7 Continuing Membership

A multi-access network may have more than one router attached to the same network. In this case, only one router is elected as a querier to be responsible for sending IGMP Query message. The other routers are known as non-querier. They need only listen to the IGMP Report in order to keep track of membership conditions in that subnet. The election of a querier in IGMP version 1 depends on multicast routing protocol. [11]

2.1.4.2 IGMP Version 2

IGMP version 2 is an extension of IGMP version 1. IGMP version 2 enhances the performance of IGMP version 1 and provides backward compatibility with IGMP version 1. [4]

In general, there are 2 enhancements with four new inclusions in IGMP version 2. Handling of membership leaving and election of querier in a multi-access network are introduced in this version. In IGMP version 1, no mechanism is defined to handle cancellation of a membership, while the querier election in a multi-access network is determined by the multicast routing protocol. [4] The first enhancement minimizes the leave latency whereas the second enhancement provides standardization in the election of querier. If different querier election mechanism is implemented in different multicast routing protocols, it may cause unexpected problems.

Leave message is introduced in IGMP version 2 in order to handle membership cancellation. If the host that wishes to leave a multicast group is the last host that responds to the IGMP Query message for that multicast group in the subnet, a leave message is sent. Otherwise, it will leave the multicast group silently. This is because other members still wish to join that multicast group. [13]

Two modifications were done on the existing Host Membership Report message and Host Membership Query message.

Host Membership Query message, which is known as Membership Query message in IGMP version 2, is divided into two types, namely General Query message and Group-specific Query message. The function of the General Query message is similar to the Host Membership Query message in IGMP version 1. The newly introduced Group-specific Query message enables routers to send a query for a specific multicast group. Group-specific Query message is used to handle the membership leaving. When a router is informed of a leaving event of a multicast group, it sends a Group-specific Query message to learn whether there are other members that are still interested in

joining that group. [13] Figure 2.8, Figure 2.9 and Figure 2.10 illustrate the leaving event in IGMP version 2.

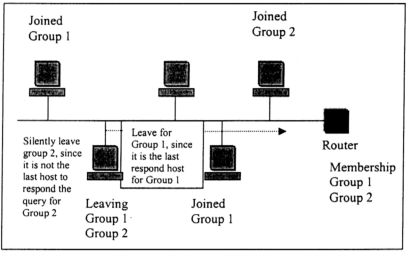

Figure 2.8 Leaving Multicast Groups

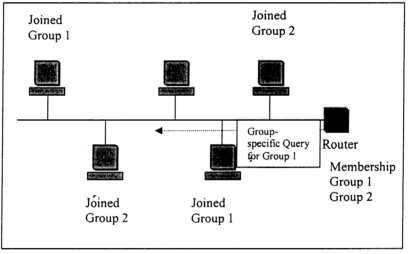

Figure 2.9 Group-specific Query

There are two types of Membership Report message in IGMP version 2, namely Version 2 Membership Report and Version 1 Membership Report. [13] The former message is used to report to the router, which runs on IGMP version 2 while the latter is used to report to the router that runs on IGMP version 1.

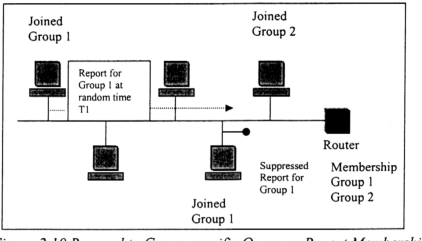

Figure 2.10 Respond to Group-specific Query or Report Membership

2.1.4.3 IGMP Version 3

IGMP version 3 introduces the support for Group-Source Report message. This message is used to inform a router that the host wishes to receive traffic from a specific source of a multicast group. [14] Previously, in IGMP version 1 and IGMP version 2, a host is unable to specify the source in a multicast group that it would like to receive. The members of the group will receive traffics sent to the multicast group from all sources.

The other enhancement in IGMP version 3 is the inclusion of Group-Source Leave message. [14] Similar to the Group-Source Report message, Group-Source Leave message allows a host to leave a certain source in a multicast group. This means, a host is able to inform the router when it is no longer interested in receiving traffic from a specific source in a particular multicast group.

2.1.5 Multicast Forwarding Algorithms

As mentioned, IGMP is only concerned with the membership condition for multicast groups between neighbouring routers and hosts in subnets. The multicast packets forwarding is done by the multicast routing protocol in routers.

In IP Multicast, the membership for a multicast group is dynamic and the receivers are located separately in the entire network. Hence, the routers may not know where to forward multicast packets. Therefore, a multicast routing protocol is needed so that the routers know where to forward multicast packets and the multicast packets could be delivered to the correct destinations. The multicast packets from a multicast group must be delivered to every single subnet, where at least one member exists for that multicast group.

To achieve this, a multicast tree is constructed by using the multicast routing protocol. It is used to forward multicast packets by the routers. There are a few different forwarding algorithms employed by the multicast routing protocol to establish the multicast forwarding trees. Basically, there are three techniques in constructing a multicast tree. There are simple-minded, source rooted trees and shared trees. [7] These algorithms are stated below.

- Simple-minded
  -Flooding
  -Spanning Trees
- Source rooted trees
  -Reverse Path Broadcasting (TPB)
  -Truncated Reverse Path Broadcasting (TRPB)
  -Reverse Path Multicasting (RPM)
- Shared trees
  -Core Based Trees

2.1.5.1 Flooding

Flooding is a simple algorithm. When a multicast packet is received, the router checks whether it is a newly received multicast packet or it has seen this particular packet earlier. In the former case, the received multicast packet is forwarded on all interfaces

except the incoming interface of the multicast packet. In the latter case, the received multicast packet is simply discarded. Figure 2.11 shows the flooding algorithm.

The advantage of flooding is that a router is only required to keep track of the most recent packets and does not need to maintain a routing table. Flooding could guarantee that the multicast packet reaches all routers in the entire network. Indirectly, this assures that all members will receive the multicast packet. However, flooding creates many duplicate packets and increases the traffic load. Hence, flooding is not able to scale widely. [9]

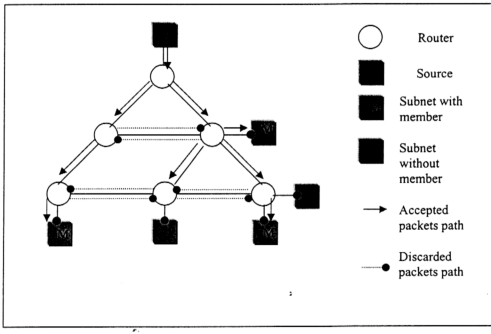

Figure 2.11 Flooding Algorithm

2.1.5.2 Spanning Tree

The spanning tree approach is implemented by selecting a subset from interconnected links to be constructed as a tree structure. The tree is constructed such that only one active path connects any two routers in the entire network. [10] Figure 2.12 illustrates an example of a spanning tree in a network topology.

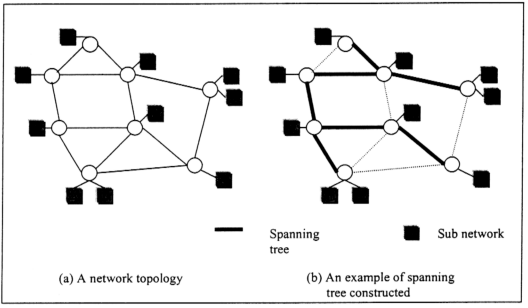

Figure 2.12 Example of a Spanning Tree

Multicast packets are forwarded through the interfaces, which are a part of the spanning tree except the interface from where the multicast packets have arrived. Figure 2.13 shows the spanning tree algorithm in multicast packets forwarding.

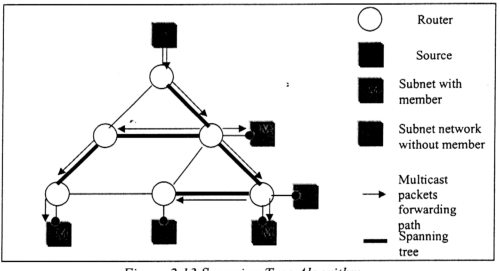

Figure 2.13 Spanning Tree Algorithm

Spanning tree is relatively easy to implement. It assures the loopless of multicast path to all routers in the entire network. [9] If compared to flooding algorithm, spanning tree creates less unnecessary traffic in the network. However, in spanning tree approach, traffic is concentrated on a set of interconnected links. This may lead to traffic overloading on certain links. Furthermore, the delivery path between source and multicast receivers is not necessarily the most efficient path.

2.1.5.3 Reverse Path Broadcasting (RPB)

RPB uses the concept of spanning tree algorithm. A spanning tree is implicitly constructed for each router in which a source is attached. [10] In other words, several spanning trees rooted at different source routers are constructed for each sending source, instead of building a network-wide spanning tree. This means if there are three sending sources in a multicast group, there will be three different spanning trees available, one for each source.

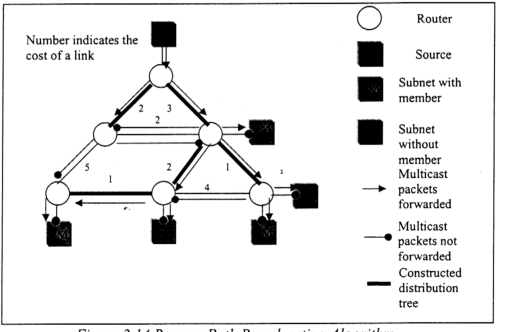

Figure 2.14 Reverse Path Broadcasting Algorithm

When a multicast packet arrives at a router, the router checks if the incoming link of the received multicast packet is the shortest path towards the originating source of that

multicast packet. If it is, the multicast packet is forwarded to all interfaces except the incoming interface. Otherwise, the packet is discarded. Hence, in RPB, unicast routing information is used as a reference in determining whether a link is the reverse shortest path back to the source during establishment of spanning tree. Figure 2.14 shows the RPB algorithm.

To improve the performance of RPB, a router may check whether the neighbour's router on the outgoing interface is also on the shortest path towards the source of a multicast packet before forwarding is done. [10] A multicast packet is only forwarded to the reverse shortest path neighbour router. By applying this mechanism, duplicate multicast packets can be avoided. Figure 2.15 shows the enhanced RPB algorithm.

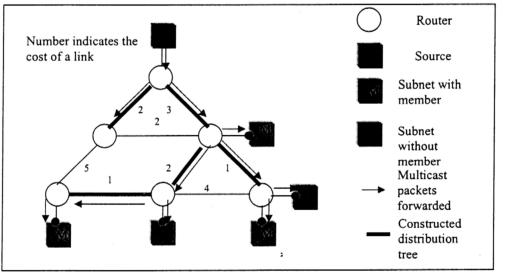

Figure 2.15 Enhanced Reverse Path Broadcasting Algorithm

RPB is relatively efficient and easy to implement. It employs existing unicast routing information as a reference in forwarding a multicast packet. Hence, the router is not required to remember the spanning trees. Furthermore, if the cost of a link state is the same in both directions, the multicast packet is forwarded along the shortest paths. This guarantees efficiency in delivery. In addition, since there may be several of spanning trees for multiple sources multicast group, the traffic is distributed over multiple paths. Therefore, it indirectly avoids traffic concentration. However, one of the major

limitations in RPB is that RPB does not take into account the information about multicast group membership when constructing the spanning trees. [4] This leads to unnecessary delivery of multicast packets to a site where no member exists.

2.1.5.4 Truncated Reverse Path Broadcasting (TRPB)

TRPB is another enhanced RPB, which is designed to overcome a few limitations in RPB algorithm. [4] TRPB avoids from forwarding multicast packets to subnets where no members exist.

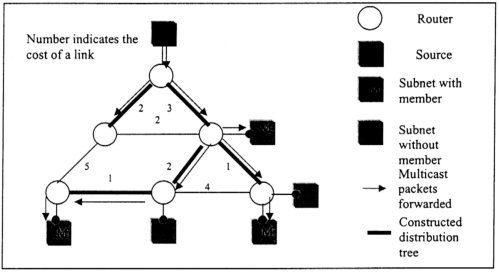

Figure 2.16 Truncated Reverse Path Broadcasting Algorithm

Information in IGMP is used to tear down the delivery path to the subnets that does not have any group members. However, TRPB does not solve the unnecessary delivery of multicast packets to the routers that do not wish to receive and forward multicast packets due to no existence of members in the attached subnets. [9] Figure 2.16 shows the TRPB algorithm.

2.1.5.5 Reverse Path Multicasting (RPM)

RPM is an enhancement of RPB and TRPB. It solves the limitation in TRPB by adding a prune function. It is used to inform routers not to forward the packets to routers that are not interested in particular multicast packets.

In RPM, multicast packets are only forwarded toward routers that are on the path leading to the leaf routers, which have members inside the attached subnets. However, the first multicast packet received by routers is forwarded based on TRPB. If there is no member in every subnet that are attached to a leaf router, the leaf router will inform its upstream router not to forward the multicast packets towards it. This can be done by sending a prune message. Once a router receives a prune message from an outgoing interface of a multicast group from certain source (hereafter it is refer to (S, G), where S is the source and G is the group), the router stops forwarding that particular multicast packet through that interface. If an upstream router does not have any member attached to it in the subnet, and it receives prune message form all the downstream routers, the upstream router will trigger a prune message to its upstream router towards the sending source. Finally, a distribution tree is implicitly established. The prune message is always sent only one hop back towards the sending source. [4] Figure 2.17 shows the RPM algorithm.

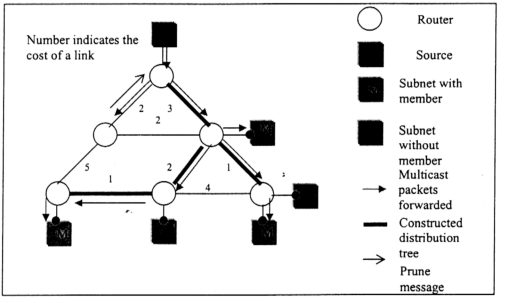

Figure 2.17 Reverse Path Multicasting Algorithm

Since the membership condition is dynamically changing, the distribution tree must be periodically refreshed. This is to ensure that the new members along a previously pruned path can receive the desired multicast packets at minimum latency. [10]

Inappropriate refreshment handling may increase the join latency for multicast group members. Besides, in RPM, routers need to maintain state information on multicast distribution trees for each pair of multicast group and multicast sending source. [10]

2.1.5.6 Core Based Trees (CBT)

CBT [15] is a shared tree algorithm. CBT constructs a single delivery tree that is shared by all members in a multicast group. It is almost similar to the spanning tree algorithm, except that CBT allows different spanning trees for different multicast groups. [4]

In CBT, a set of routers is elected to act as core routers, which are the root of the distribution tree. [15] One multicast tree is probably rooted at one of the router from the core router set. Each host that would like to join a multicast group will send a join request towards the core router. On the path towards the core router, intermediate routers mark the interface where a join request is received. The combination of the information in all routers creates a distribution tree for a multicast group.

Each sending source for a multicast group is also sending its traffic toward the core router. The core router will forward the multicast packets based on the constructed distribution tree. Figure 2.18 shows the CBT algorithm.

Join requested from a multicast member and multicast packets sent by a source are done by unicasting. At the core router, multicast packets are multicast towards the members of the multicast group.

Compared to other source rooted tree algorithm, such as RPM algorithm, CBT needs only one distribution tree for one multicast group, instead of creating multiple distribution trees for multiple multicast sending sources. Hence, less information is maintained in a CBT router. Besides, CBT conserves network bandwidth. This is because routers are not required to forward multicast packets to all routers in the entire network periodically and are not required to wait for pruning from the downstream routers. [4] However, CBT may create traffics concentration as all traffic travels toward

a core router. In addition, the path used to deliver multicast packet from a source to multicast group members is not necessarily the most efficient path because all multicast traffic traverses the core router before reaching the members.

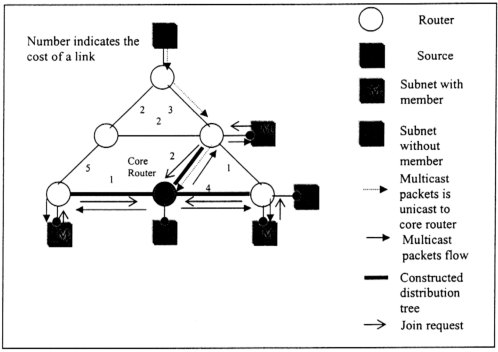

Figure 2.18 Core Based Trees Algorithm

2.1.6 Multicast Routing Protocols

There are many multicast routing protocols available today. Each of the protocols has its strengths and weaknesses. In general, existing multicast routing protocols can be classified into three types according to the method in establishing of the multicast tree. There are Reverse Path Multicast, Membership Advertisement and Center Based Trees. [2]

These three classes of multicast routing protocols can be further divided into two groups, namely dense mode and sparse mode. [16] Dense mode approach assumes that the memberships of a multicast group are densely distributed and bandwidth is sufficient. Hence, dense mode multicast routing protocols mostly rely on flooding to

propagate group information to all network routers. On the other hand, sparse-mode multicast routing basically assumes that the multicast group members are sparsely distributed throughout the network and bandwidth is limited. Thus, sparse mode multicast routing protocols rely more on selective techniques to set up and maintain multicast trees. [17] The multicast routing protocols are stated as below.

- Dense Mode Multicast Routing Protocols
  -Reverse Path Multicast
    -Distance Vector Multicast Routing Protocol (DVMRP)
    -Protocol Independent Multicast – Dense Mode (PIM-DM)
  -Membership Advertisement
    -Multicast extension to Open Shortest Path First (MOSPF)
- Sparse Mode Multicast Routing Protocols
  -Center-Based Trees
    -Core Based Trees
    -Protocol Independent Multicast – Sparse Mode (PIM-SM)

2.1.6.1 Protocol Independent Multicast – Dense Mode (PIM-DM)

The motivation for designing PIM-DM is to provide the construction of delivery trees in densely distribution pattern of multicast group members for PIM architecture. [4]

PIM-DM employs RPM to establish the delivery tree. It is a data driven protocol, where the delivery tree is established on the first received multicast data. PIM-DM constructs different delivery trees for each pair of source and destination group. Each delivery tree is a spanning tree to all multicast receivers rooted at the multicast source.

The operation of PIM-DM is very simple. Initially, PIM-DM assumes that when a source starts sending a multicast packet for a multicast group, all hosts in the entire network are the members for that multicast group. [18] Hence, when a router receives a new (S, G) multicast packet from an interface where it is the shortest path back to the sending source (hereafter known as Reverse Forwarding Path (RPF)), the multicast packet is forwarded to all downstream interfaces. The forwarding branches of the tree

are explicitly pruned if no member exists at those branches. A more detail description is given below.

When a source sends multicast packets to a multicast group, the router will check whether the (S, G) multicast forwarding entry exists. If it does, the multicast packet will be forwarded according to the forwarding entry. Otherwise, a new entry is created for that (S, G) multicast packet. The entry includes source address, multicast group address, the incoming interface and a list of outgoing interfaces. [18] If the multicast packet is coming from a reverse shortest path back to the source, the multicast packet is forwarded to all interfaces except the incoming interface.

At the leaf routers, if there is no member exists at the downstream subnet, a prune message is sent to upstream routers towards the source, so that the (S, G) multicast packets will not be forwarded to the routers.

On the other hand, PIM-DM support graft mechanism. It enables a previously pruned branch of a distribution tree to receive multicast packets from an upstream router immediately when a new member joins just before the periodical broadcasting occurs. A graft request is sent toward the sources of a multicast group.

As implemented in RPM algorithm, PIM-DM includes not only periodical flooding but also waits for a new set of prune messages. This allows PIM-DM to reinitiate the construction of the delivery tree so that the newly joined members could receive multicast traffics in a short latency.

One of the advantages of PIM-DM is that it is relatively simple to implement. Besides, it is flexible, as it does not depend on the unicast routing protocol. PIM-DM works well for a multicast group that is densely distributed in a campus network, but not for multicast group members that are sparsely distributed over a wide-area network. This is because the periodic broadcast behavior would affect performance. [18]

2.1.6.2 Distance Vector Multicast Routing Protocol (DVMRP)

DVMRP [19] was the first protocol defined in RFC 1075 to support multicast routing. It was driven from Routing Information Protocol (RIP). It has been widely used on the Multicast Backbone (MBONE) [20].

Initially, DVMRP used TRPB algorithm to construct the distribution tree. Later on, RPM is employed in order to enhance performance of DVMRP. [10] Hence, the operation of DVMRP is generally similar to PIM-DM. The main difference between DVMRP and PIM-DM is that PIM-DM is independent of the unicast routing protocol that is used in the network while DVMRP makes use of its own RIP-like method to compute the required unicast routing information. [9] DVMRP runs two routing protocols in order to support the unicast traffic and multicast traffic. Thus, DVMRP contains its own integrated unicast routing protocol. Routing table update that consists of previous hop router, source subnet and shortest path to the source subnet, is exchanged periodically between the DVMRP capable neighbor routers. A multicast forwarding table is constructed from the routing table, the membership information from IGMP and the received prune message. The created delivery tree in DVMRP provides the shortest path between the source and each multicast receiver in the group, based on the number of hops in the path. The DVMRP routing table update is also used for leaf router detection.

DVMRP provides tunneling multicast traffic across the parts of an IP network that have not support multicast. [4]

One of the advantages of DVMRP is it is relatively simple to implement. Another advantage is the modest processing demands that DVMRP places on routers. [7] However, DVMRP is more complex than PIM-DM. [9] The other weakness in DVMRP is the amount of multicast routing state information that must be stored in the multicast routers is large. All the multicast routers must contain state information for every (S, G). Hence, DVMRP does not scale well in sparsely distributed multicast members over a large network.

2.1.6.3 Multicast extension to Open Shortest Path First (MOSPF)

MOSPF [21] is defined in RFC 1584 and it is an extension to the unicast routing protocol, Open Shortest Path First (OSPF). It is built on top of OSPF version 2 in order to support multicast routing in the existing OSPF routing protocol. [22]

MOSPF is intended for implementation within a single routing domain or a network controlled by a single organization. MOSPF is dependent on the use of OSPF as the accompanying unicast routing protocol. Hence, link-state information is used in constructing multicast distribution trees. [23]

In MOSPF, a router periodically collects information about multicast group membership from IGMP. Then, this information is flooded to all other routers in the routing domain along with link-state information. Routers will use this update information to have a view on the membership information in the entire network, as what is done on OSPF. The routers understand the topology of the entire network and membership condition. Hence, the routers can independently calculate a least-cost spanning tree with the multicast source as the root and the group members as leaves. Since all routers periodically share link-state information, the distribution tree calculated at all routers will be same. [23]

Similar to DVMRP and PIM-DM, different distribution tree is calculated for each (S, G) multicast packets. However, MOSPF does not support tunnelling. [22] In fact, MOSPF does not scale well since it has to flood the link-state information among the routers periodically.

2.1.6.4 Core Based Trees (CBT)

CBT [24] is defined in RFC 2189. It is a centre-based protocol, which implements CBT algorithm. CBT constructs a single tree that is shared by all members of the group. Multicast traffic for the entire group is sent and received over the same tree, regardless of the source. [25]

A CBT shared tree has a core router that is used to construct the tree. Routers join the tree by sending a join message towards the core router. When the core router receives a join request, it returns an acknowledgment to the routers using the reverse path, thus forming a branch of the tree. In order to decrease latency, acknowledgements of join messages may be sent back by the upstream router, which is already become one of the branches on the path toward the core router. The router that sent the join is then connected to the shared tree. A sending source will send multicast packets to the core router and then the core router will forward the multicast packet to the distribution tree. [24]

Using shared tree can reduce in the amount of multicast state information that is stored in individual routers. Besides, shared tree is relatively easy to construct, and it reduces the amount of state information that must be stored in the routers. However, as stated in section 2.1.5.6, CBT aggregates traffic onto a smaller subset of links, resulting in a concentration of traffic around the core, which may degrade the performance of CBT. Hence, multiple cores are used to support load balancing and avoid the traffic concentration.

2.1.6.5 Protocol Independent Multicast-Sparse Mode (PIM-SM)

PIM-SM [26] is defined in RFC 2362. Similar to the CBT, PIM-SM is designed to restrict multicast traffic only to routers that are interested in receiving it.

PIM-SM constructs a multicast distribution tree rooted at a router called a rendezvous point (RP), which has almost similar role as a core router in CBT. [27] However, PIM-SM is more flexible and efficient than CBT. In CBT, only group-shared trees are constructed whereas for PIM-SM a multicast group member may choose to construct either a group-shared tree or a shortest-path tree. [23] This means the shortest path distribution tree can be created in PIM-SM from every member to every source in a multicast group.

Initially, a group-shared tree is constructed to support a multicast group. This tree is formed when the senders and receivers of a multicast group report to the rendezvous point. This is similar to CBT. Once the distribution tree is constructed, a member router can request a connection to a particular source via the shortest path tree by sending a join message to the source. Once the shortest path from source to receiver is created, the extraneous branches through the RP are pruned. Different types of trees can be selected for different sources within a single multicast group. [23]

2.2 Reviews on Computer Simulation

One of the objectives of this study is to experimentally simulate, test and evaluate PIM-DM multicast routing protocol through a simulation environment. Hence, computer simulation is studied.

2.2.1 Computer Simulation

Three general approaches are employed to evaluate the performance of a system. There are measurement tools, analytical techniques and simulation. [28] Measurement tool is used for real networks [29], while analytical techniques make use of a system of equations or mathematical solution to predict the result and performance of a network. A simulation uses a computer to evaluate a model numerically. It determines or estimates the performance and the characteristic of a model based on the data collected during simulation. [30]

Computer network simulation can produce dynamic environment and illustrate the reality as well as the complexity of the actual network. It is an effective tool to analyse and evaluate the behaviour of a network without using an actual network. Furthermore, simulation allows testing on various types of new topologies. Hence, to achieve the objective of the study, it is more suitable to use simulation as an evaluation tool.

Computer simulation is generally divided into three categories based on the simulation approach. These simulation approaches are Monte Carlo, Continuous and Discrete Event. [5] Monte Carlo simulation is a method that does not require explicit representation of time whereas Continuous simulation uses continuous function of variables within the simulation. Discrete Event simulation allows alteration of program variables at any finite number of times during simulation. However, in an actual simulation, a combination of different techniques are involved.

In general, simulation is divided into three main fields. There are model design, model execution and model analysis. [6] At the first stage of the computer network simulation, a model of a theoretical or actual network is designed. Basically, the characteristic of a network is studied and then compiled into a simulation program by using a programming language. The next step is to execute the simulation. Normally, simulation is executed on a model and each model has a set of predefined input to be executed. Then, the output from the execution is analysed in order to obtain the desired result. [31]

A model is an abstraction of a system intended to replicate some properties of that system. [32] A model represents the characteristics and the problem domain of a system simulated. A model must conform to the objective of a simulation. A simulation model can consist of a collection of objects that interact with each other.

2.2.2 Programming Approach and Language

A network simulator is developed with a programming language on certain programming approach. An efficient and powerful simulator is highly related to the programming language and the programming approach.

2.2.2.1 Approach

In general, two programming approaches are used to develop a network simulator. There are procedural approach and object-oriented approach.

Procedural approach utilizes procedural programming language, such as Pascal, C and FORTRAN. Procedural approach tends to be action-oriented. Groups of action that perform some common tasks are formed into a function or procedure. These functions are grouped to form a program. [33] Procedural approach combines all the function or procedure step by step from start to finish. It corresponds to a step-by-step list of computations. Since network simulator requires concurrent processing to simulate the real network, it is not suitable to implement procedural approach in network simulator.

An object-oriented approach is implemented with object-oriented programming language. Examples of object-oriented programming languages are Java, C++, Smaltalk and so on. In the object-oriented approach, software is a collection of discrete objects that encapsulate their data and functionalities to model real world objects. [36] The primary concepts of the object-oriented approach are encapsulation, inheritance and polymorphism [37].

Encapsulation allows an object to be associated with a set of properties and methods. The implementation data is hidden from the user. Inheritance enables sub classes to be defined from a super class. Each sub class has common properties and method in the super class. Polymorphism allows objects to behave differently in different situations. These characteristics offer an advantage in developing a network simulator.

An object orientation produces systems that are easier to evolve, more flexible, more robust, and reusable than procedural approaches. It is a way to develop software by building self-contained modules that can be more easily replaced, modified and reused Furthermore, object technology emphasizes modelling the real world. Hence, it provides a stronger equivalence of the entities in the real world.

From the above comparison, an object-oriented approach is a better choice for creating a network simulator. There are many object-oriented languages available. Since the simulator used in the study is written in Java, in the next section, Java programming language is discussed.

2.2.2.2 Java

Java is an object-oriented programming language developed in Sun Microsystems and was introduced in late 1995. [37] It is an interpreted platform independence programming language. In Java, the source code is translated into instructions by Java compiler then Java Virtual Machine is used to interpret these instructions. [37] Java Virtual Machine consists of an interpreter and a run-time system. The Java compiler generates byte-codes for the Java Virtual Machine (JVM), rather than native machine code. When running a Java program, Java interpreter is used to execute the compiled byte-codes. Since Java byte-codes are platform-independent, Java programs can run on any platform that the JVM is ported to.

Java is a dynamic language. [35] Any Java class can be loaded into a running Java interpreter at any time. It also provides a lot of high-level support for networking. These features enable Java interpreter to download and run code from across the Internet.

Java is designed based on C++ by removing some of C++ features, which are poor programming practices or were rarely used. Hence, it is a simple language that a programmer could learn quickly.

Java provides high robustness because it allows extensive compile-time checking for potential type-mismatch problems. [34, 35] Besides, Java can effectively allocate and reallocate memory in run time. It could prevent occurrence of error during run time.

Java is a high security language. [34] Since Java does not use pointers to direct reference of memory locations, Java has a great deal of control over the code that exists within the Java environment.

Finally, Java supports multiple, synchronized threads that are built directly into the Java language and runtime environment. Synchronized threads are extremely useful in creating distributed, network-aware applications. [35]

2.2.3 Computer Network Simulator

There are various types of network simulators available. Basically, these simulators can be divided into two categories. There are general-purpose simulators and special-purpose simulators. A general-purpose simulator is designed for a wide range of network simulations, while a special-purpose simulator is developed to target a particular area of research. [38] A few available simulators are NIST ATM/HFC Network Simulator [39], cnet [40], Internet Simulated ATM Networking Environment (INSANE) [41], REAL5.0 Network Simulator [42, 43], NS-2 [44], Objective Modular Network Test Bed in C++ (OMNET++) [45], Optimised Network Engineering Tool (OPNET) [46], Parallel Simulation Environment for Complex System (PARSEC) [47], and UMJaNetSim [48].

A simulation tools is needed for this study. In principle, to simplify the process, a simulator is chosen and used as the simulation tool from the existing simulators. Modification and extension of the functionality of the simulator would be done so that a simulation environment could be provided to achieve the objectives of the study.

There are many factors to be considered in order to select a suitable simulation tools for the study. Some simulators have advantages in simulation techniques, programming approach, provision of graphical user interface (GUI) or platform independence. Meanwhile, some simulators are purposely designed for certain area of network research.

UMJaNetSim network simulator is chosen to be the simulation tool in this study based on the requirement of the study and the strength of UMJaNetSim. The main reason to use UMJaNetSim is because UMJaNetSim is a Java-based object-oriented network simulator. Hence, it is much easier and flexible to add in and modify the functionalities in the simulator. Besides, it provides good GUI simulation environment on IP networking. Moreover, the current version of UMJaNetSim (version 5) does not support IP Multicast. Adding a Multicast module in the UMJaNetSim could be a contribution to provide an IP Multicast enabled environment.

2.3 Analysis on UMJaNetSim

UMJaNetSim [48] network simulator is a flexible test bed for studying and evaluating the performance of ATM and MPLS network without building a real network. This simulator could provide simulation for IP-over-ATM.

UMJaNetSim is a network simulator, which is developed based on NIST ATM/HFC Network Simulator [39]. One of the major differences between UMJaNetSim and NIST ATM/HFC Network Simulator is the former uses a pure Java solution and the latter is developed in C. UMJaNetSim is developed in Java programming language for the entire simulation engine, the GUI, and the component development API. Therefore, UMJaNetSim is a fully object-oriented simulator. Hence, UMJaNetSim has high portability among various platforms. It also provides a set of extensible Application Programming Interfaces (APIs) in order to simplify the process of creating new simulation environment. [49] Furthermore, the simulator is readily web-enabled by using an applet version of the simulator. Users can also easily modify and add new components to the simulator without affecting the structure of the simulator.

UMJaNetSim allows user to simulate different network topologies, adjust parameters of each component used in simulation operation, measure network activity, save and load different simulation configuration and log data during simulation execution.

UMJaNetSim provides better GUI that could facilitate simulation process. The other advantage of this simulator is the output performance can be viewed in text based and graphical representation on the screen while the simulation is running.

2.3.1 UMJaNetSim Architecture

UMJaNetSim is developed in JAVA. It is an object-oriented approach network simulator. Basically, all the elements inside the simulator are constructed into a module. UMJaNetSim consists of two parts, the simulation engine and the simulation topology.

The simulation engine is the main controller of the entire simulation. It is responsible for event management task, GUI management task, input and output process handling for simulator and providing assisting tools. [49]

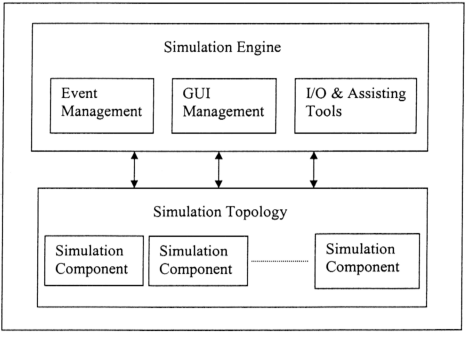

Figure 2.19 Architecture of UMJaNetSim

The simulation topology consists of simulation components, such as including switch or router, physical links, source application and so on. These components are the main subject of the simulation scenario. [49] Simulation process requires the interaction of simulation engine and simulation topology. Figure 2.19 shows the overview architecture of the UMJaNetSim.

2.3.1.1 Event Management Architecture

Event Management handles all the occurrence of event during the simulation process. In the event management architecture, *JavaSim* object is the main object responsible for the simulation process. It is the simulation engine that manages event scheduler, event queue (*SimEvent*), and simulation clock (*SimClock*). Event scheduler resides in *JavaSim* object. [49] Figure 2.20 shows the Event Management Architecture for UMJaNetSim.

During the simulation process, the simulation engine interacts with the simulation topology. A simulation component may schedule an event to happen at a specific time for a target component. *SimEvent* object handles the event by queuing it in a list sorted by the event-firing time. The event in the front of the queue is processed first. When the specific time is reached, the simulation engine invokes the event handler of the target component. Then, the target component will react to the event according to its behaviour. [49] UMJaNetSim allows any event to occur at any time, because it uses an asynchronous approach of the discrete event model.

SimClock object manages the simulation time for the simulator. It is the global time reference used by every component in the simulation and is managed by the simulation engine. [49]

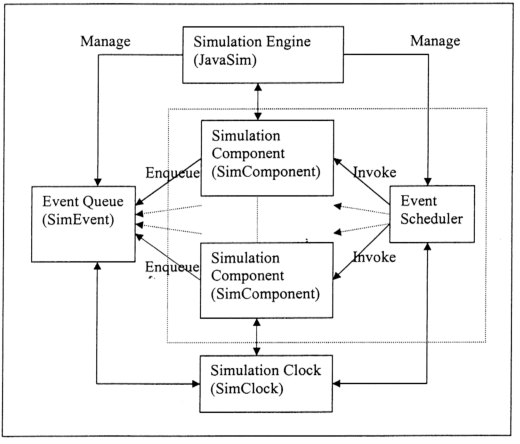

Figure 2.20 Event Management Architecture for UMJaNetSim

2.3.1.2 GUI Management Architecture

GUI Management has several functions. The main functions are user inputs handling, drawing and network topology displays for simulation purpose, and various on-screen windows managing. Once again, the *JavaSim* object is the main controller in the GUI Management architecture for UMJaNetSim. *SimPanel* object is used to keep track of the latest set of simulation components and the interconnection among the components so that simulation topology for a particular simulation could be displayed visually to the user. Users can modify the simulation topology on the GUI at any time during the simulation process. [49]

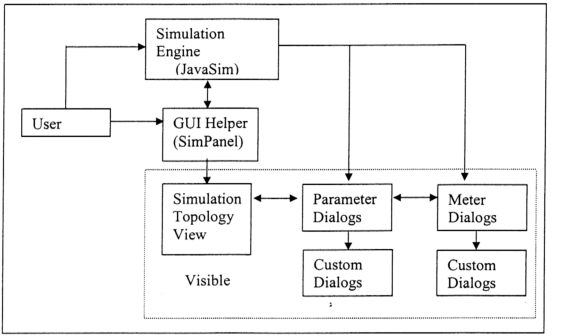

Figure 2.21 GUI Management Architecture for UMJaNetSim

As shown in Figure 2.21, Parameter Dialogs and Meter Dialogs are also located under the GUI Management Architecture. These two dialogs are used to manipulate the external parameters for a component in the simulator and graphically displays output from the simulation process for a component. Each simulation component is associated with a Parameter Dialog. But a Meter Dialog is only assigned to certain components, depending on the need of the component in the simulation. In addition, each Parameter

Dialog or Meter Dialog can create and maintain one or more Custom Dialogs. These Custom Dialogs may be used to show additional information about the simulation [49], such as OSPF routing table, IGMP Membership Table and PIM-DM Table.

2.3.1.3 Simulation Components

Simulation components are the primary objects in UMJaNetSim. The simulation components are derived from *SimComponent* object. *SimComponent* object is a well-defined base object with all the necessary interfaces that enable the interaction between the simulation engine and the simulation component. Each simulation component has a list of external parameters, which allows user-defined properties of a simulation component. *SimParameter* object is used to achieve this purpose. *SimParameter* object is an object that has a well-defined interface with the simulation engine. [49]

An event handler is included in every simulation component. It is a well-defined method in *SimComponent* that accepts event from *SimEvent* object. All interactions between simulation components are achieved through the sending of messages in the form of a *SimEvent* through an event handler. [49] (Refer Figure 2.20)

2.3.2 UMJaNetSim API

In this section, the Application Programming Interface (API) of UMJaNetSim is discussed. API is defined to provide a consistent way of creating simulation components so that it could facilitate design issues in a simulation. [49] The API focuses on defining well-known interfaces for simulation objects such as *JavaSim*, *SimEvent*, *SimClock*, *SimComponent* and *SimParameter*. Only important features of the API are presented.

2.3.2.1 JavaSim

JavaSim object manages event queue, event scheduler and a simulation clock in the event management. It also provides GUI functions in the GUI management. [49] Important methods of the *JavaSim* object are listed in Table 2.2.

Table2.2 Method in JavaSim

Method	Function
java.util.List getSimComponents()	Return a list of all existing *SimComponent*
long now()	Return current simulation time (in tick).
boolean isCompNameDuplicate(String name)	Ensures no duplicated name for *SimComponent*
void notifyPropertiesChange(SimComponent comp)	Executes whenever there are structural changes to the parameters
void enqueue(SimEvent e)	Enqueues an event that has been invoked.
void dequeue(SimEvent e)	Dequeues an event when a specific time is reached for that event to be executed

2.3.2.2 SimEvent

Simulation process involves interaction of simulation components. Each simulation components communicate with each other by passing a message through an event, which is managed by *SimEvent* object. Every invoked event contains an event ID to differentiate the type of occurred event, the event invoked source, the destination of the triggered event, the event occurrence time, and the set of parameters relevant to an event. [49] The set of parameters passed in an event could help a simulation component to take action on the event.

An event can be either private or public, depending on the type of communication. A private event is an event that passes among components of the same type whereas a public event is an event that passes among components of different type. [49] Some of the important methods of the *SimEvent* object are shown in Table 2.3.

Table 2.3 Methods in SimEvent

Method	Function
SimEvent(int aType, SimComponent src, SimComponent dest, long aTick, Object [] params)	Constructor
SimComponent getSource()	Retrieves the source *SimComponent*
SimComponent getDest()	Retrieves the destination *SimComponent*
int getType()	Retrieves the event type
long getTick()	Retrieves the event-firing time
Object [] getParams()	Retrieves the event parameters

2.3.2.3 SimClock

A simulator needs a global time reference. In UMJaNetSim, *SimClock* object provides the global time reference for every simulation components in the simulator. Each occurrence of an event is based on the time in *SimClock* object. In addition, *SimClock* object also defines time conversion method to translate simulation time in microsecond, millisecond or second to tick. A tick is the fundamental unit of time used in the UMJaNetSim. A tick is equal to 10 nanoseconds. [49]

2.3.2.4 SimComponent

In UMJanetSim, there are a number of simulation components. Some of these essential simulation components are switch, router, broadband terminal equipment (BTE), link, and various type of application. In UMJaNetSim, *SimComponent* object is an important object that defines those simulation components. Every simulation component in the simulation must inherit *SimComponent* object. [49] *SimComponent* provides the fundamental structure for a simulation component. Each simulation component contains a list of simulation parameters. Listed in Table 2.4 and Table 2.5 are a few important properties and methods for *SimComponent*.

Table 2.4 Properties in SimComponent

Property	Description
protected transient JavaSim theSim	A reference to the main *JavaSim* object
protected java.util.List neighbors	A list of all (directly connected) neighbors of the *SimComponent*
protected java.util.List params	A list of all external parameters of the *SimComponent*

Table 2.5 Methods in SimComponent

Method	Function
SimComponent(String aName,int aClass,int aType, JavaSim aSim,Point loc);	Constructor Every new component must provide a constructor with exactly the above parameters
Object [] compInfo(int infoid,SimComponent source, Object [] paramlist)	Provides a way for inter-component information exchange without sending run time events
boolean isConnectable(SimComponent comp)	Called by the simulation engine when a new component is about to be connected to this component. This method verifies whether the new component can be connected to this component
void addNeighbor(SimComponent comp)	Called by the simulation engine when a new component is connected to this component
void removeNeighbor(SimComponent comp)	Called by the simulation engine when a new component is disconnected to this component
void copy(SimComponent comp)	Used to copy parameter values from another *SimComponent* of the same type
void reset()	Performs a reset operation in order to bring the status of the component back to the same status as if it is just newly created

void start()	Performs any operations needed when the simulation starts
void resume()	Performs any operations needed when the simulation need to be resume. One possible use is to capture any special changes /that have been done by the user during the pause period
void action(SimEvent e)	An event handler of this component, and will be called by the simulator engine whenever a *SimEvent* with this component as the destination fires

2.3.2.5 SimParameter

There are internal parameters and external parameters in every *SimComponenet*. An internal parameter is accessible only by the simulator while an external parameter is accessible by the user. [49] These parameters are used to generate a simulation.

Table 2.6 Important Overrides in Simparameter

Method	Function
SimParameter(String aName,String compName, long creationTick,boolean isLoggable)	Constructor
String getString()	Returns a String representation of the parameter value for logging purpose
void globalSetValue(String value)	Supports setting of the same parameter values for multiple components in one command
int getValue()	Reads a value
void setValue(int val)	Writes a value
JComponent getJComponent()	Returns a Java swing component and its name

External parameters are inherited from *SimParameter* object. [48] Since *SimParameter* object provides parameter logging and meter display, every external parameter also has these features. Every inherited external parameter must have the name of the parameter, the name of the component that owns the parameter, the time when the parameter is

created and the information on whether the parameter can be logged in the log file. Besides the fundamental value, other values can be added according to the requirement of each external parameter. The important overrides for *SimParameter* are listed in Table 2.6.

2.4 Summary

This chapter provides a detailed description of IP Multicast, computer simulations, and the UMJaNetSim network simulator. IP Multicast basic operation, multicast address, IGMP, multicast forwarding algorithms, and various types of multicast routing protocols are included in the first section of review. A brief description on the computer simulation and programming approach used in computer simulation is discussed in the review on computer simulations. Reasons of choosing UMJaNetSim as the network simulator for this thesis are also listed. Finally, the architecture and API of the UMJaNetSim are concentrated in the analysis of the simulator itself.

The following chapter analyses IGMP and the PIM-DM.