Multicast Routing on the Internet

Contents

1.Introduction

In recent years, multicast transmission has become more and more popular. There are a lot of applications that require intra- or internetwork multicasting. Examples include video and audio sharing (video/audio conferencing, real time video distribution), network reconciliation (such as synchronization protocols), software update distribution, web caching (temporary storage of web objects such as HTML documents), web clustering, and resource discovery.

Reverse Path Multicasting is one of the most prevalent algorithms that are used today. The primary goal of this project has been to simulate a simplified version of the Reverse Path Multicasting technique in a network where both unicast and multicast forwarding are possible.

2.Theoretical background

2.1 What is Multicast?

Multicast is a technique for the efficient distribution of identical packet streams to groups of selected hosts on one or more local area networks [1]. Simply, Multicast is a point-to-multipoint transmission.
There several features related to this type of transmission:

- **Individual hosts are configured as members of multicast groups.** Hosts can join and leave a multicast group at any time. There are no restrictions on the physical location or the number of members in a group .One host can be a member of more than one group at any given time and doesn't have to belong to a group to send messages to the group members.
- **Multicasting is not connection oriented.** Packet loss and out of order delivery is possible.
- **An IP multicast group is identified by class D address.** The most significant four bits of class D addresses are set to "1110" (Figure 1). The 28-bit number following these four bits is called "Multicast Group ID". All members of one particular group are assigned the same class D address [2,3].

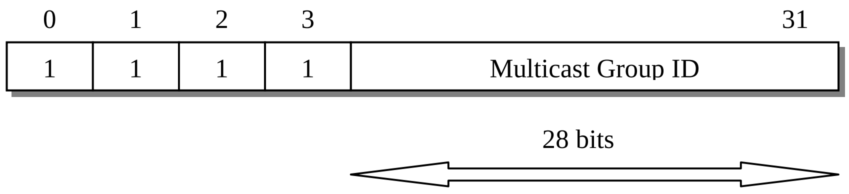


Figure 1. Class D address format

2.2 Multicast IP Delivery Service

A simple picture characterizing a Multicast IP Delivery Service is represented in Figure 2. The Protocol through which hosts can communicate with their immediately neighboring multicast routers is called the Internet Group Management Protocol. A Multicast Routing Protocol runs between the multicast routers. It is responsible for the construction of multicast delivery trees and forwarding of multicast datagrams across an internetwork.

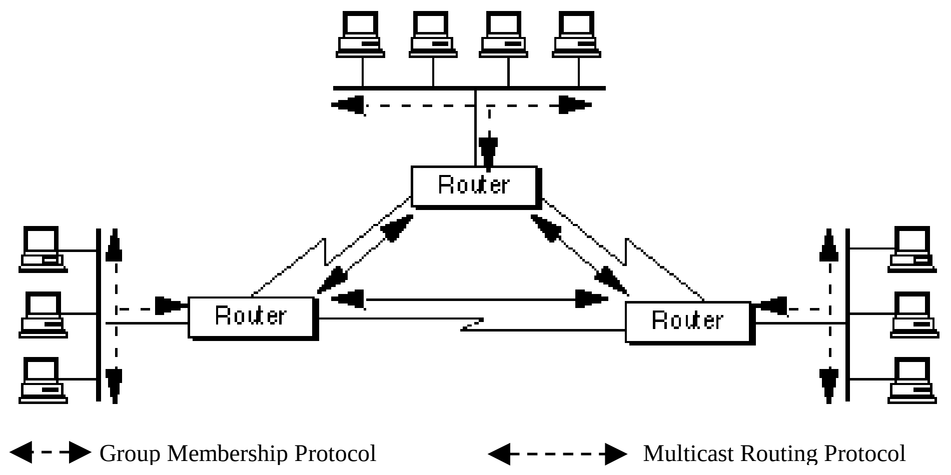


Figure 2. Multicast IP Delivery Service [2]

2.3 Internet Group Management Protocol

The Internet Group Management Protocol (IGMP) runs between hosts and their immediately neighboring multicast routers. The main properties of this protocol are listed below[1]:

- **A multicast router periodically transmits Host Membership Query messages** on its directly attached networks to check whether the known group members are still active (Figure 3).

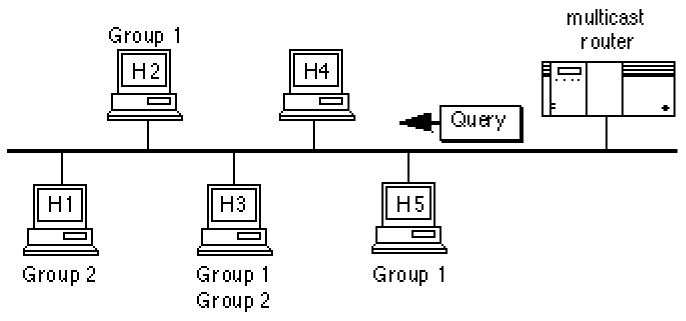


Figure 3. Internet Group Management Protocol

After a host receives a Query message, it responds with a Host Membership Report for each host group to which it belongs**.** In order to avoid a flurry of Reports, each host starts a Report delay timer for each of its group memberships. If more than one Report were heard during the delay period, the local host resets its timer to a new random value. Otherwise, the host transmits a

[1] Here we are talking about version 2 of the IGPM

Report to the reported address, causing all other members of the group to reset their Report message timers [2].

- When a host wants to join a multicast group it immediately transmit a Host Membership Report to the local router without waiting a Query.

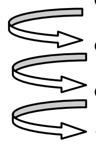
- Election of the querier (the router with the lowest IP address) is supported when there is more than one router in a given network.

- A router is also able to transmit Queries to one specific multicast group (a Group-Specific Query Message).

- When a host wants to leave a group, it responds with a Leave Group message to a Query.

2.4 Multicast Forwarding Algorithms

There are several different algorithms that may potentially be employed by multicast routing protocols:



- Flooding
- Spanning Trees
- Reverse Path Broadcasting
- Truncated Reverse Path Broadcasting
- ✓ Reverse Path Multicasting
- ✓ Core-Based Trees

The first three of them, Flooding, Spanning Trees and Reverse Path Broadcasting, do not take into account multicast group membership. They just give a possibility to reach every node in a given network and don't pay attention to the fact that datagrams can be forwarded to subnetworks, which have no members in the destination group. We studied all these algorithms in details and decided to choose the last two for our project. As you can see from the evolution, represented here by arrows, Reverse Path Multicasting is the last enhanced version of the simple Reverse Path Broadcasting algorithm. As every algorithm, Reverse Path Multicasting has some drawbacks. Core-Based Trees algorithm has been created to overcome Reverse Path Multicasting limitations. One of our project goals (it will be described in details later) is to compare these two algorithms. Both of them are implemented in two of the most prevalent multicast routing protocols, which are used today. Reverse Path Multicasting is used in both Distance Vector Multicast Routing protocol and Dense Mode of Protocol Independent Multicast. Sparse Mode of the latter protocol was created on the base of Core-Based Trees algorithm [2,3].

2.5 Reverse Path Multicasting (RPM)

How does the Reverse Path Multicasting algorithm work?
Every time a router receives a packet on one of its links, it checks whether or not it can consider this link as belonging to the shortest path towards the source. If yes, it forwards the packet to on all links except the incoming one. Otherwise, it simply discards the packet and "poison reverses" the link. We can see how it happens on the example, represented in figure 4. Router C will forward packets received from A to all its directly connected networks and will "poison reverse" the link from B, because B doesn't belong to the shortest path from C to A. "Poison reverse"

means that C will send a message to B saying that there is no need to forward packets onto link B-C. This is what the Reverse Path Broadcasting algorithm does. The interface over which a router expects to receive multicast packets from a particular source is called the “parent” link. The “child” links are the outgoing links over which the router forwards multicast packets.

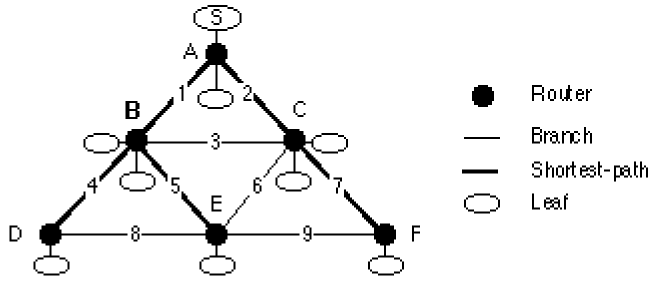


Figure 4. Reverse Path Broadcasting [2].

The Internet Group Membership protocol gives to a router a possibility to determine whether it has any group members on its directly connected subnetworks. The router avoids sending packets to a subnetwork that has no members in the destination group. This is the way Truncated Reverse Path Broadcasting works. It was improved after some time and the improved version was called Reverse Path Multicasting. A router has become able to transmit a “prune” message on its parent link saying that there is no need to send messages to him for a particular multicast group in the case when it doesn’t have any members of that group on its directly attached networks. As it can be seen from figure 5, “prune” messages are sent only one hop back towards the source.

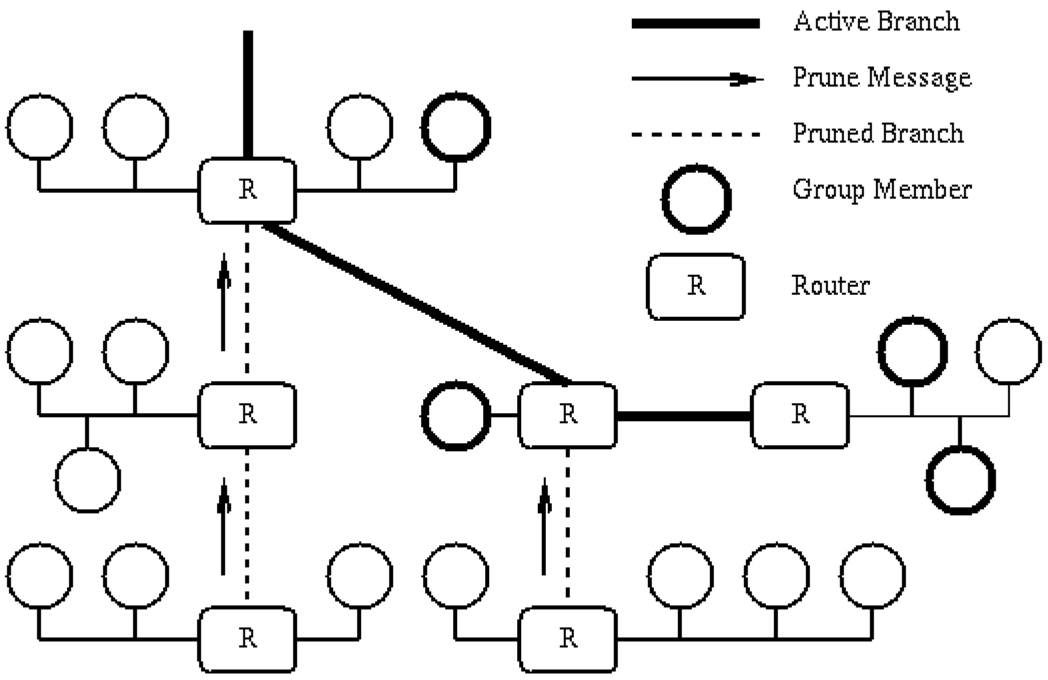


Figure 5. Reverse Path Multicasting delivery tree [3].

After receiving a “prune” message on its child link, a router can transmit its own “prune” message to the parent if there are no group members on its network. As a result, a source-rooted, shortest path tree for each (source, group) pair is produced.
This algorithm seems to be efficient and easy to implement. Furthermore since the packets are forwarded through the shortest path from the source to the destination nodes, it is very fast. But it still has **two essential drawbacks**. Multicast packets are still periodically forwarded to all routers in a given network, because “prune” information is removed from routers memory from time to time. The purpose is to refresh the “prune” state of delivery tree. Since multicast is not connection oriented, a refresh operation should be performed regularly. The algorithm is not scalable, especially in the case of very large internetworks, because a lot of memory space is required to store information about all (source, group) pairs.

2.6 Core-Based Trees (CBT)

As we are going to compare Reverse Path Multicasting efficiency with this algorithm efficiency, we decided to include a short description of this algorithm into the Final report, although it would be simulated in our program. We simulated only mechanisms required for unicast packet forwarding, which is used by CBT, as it is described below.
Instead of building a delivery tree for each host, CBT separates the entire network into many different *multicast groups* and develops a single delivery tree for each group. Within the group there is a *core router*, the joining node for the receivers and the sources. All other routers in the multicast group are known as *non-core routers* (Figure 6).

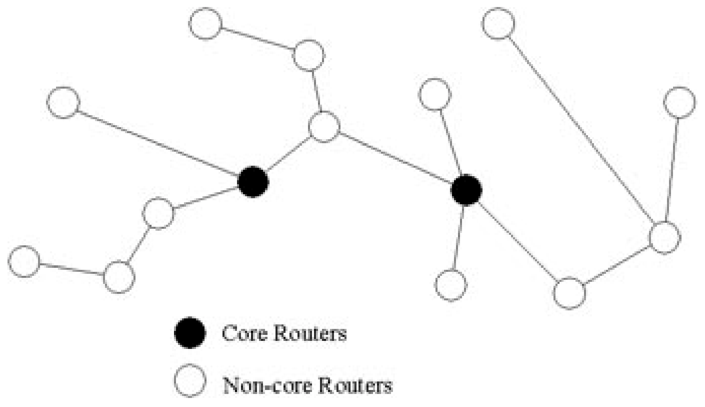


Figure 6. Example of a Core Based Tree.

One of the major features of CBT is its *tree creation*. A node will not be considered as part of a tree unless it decides to join a multicast group. When it desires to receive traffic for a particular multicast group, it sends a join request message to the core router of that particular group. In this case only the address of the group’s core router is required. The joining process is known to be successful if a *join* acknowledgement message is received from the core router. Likewise, if a node desires to quit a multicast group, it sends a quit request message to the core route of that group [4].
Another major feature is *unicast routing separation*. Multicast routing in CBT are decoupled from unicast routing algorithms. That is, broadcasting to a particular multicast group may be

achieved merely by transmitting data from a source to one single core router via the shortest path. In this case fewer processes and computations are required.
There are, however, **a few limitations**. First of all, CBT generate heavy traffic near the core routers since packets from all sources to all receivers within the multicast group travel through the same path and through the same core router. Also, all transmissions are directed to the core routers. But the path through the core routers might not be the shortest path between the source and the receiver. Therefore, unnecessary delays might result.

3. Project Goals

- Simulation of unicast forwarding since in real life both unicast and multicast communication pattern are used in the same network
- Simulation of Reverse Path Multicasting using a special data structure to store the information
- Evaluation of the algorithm efficiency in the case of the large internetworks (up to 10000 nodes)
- Comparison with another multicast forwarding algorithm, Core-Based Trees, which was preliminary created to overcome the limitations of Reverse Path Multicasting

4. Implementation details

The code for the given project has been written on C++. The description of the code, including all the assumptions, which have been made during the creation, is represented in the next two sections.

4.1 Data structure

Each node in a graph, which can be a router or a host, is assigned a unique identification number. Costs of links are always greater then zero. A simple topology is represented in Figure 7.

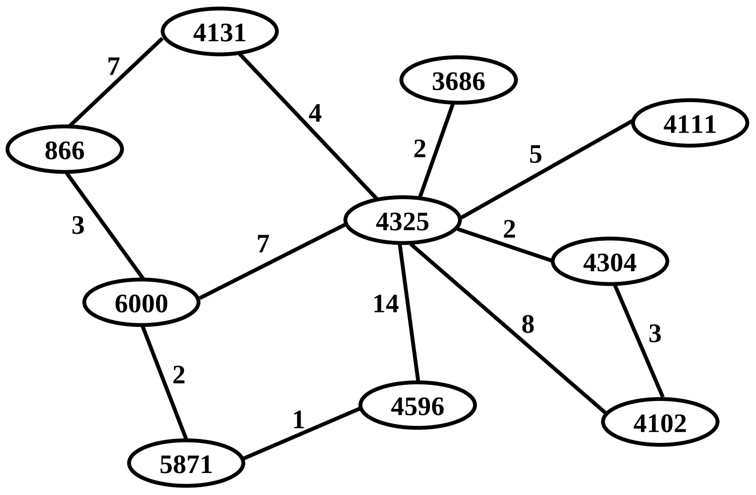


Figure 7. A simple network topology.

We decided not to use IP addresses to facilitate our simulation. Group addresses are also specified by numbers, such as 55555, 66666, and 77777.
A graph is stored as a linked list of HEADs, where each HEAD stores its own linked list of NEIGHBORs. Figure 8 shows how to store the simple graph represented below. The most left column is a linked list of all nodes in the graph; it doesn't matter whether or not they connected to each other. It's just a way to store information. Each node has a list of its directly connected members. For example, node 4111 has only one neighbor, node 4325. The main reason we chose such a data structure is that nodes should be inserted and deleted dynamically, and this data structure allows to do it without any problems.

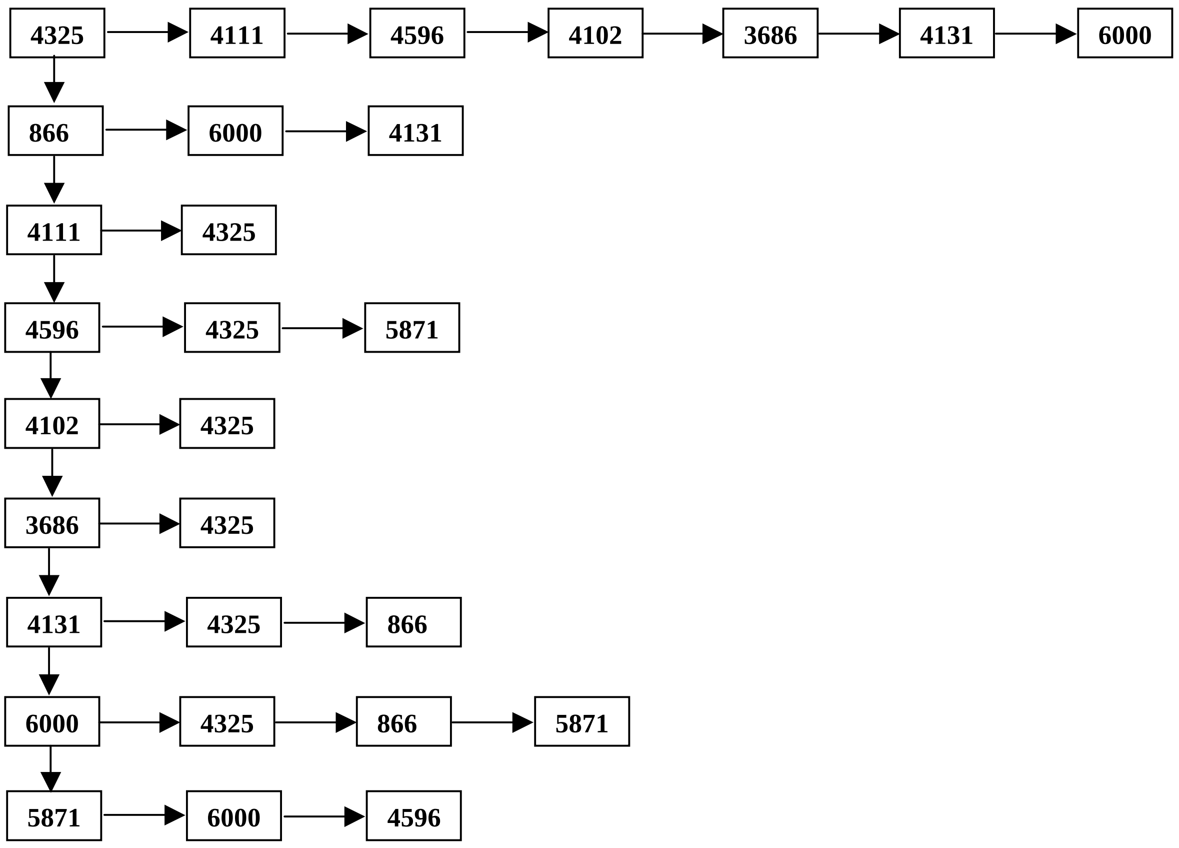


Figure 8. A data structure that is used to store the topology, represented in Figure 6.

Information about multicast groups is stored in the similar data structure (Figure 9). The most left column is a linked list of all current multicast sessions (group Ids). Each group stores its members as a linked list.

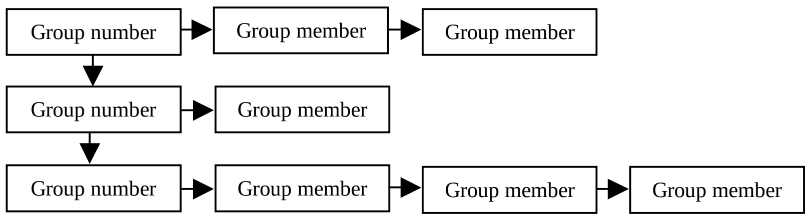


Figure 9. A data structure that is used to store information about multicast groups.

4.2 Simplification and Design Assumptions

The following events have been supposed to happen in both multicast and unicast simulations:

- Initialization
- A LSP arrival
- Forwarding of an unicast packet
- Advertisement of a new multicast session
- A join multicast group message
- A quit multicast group message
- Multicast packet forwarding

The events are written in special data files, each event begins with a new string. The code is able to read information from a data file and distinguish between different types of events. After the type of event is defined, all necessary parameters corresponding to that event are written for a while into special class EVENT. Then the program begins to process the event. After the processing is complete it reads the next event from a data file until it reaches the end. Every event is characterized by the time of creation, timestamp, which is always present at the beginning of a string. The timestamp is followed by special abbreviations indicating on the type of an event. The node id is written after the type. It may be the id of a node which was an event initiator, for example, a node which sent a join multicast group request message, or it may be a node to which a unicast packet should be forwarded. Other parameters differ and depend on the type of an event.

The first event, **Initialization**, is encountered in the data file only once. It is needed to start up a network graph, in which it will act as a unicast and multicast router. Every node in a graph is supposed to know the information about its directly connected neighbors. We assume that all LSP can be received correctly by the router and don't take into account how the costs of links are calculated.

The second event, **a LSP arrival**, is used to create and dynamically change a topology. According to state-link routing mechanism, described in [5] in details, a link state packet in our simulation contains the following information:

- the timestamp of the packet
- the ID of the node that created this packet
- a sequence number
- a list of directly connected neighbors, with the cost of the link to each one

The program keeps track of the largest sequence number for each host and discards LSP packets with a smaller or equal value of a sequence number. If a packet is not discarded the router stores it and updates the current data structure. The program is able to fix two special cases related to a LSP arrival event:

1) A Graph HEAD has fewer neighbors than the arrived LSP advertises. Lack neighbors are added.
2) A Graph HEAD has more neighbors than the arrived LSP advertises. Extra neighbors are deleted.

The third event, **Unicast Packet Forwarding**, requires knowledge of the shortest path from the router to a node to which a packet should be forwarded. The program uses Dijkstra's algorithm

[6] to compute the shortest paths from the router to all other nodes. Dijkstra’s algorithm is run only when it is necessary: if the topology of the graph has changed and a new forwarding request arrives. As a result, the forwarding tables are created. We omit how to forward a given packet to the destination and simply print out a message about unicast forwarding to a particular node and display all the information, related to this event, including current graph topology, shortest paths and forwarding tables.

The forth event, **Advertisement of a new Multicast Session**, is used to announce about subscription to a new multicast group. In our simulation a node, which initializes this event, is considered as the first member of this new group. Time-to-live for each new multicast group is specified in this event and checked regularly during a code run. For example, when multicast forwarding is required, the code at first checks whether the time-to-live for the destination group is expired, comparing a TTL value with the time of creation of the multicast forwarding event. When the time-to-live is expired, the code simply removes this group number and all its members from the router’s memory.

The next two events, **Join and Leave Group Request Messages**, are similar to each other. We implemented a simplified version of the IGMP, where we pay attention only to the fact of a message arrival and omitted transmittance of Host Membership Queries and Reports. Either we didn’t take into consideration elections of the querier, because all these facts wouldn’t make any significant impact on the evaluations, which we were going to do. When the router gets a join or leave request, at first it checks whether it is aware about such a group. If it doesn’t know such a group, it discards the packet. If it is aware about this group, but the time-to-live is expired, it discards the packet and removes reference to the group and all its members from the memory. Also it is able to fix the situation of a duplicate request checking ids of group members before writing a node that has sent a request.

The last event, **Multicast Packet Forwarding**, has been simulated using the Reverse Path Multicasting technique. After receiving a packet for a particular multicast group, the router checks whether it is aware about this group. If not, it discards the packet and print out a corresponding message on a screen. If yes, the program computes the shortest path from each group member to the sender. The packet is forwarded only if the shortest path back to the source goes through the router. The program arranges a list of hops through which the router intends to send a packet to the destinations. When forwarding occurs, a special message about it appears on the screen. If the shortest path from any node to the sender doesn’t go through the router, the packet is not forwarded to this node. It is supposed, that any other router, which is located on the shortest path, will take responsibilities of forwarding the packet to the node. With a message of a multicast forwarding we display information of all current multicast groups. It is also should be mentioned that we didn’t specify transmittance of “prune” messages. They could make our simulation look better but they would not influence on the evaluations.

The code is represented on our Web page. We added comments that would explain the necessity of creation of all the functions we used. We provide also a simple file to demonstrate how the code works. This file contains the information required to build the topology, represented in Figure 7.

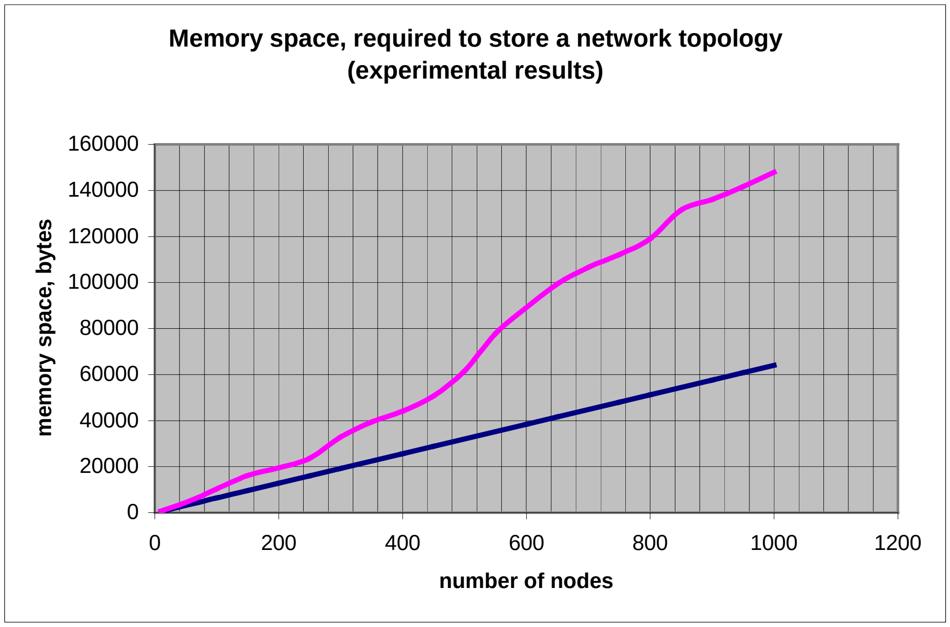
5. Analysis of the simulation

5.1 Comparison of the two algorithms

In terms of scalability, Core Based Trees algorithm has an advantage over Reverse Path Multicasting. CBT has a scaling factor proportional to the total number of multicast group in the network. While with source-based trees, built as a result of RPM, the scaling factor is proportional to the total number of (source, group) pairs in the network. It can be easily explained as following. In the case of a source-based tree, maintenance of information for each (source, group) pair is required. The scaling factor of each network group is equal to the total number of active nodes within that group. If there are *M* groups in the network and each group contains an average of *m* active nodes, the scaling factor of a source-based tree will be $M \bullet m$ while for the CBT it is just *M*. Furthermore, unlike source-based trees, CBT does not require multicast messages to be periodically forwarded to all multicast routers within the network since refreshing the connections of the entire network is not necessary.

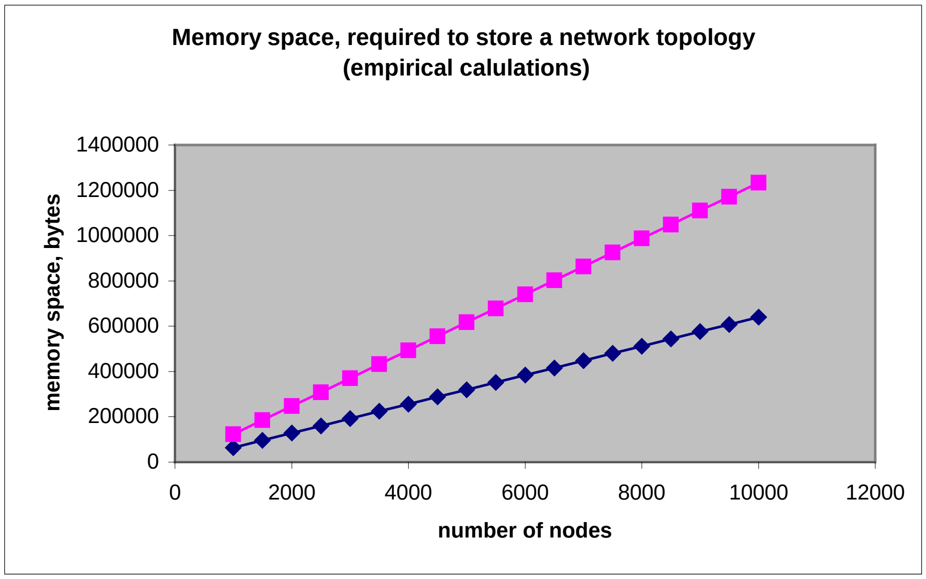
5.2 Mathematical analysis

Since both algorithms are related to unicast forwarding (CBT uses it and RMP is used in a networks where it works), we will consider at first how much memory space is required to store general information about the graph or that information the unicast router should keeps. If a given graph consists of N nodes, then the total required memory space is equal to (N*40+12*n), where n is the total number of NEIGHBOR structures. Assuming N=10000 and n=(N*(N-1))/2, we get that about 50 MB is needed only to store a topology, which is significant in comparison with memory resources available today. Such a number of n is interesting to evaluate, because 2*n is the upper bound to a number of NEIGHBORs of a graph containing N nodes. We did evaluations of a graph, which includes up to 1000 nodes. We provide several graphs, which can illustrate our evaluations. The upper curve in Graph 1 designates an experimentally calculated



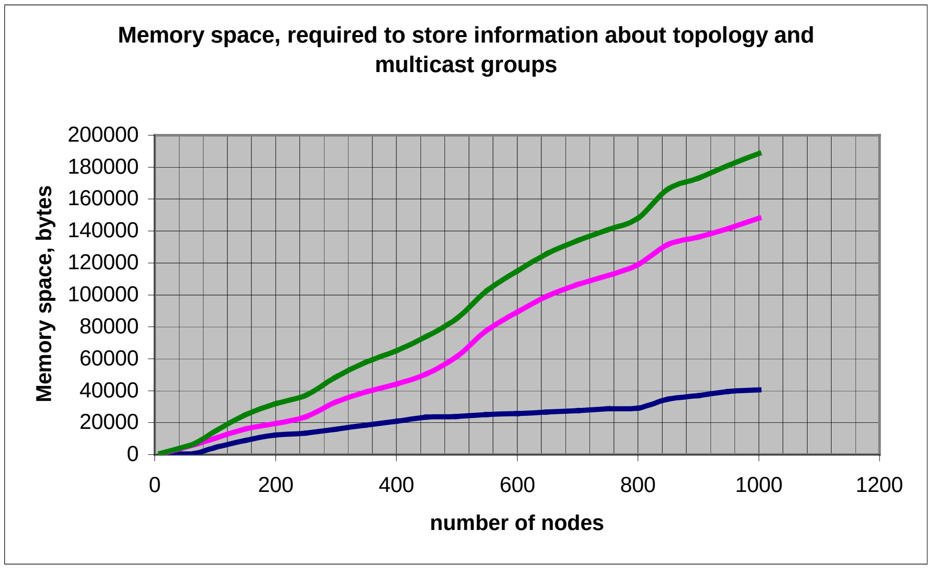
Graph 1. Memory space, needed to store a network topology.

memory space. The lower curve is the lower limit of memory. It corresponds the situation, when each node in a graph is connected only with one particular node. Since this graph is useful to a particular graph, which is created in the simulation, we did only empirical evaluations for the case of large networks. We calculated the average value of N/n, where N is the number of nodes and n is the total number of NEIGHBOR structures and then used this structure to calculate an empirical curve, represented in Graph 2.



Graph 2. Memory space, needed to store a network topology in a case of large networks.

We also made evaluations related to storage of multicast group information. It depends on a number of group and their members. The case corresponding to our graph is represented in Graph 3. . The red curve corresponds to the amount of memory, required to store the information related to multicast groups. The blue curve corresponds to unicast information and the green curve represents total sum.



Graph 3. Memory space, required to store information about topology and multicast groups

CBT algorithm can be illustrated by red curve. The deviation will be insignificant, if we suppose we have enough core-routers in a given network. RPM is characterized by the green curve.

6. Conclusions

Implementation of the project goals required deep knowledge of different conceptions and algorithms (the shortest path, LSP, Dijkstra algorithm, RPM), which we gained creating the simulation. We significantly improve our abilities in simplifying problems and making reasonable assumptions. The project also helped us to improve our programming skills.

7. References

1. http://www.otc.psu.edu/policies/multicast.html
2. http://www.icis.ntu.edu.sg:8000/top_pages/useful.htm
3. http://cs-people.bu.edu/guin/multi.html
4. http://community.roxen.com/developers/idocs/rfc/rfc2201.html
5. Larry L. Peterson & Bruce S. Davie. Computer Networks. A system approach.
6. T.Standish & Adisson Wesley. Data structures, Algorithms and Software Principles in C.